# Praktikum Automatisierungstechnik

Wintersemester 2016/2017

Protokoll zum

## Versuch 5: 3-Tank-System

der Gruppe 5


Xiaoyu Xie
Shaochen Qian
Jun Lou


Tag der Versuchsdurchführung: 24 Januar 2017
31 Januar 2017

Hiermit versichern wir, dass wir dieses Protokoll selbstständig angefertigt haben.
Karlsruhe,

# Inhaltsverzeichnis

# 1. Einleitung und Versuchsaufbau

## 1.1 Einleitung

Die klassische Regelungstechnik basiert auf der modellbasierten Analyse und Synthese von Systemen, wofür ein mathematisches Modell des betrachteten Systems notwendig ist. Wenn eine analytische Modellbildung oder eine Systemidentifikation aufgrund der Komplexität des Systems oder aus wirtschaftlichen Gründen nicht möglich ist, werden wissensbasierte Methoden wie Fuzzy-Regler oder künstliche Neuronale Netze zur Reglung eines Systems eingesetzt.

**Ziele des Versuchs**

- Entwurf eines Neuronalen Netzes zur Approximation des Regler „Mensch“
- Entwurf eines Fuzzy-Reglers
- Vergleich und Einschätzung der entworfenen Regler

## 1.2 Versuchsaufbau

Die Anlage besteht aus drei zylinderischen Behältern, die durch Ventile miteinander verbunden sind. Die beiden äußeren Tanks können mit jeweils einer Pumpe befüllt werden. Die einstellbaren Volumenströme der Pumpe liegen zwischen 0 und 6 l/min. Bis auf die Ventile V1-3u, V3-2u und V0 sind alle Ventile geschlossen. Das Ventil V0 am rechten Tank dient als Abflussventil der gesamten Anlage. Mit dem Öffnen der Ventile, die normalerweise geschlossen sind, können während der Reglung Störgrößen auf die Strecke gegeben werden.

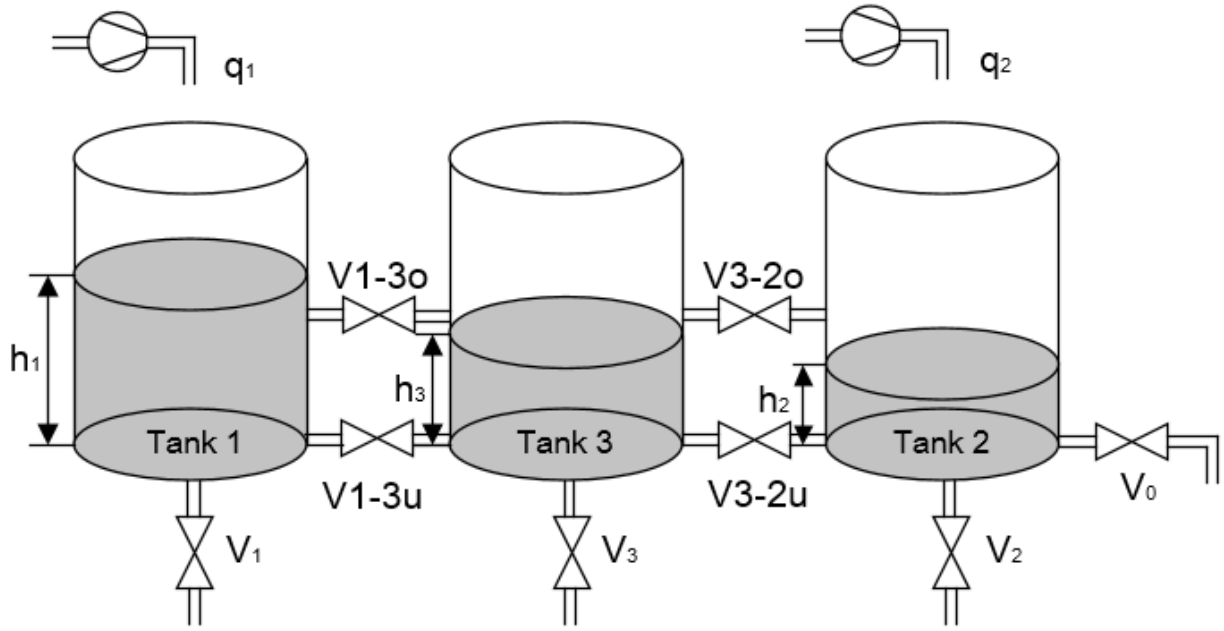


Abbildung 1.1 Aufbau des 3-Tank-Systems

Das 3-Tank-System ist über eine dSPACE-Karte an einen PC angeschlossen. Dadurch ist es möglich, die gemessenen Füllstände der Tanks und die Pumpendurchflüsse in den

Rechner einzulesen und den Regler als Software im PC zu realisieren. Vom PC aus können über die dSPACE-Karte außerdem die beiden Pumpen angesteuert werden.

# 2. Theoretische Grundlagen

## 2.1 Künstliche Neuronale Netze

Künstliche Neuronale Netze werden oft als Versuch dargestellt, die Leistungsfähigkeit des menschlichen Gehirns auf Hard- und Software umzusetzen, um mithilfe dieser Systeme die gleichen Eigenschaften anzunehmen wie ihre biologischen Vorbilder. In der Automatisierungstechnik werden künstliche Neuronale Netze zur Modellierung und zur Regelung komplexer Prozesse eingesetzt.

### 2.1.1 Strukturen der Neuronalen Netze

### Das Perceptron

Das Perceptron besteht aus Neuronen, die in zwei Schichten angeordnet sind. Ein typisches künstliches Neuron, wie es in der Abbildung 2.1 dargestellt ist, ist eine mathematische Vorschrift zur Berechnung eines Ausgangswertes $o_k$ aus einer Reihe von Eingangswerten $e_j$ :

$$o_k = g(a_k) = g\left(\sum_{j=1}^{n} w_{kj} e_j + \vartheta_k\right) \quad ,$$

dabei ist:

- $e_j$ Eingangswert (input)
- $w_{kj}$ Gewicht (weight) von Eingang j zu Neuron k
- $\vartheta_k$ Schwellwert (threshold) des Neurons k
- $a_k$ Aktivierung (activation) des Neurons k
- $g(a_k)$ Aktivierungsfunktion (activation function)
- $o_k$ Ausgangswert (output) des Neurons k

Die Struktur eines einschichtigen Neuronalen Netzes ist in der Abbildung 2.2 dargestellt. Die Eingangsneuronen sind keine Neuronen im eigentlichen Sinn, da sie ihren Ausgangswert nicht über eine Aktivierungsfunktion erzeugen. Sie dienen lediglich als „Verteilungspunkte“ der Eingangsgrößen an die Neuronen der nächsten Schicht. Damit ist das Perceptron ein einschichtiges Neuronales Netz. Hier ist die Aktivierungsfunktion eine Treppenfunktion.

Aber nur linear separable Problem können vom Perceptron gelöst werden, was eine zu große Einschränkung für die praktische Anwendung darstellt. Diese Einschränkung kann jedoch überwunden werden, wenn man eine zusätzliche Schicht mit Neuronen einführt.

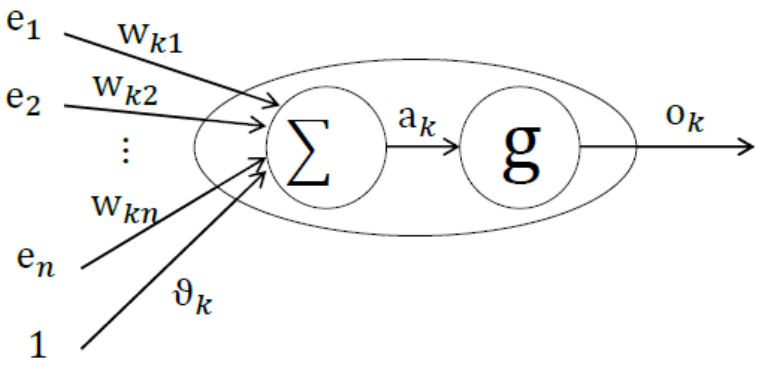


Abbildung 2.1 Typische Realisierung eines künstlichen Neurons

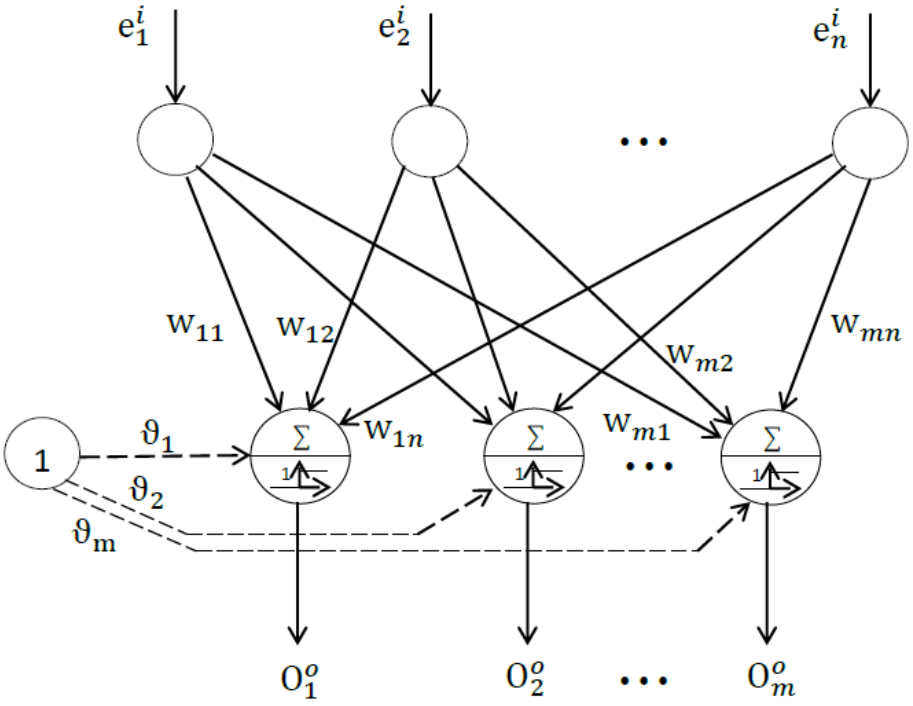


Abbildung 2.2 Struktur des Perceptrons (einschichtiges Neuronales Netz)

## Aktivierungsfunktion

Um die Aktivierung $a_k$ eines Neurons in eine Ausgangsgröße zu verwandeln, gibt es verschiedene Aktivierungsfunktionen. Einige häufig eingesetzte Aktivierungsfunktionen sind in der Abbildung 2.3 – 2.5 dargestellt.

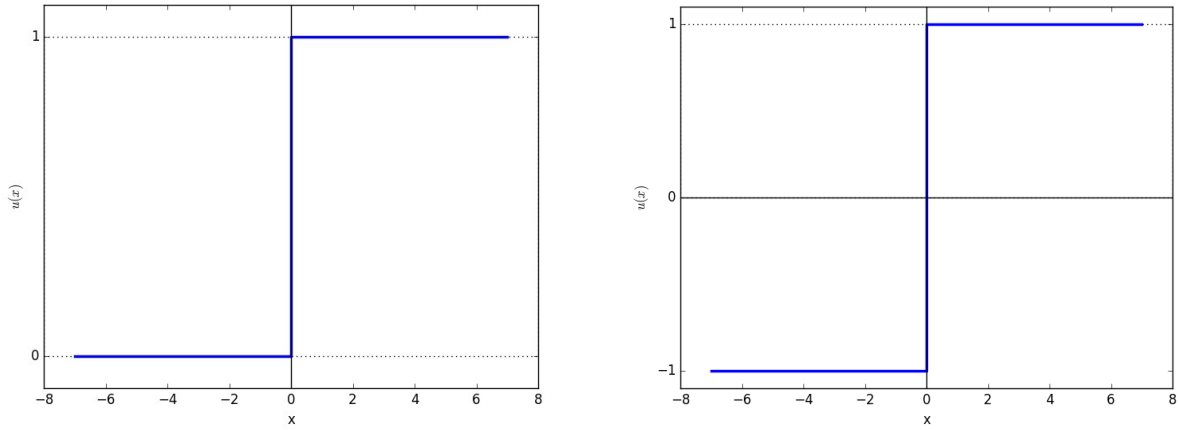


Abbildung 2.3 Treppenförmige Aktivierungsfunktion

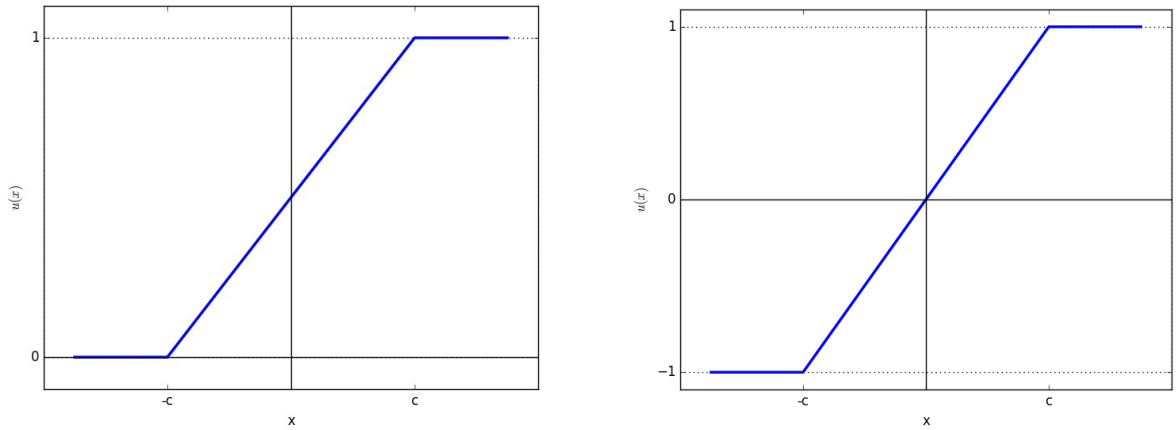


Abbildung 2.4 Semi-lineare Aktivierungsfunktion

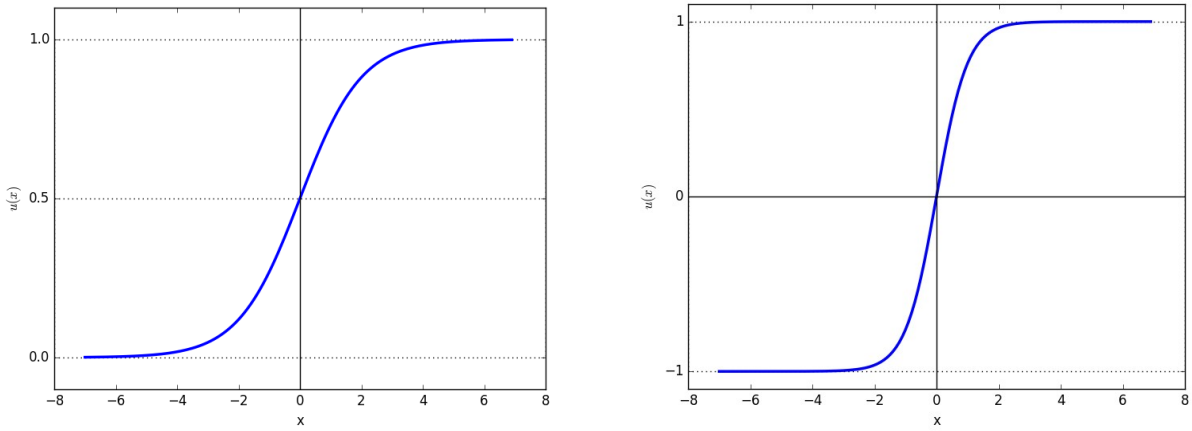


Abbildung 2.5 Sigmoid-Funktion und tanh Funktion

**Treppenförmige Aktivierungsfunktion**

Neuronen mit treppenförmiger Aktivierungsfunktion sind einfach in Hardware umzusetzen, da sie jedoch nur zwei mögliche Ausgangswerte haben, sind sie in ihren Fähigkeiten begrenzt.

**Semi-lineare Aktivierungsfunktion**

Die semi-lineare Aktivierungsfunktion ist für Eingangswerte zwischen -c und c linear. Für Aktivierungen größer als c nimmt sie den konstanten Wert 1, für Werte kleiner als -c den konstanten Wert 0 bzw. -1 an. Damit ist die Funktion stetig, aber nicht differenzierbar. Nimmt die Aktivierung nur Werte zwischen -c und c an, kann die Aktivierungsfunktion durch eine lineare Funktion ersetzt werden, was gelegentlich bei Ausgangsneuronen angewendet wird.

**Sigmoidale Aktivierungsfunktionen**

$\tanh(a_k)$ oder die sogenannte Sigmoid-Funktion

$$f(a_k)=\frac{1}{1}+e^{-a_k}$$

sind beispielsweise Sigmoidale Aktivierungsfunktionen. Die Sigmoid-Funktion ist im gesamten Definitionsbereich stetig differenzierbar und bietet für den später vorgestellten Backpropagation-Algorithmus den Vorteil, dass sich die Abteilung der Funktion durch die Funktion selbst ausdrücken lässt:

$$\begin{aligned} f'(a_k) &= \frac{-1}{(1+e^{-a_k})^2}(-e^{-a_k}) \\ &= \frac{1+e^{-a_k}-1}{(1+e^{-a_k})^2} \\ &= f(a_k)-f(a_k)^2 \\ &= f(a_k)(1-f(a_k)) \end{aligned}$$

So wird daher oft in den verdeckten Schichten des mehrschichtigen Perceptrons verwendet.

## 2.1.2 Multilayer-Perceptron

Das Multilayer-Perceptron (MLP) ist die verallgemeinerte Form des zweischichtigen Perceptrons. Es besteht aus einer Eingangsschicht (Index i). Die Zahl der Eingangsneuronen ist in statischen Netzen gleich der Zahl der Eingänge. Danach kommen üblicherweise eine bis zwei Zwischenschichten, die sogenannten verdeckten Schichten (hidden layer). Die Zahl der Neuronen in diesen Schichten hängt von der Komplexität der zu approximierenden Funktion ab. Die letzte Schicht ist die Ausgangsschichte (Index o). Die Zahl der Ausgangsneuronen ist gleich der Zahl der Ausgänge. Die Struktur des MLP ist in der Abbildung 2.6 dargestellt.

## 2.1.3 Training eines Neuronalen Netzes

In einem Neuronalen Netz stehen verschiedene Parameter zur Verfügung, die geeignet gewählt werden müssen, damit das Netz die gewünschten Ausgangswerte zu gegebenen Eingangswerten liefert. Die Optimierung dieser zunächst zufällig belegten Parameter mit dem Ziel einer möglichst guten Funktionsapproximation nennt man Training. Bei den bisher vorgestellten Netztypen sind die Parameter die Gewichtsfaktoren $w_{kj}$ und die

Schwellwerte $\vartheta$ .

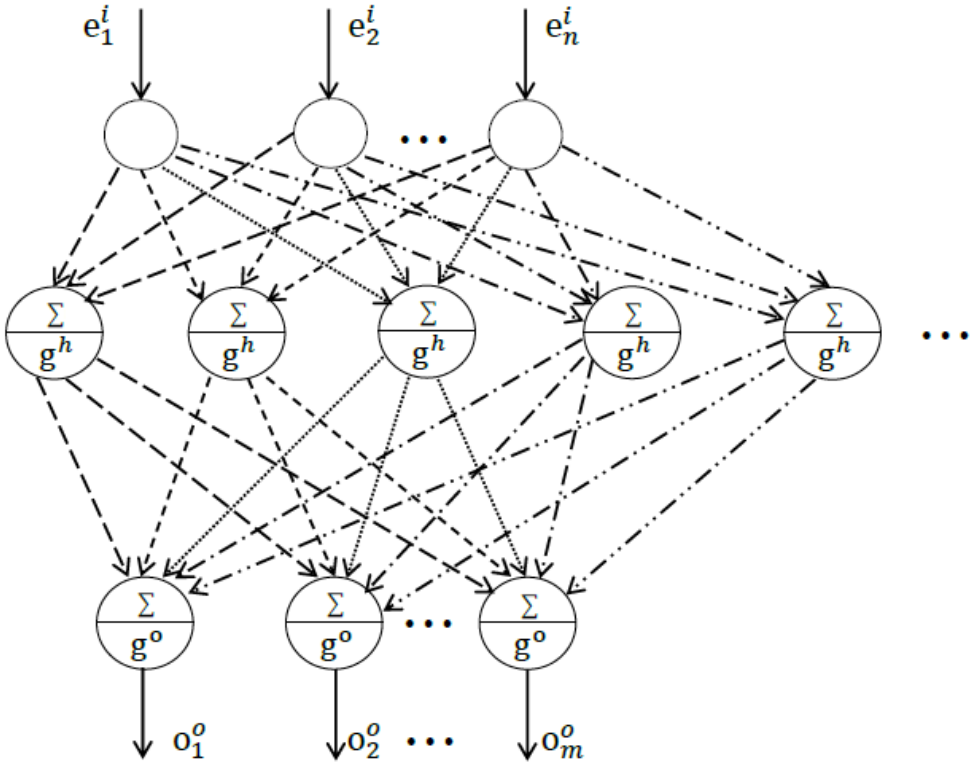


Abbildung 2.6 Multilayer-Perceptron mit einer verdeckten Schicht

## Lernverfahren

Die Optimierung der Parameter, oder sogenannte Training, wird von Lernverfahren durchgeführt. Hier wird eine allgemeine Klassifizierung der Lernverfahren angegeben, nämlich Überwachtes Lernen, Reinforcement Lernen sowie Nichtüberwachtes Lernen. Im Praktikumsversuch wird nur das sogenannte „überwachte Lernen“ eingesetzt.

Bei überwachtem Lernen werden an die Eingänge des Netzes zunächst Eingangswerte angelegt. Die vom Netz errechneten Ausgangswerte werden mit den realen Ausgangswerten der zu modellierenden Funktion verglichen. Aus der Abweichung zwischen den vom Netz berechneten und den realen Ausgangswerten werden mit Hilfe eines Fehlerminimierungsverfahrens die Parameter des Netzes adaptiert. Dies wird mit unterschiedlichen Ein-/ Ausgangsdaten so lange wiederholt, bis das Neuronale Netz die gewünschte Funktion mit ausreichender Genauigkeit approximiert oder bis man zu dem Schluss gelangt, dass die zu approximierende Funktion mit dem gewählten Netz oder unter den gewählten Randbedingungen nicht befriedigend approximiert werden kann.

## Training des Perceptrons

Hier wird ein überwachtes Lernverfahren betrachtet. Demnach existiert ein Trainingsdatensatz, in dem Ein-/Ausgangsdaten abgespeichert sind. Zu Beginn des Trainings werden die Verbindungsgewichte $w_{kj}$ und die Schwellwerte $\vartheta_i$ mit Zufallswerten belegt. Ein Eingangsvektor $\underline{x}$ wird aus den Trainingsdaten ausgewählt und am Eingang des Netzes

angelegt. Anhand der zufällig initialisierten Parameter berechnet das Netz dazu eine Ausgangsgröße $o_k^o$ . Diese wird mit der gewünschten Ausgangsgröße $y_k$ aus dem Lerndatensatz verglichen und der Fehler wird folgenderweise berechnet:

$$\delta_k = y_k - o_k^o \quad .$$

Wenn $\delta_{k0}$ gleich Null ist, bleiben die zum Ausgangsneuron $k_0$ führenden Gewichte $w_{k0j}$ und der Schwellwert $\vartheta_{k0}$ unverändert. Ansonsten werden diese zwei Parameter gemäß

$$\Delta w_{kj} = \eta \delta_k e_j^o = \eta \delta_k e_j^i$$

$$\Delta \vartheta_k = \eta \delta_k$$

adaptiert. Dabei ist $\Delta w_{kj}$ die Änderung des Verbindungsgewichts zwischen Neuron j der Eingangsschicht und Neuron k der Ausgangsschicht. $\eta$ Ist eine Konstante, die als Lernrate bezeichnet wird. $e_j$ Ist der j-te Eingangswert der Ausgangsschicht und kann damit dem Eingang des j-ten Eingangsneurons gleichgesetzt werden, wenn die Neuronen der Eingangsschicht leiglich als Verteilungspunkte angesehen werden. Dieses Verfahren wird iterativ so lange wiederholt, bis für alle Eingangsvektoren des Datensatzes eine korrekte Klassifizierung erfolgt, oder bis eine maximale Zahl von Iterationen überschritten wird.

## Backpropagation-Algorithmus

Der Algorithmus ist als Lernalgorithmus bekannt, mit dem die Gewichte $w$ zu den verdeckten Schichten im Multilayer-Perceptron optimiert werden konnten. Beim Backpropagation-Algorithmus wird ein quadratisches Gütemaß $J$ mit Hilfe eines Gradientenabstiegsverfahrens minimiert, das die Summe der quadratischen Fehler aller Ausgangsneuronen über alle Trainingsdatensätze ist. Bei $n^o$ Ausgangsneuronen und $N$ Datensätzen ist das quadratische Gütemaß also durch

$$J = \frac{1}{2} \sum_{k=1}^{n^o} \sum_{m=1}^{N} \left( y_{m,k} - \hat{y_{m,k}} \right)^2$$

gegeben. Dabei ist $y_{m,k}$ der k-te Ausgangswert im m-ten Datensatz, $\hat{y_{m,k}}$ ist der vom Neuronalen Netz dafür berechnete Approximationswert. Im Folgenden wird dieser Algorithmus vereinfacht, indem das Gütemaß in die Anteile aufgespaltet wird. Man bezeichnet den Beitrag des k-ten Ausgangsneurons im m-ten Datensatz mit

$$J_{m,k} = \frac{1}{2} \left( y_{m,k} - \hat{y}_{m,k} \right)^2 \quad ,$$

so ergibt sich der Anteil eines Datensatzes am Gütemaß zu

$$J_m = \sum_{k=1}^{n^o} J_{m,k}$$

und das gesamte Gütemaß als

$$J = \sum_{m=1}^{N} J_m \quad .$$

Nach jedem Schritt des Lernverfahrens sollen die Verbindungsgewichte des MLP so verändert werden, dass das Gütemaß möglichst stark verringert wird. Die Optimierung kann mittels eines Gradientenabstiegsverfahrens durchgeführt werden, bei dem ein Gewicht proportional zum Gradienten des Gütemaßes bezüglich dieses Gewichts geändert wird.

Die Regel für die Adaption des Gewichts zwischen dem $j_1$ -ten Neuron der verborgenen Schicht und dem $k_1$ -ten Ausgangsneuron ist

$$\Delta w_{k_1, j_1} = \eta \sum_{m=1}^{N} (y_{m,k_1} - o^o_{m,k_1}) \frac{\partial g^o}{\partial a^o_{m,k_1}} o^h_{m,j_1} \quad .$$

Die Regel für die Adaption des Gewichts zwischen dem $l_1$ -ten Neuron der Eingangsschicht und dem $j_1$ -ten Neuron der verborgenen Schicht ist

$$\Delta w_{j_1, l_1} = \eta \sum_{m=1}^{N} \left( e_{l_{1,m}} \sum_{k=1}^{n^o} (y_{m,k} - o^o_{m,k}) \frac{\partial g^o}{\partial a^o_{m,k}} w_{k j_1} \frac{\partial g^h}{\partial a^h_{m,j_1}} \right) \quad ,$$

dabei bezeichnet $\eta$ die Lernrate, mit der die Konvergenzgeschwindigkeit des Lernverfahrens eingestellt werden kann.

### 2.1.4 Neuronale Netze in der Regelungstechnik

Das Neuronale Netz kann nicht nur zur Approximation von Funktionen verwendet werden, sondern auch zur Regelung angewandt werden. Mit Hilfe eines Neuronalen Netzes lässt sich eine Strecke, deren mathematischen Modell für Rechnersimulationen oder für einen Reglerentwurf am Rechner nicht durch analytische Modellbildung gewonnen werden kann, approximieren. Das bedeutet, dass ein Neuronales Netz den mathematische Zusammenhang zwischen Ein- und Ausgängen nachbilden kann. Um ein solches Neuronales Netz zur Approximation dieses Zusammenhangs zu trainieren, sind Lerndaten erforderlich.

Dabei ist es wichtig zu beachten, dass das Neuronale Netz nur in den Bereichen, in denen Lerndaten verfügbar waren, eine korrekte Approximation errechnen kann. Die Identifikation eines Streckenmodels ist schematisch in der Abbildung 2.7 dargestellt.

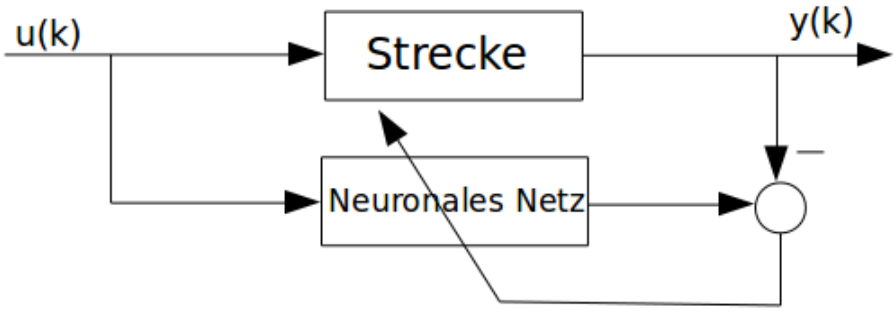


Abbildung 2.7 Approximation der Strecke mit einem Neuronalen Netz

Neuronale Netze können auch als Regler eingesetzt werden. Dabei wird ebenfalls ein ma-

thematischer Zusammenhang zwischen Ein- und Ausgangssignalen approximiert. Außerdem kann ein Neuronales Netz ein inverses Modell der Strecke approximieren.

## 2.2 Fuzzy-Control

### 2.2.1 Einführung

Ziel einer Fuzzy-Regelung ist es, die menschliche Vorgehensweise bei der Regelung technischer Anlagen nachzuahmen. Dadurch wird es möglich, empirisch gewonnenes Prozesswissen und linguistisch formulierte Steuerstrategien zur Regelung des Systems einzusetzen. Ein mathematisches Modell der Strecke ist dazu nicht erforderlich.

### 2.2.2 Struktur einer Fuzzy-Regelung

In der Abbildung 2.8 ist die Struktur einer Fuzzy-Regelung dargestellt.

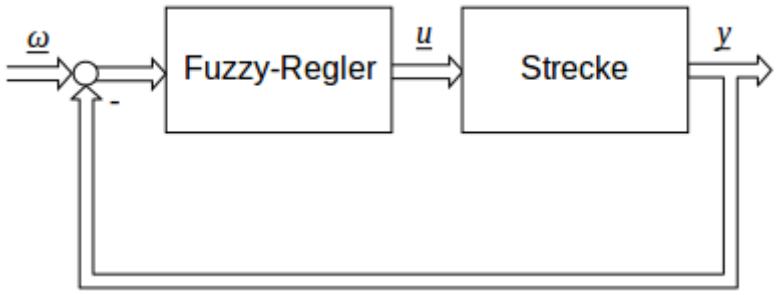


Abbildung 2.8 Struktur einer Fuzzy-Regelung

Die Sollwert $\underline{w}$ , die Stellgrößen $\underline{u}$ und die Ausgangsgrößen $\underline{y}$ sind - wie bei der Klassischen Regelung auch - scharfe Prozessgrößen, im Fuzzy-Regler werden jedoch unscharfe Größen verarbeitet. Daher müssen die scharfen Eingangsgrößen des Fuzzy-Regler zunächst in unscharfe Größen umgewandelt werden, dieser Schritt wird als Fuzzifizierung bezeichnet. Anschließend wird aus den unscharfen Eingangswerten in einem Inferenzverfahren auf unscharfe Ausgangswerte geschlossen. Dazu werden im Wissenserwerb gewonnene Wenn-Dann-Regeln verarbeitet. Schließlich muss auch den unscharfen Ausgangswerten wieder eine scharfe Stellgröße gewonnen werden, dieser Vorgang wird als Defuzzi-fizierung bezeichnet.

Beispiel: In einem Flughafen wird eine technisch veraltete Personenbahn modernisiert. Die Beförderungskapazität soll deutlich erhöht werden. Das Anfahr und Bremsverhalten der Zugführer soll von Fuzzy-System nachgeahmt werden. Für die Steuerung werden die Zuggeschwindigkeit v, die Geschwindigkeitsdifferenz Δv und der Abstand Δs zum vorderen Zug gemessen.

### 2.2.3 Fuzzifizierung

### Scharfe Mengen

Eine scharfe Menge A ist festgelegt durch einen Grundbereich X, der die betrachteten Elemente x enthält, und eine binäre Zugehörigkeitsfunktion $u_A$ , die jedem Elemente $x_0$ des Grundbereichs entwerder den Zugehörigkeitsgrad 0 oder den Zugehörigkeitsgrad 1

zur Menge zuordnet:

$$u_A : X \rightarrow \{0,1\}$$

$$A = \{\, x \in X \mid u_A(x) = 1 \,\}$$

Der Zugehörigkeitsgrad des Elements x zur Menge A ist also genau dan gleich 1, wenn x Element der Menge A ist. Die Zugehörigkeitsfunktion einer scharfen Menge ist in der Abbildung 2.9 dargestellt.

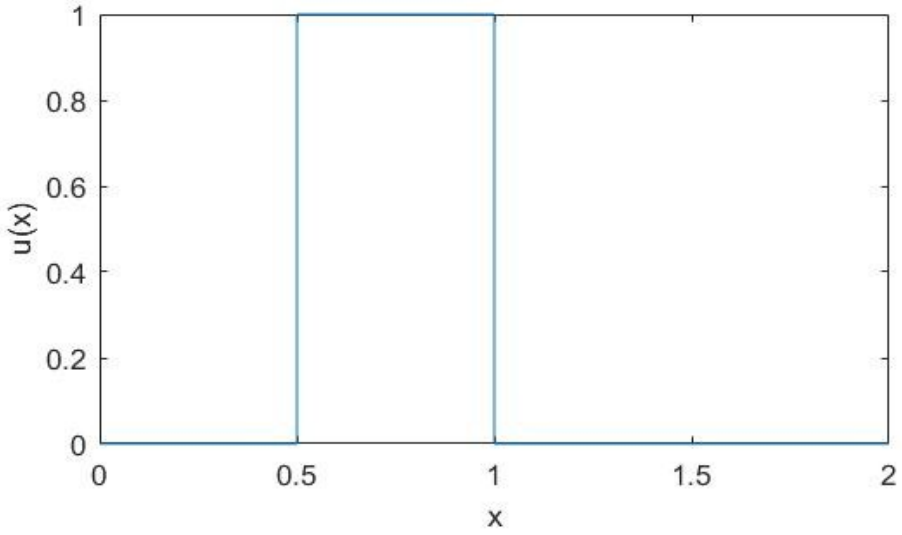


Abbildung 2.9 Zugehörigkeitsfunktion einer scharfen Menge

## Unscharfe Mengen und Zugehörigkeitsfunktion

Bei unscharfen Menge kann eine scharfe Grenze angegeben werden, ab welcher der Wert x eindeutig zur Menge A gehört. Der Übergang ist fließend und die Zugehörigkeitsfunktion $u_A(x)$ kann alle Werte im Intervall $[0,1]$ annehmen. In der Abbildung 2.10 sind die gebräuchlichsten Funktionen dargestellt.

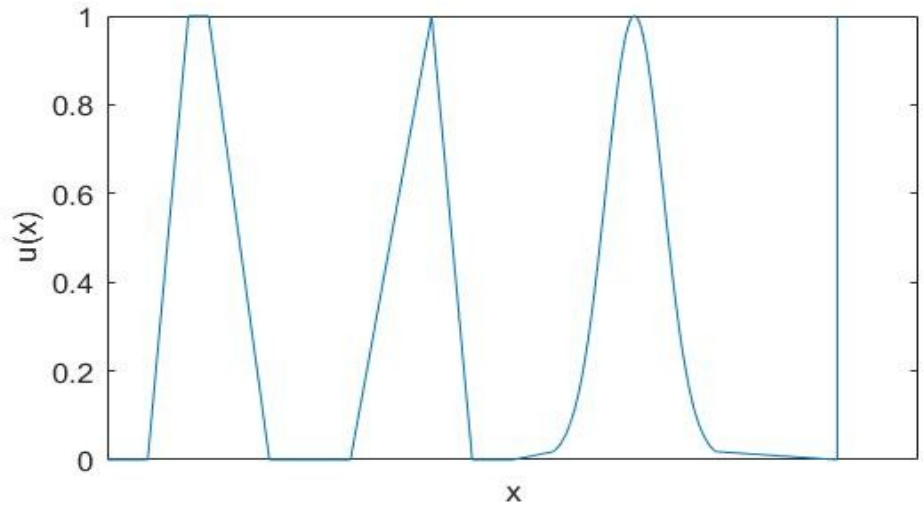


Abbildung 2.10 Einfache Zugehörigkeitsfunktionen unscharfer Menge

Diese Zugehörigkeitsfunktionen sind durch wenige Parameter eindeutig bestimmt und daher mathematisch einfach handhabbar.

## Linguistische Variable und linguistischer Wert

Die unscharfen Begriffe der menschlichen Sprache werden als *linguistische Werte* bezeichnet. Eine linguistische Variable ist dann eine Variable, die nicht Zahlenwerte, sondern lin-guistische Werte annehmen kann. Beispielweise ist „Geschwindigkeit“ eine linguistische Variable, die die linguistischen Werte

$$A_{11}=langsam\,,A_{12}=mittel\,,A_{13}=schnell\,,A_{14}=sehr\ schnell$$

annehmen kann.

Beschreibt man diese linguistischen Werte durch unscharfe Mengen, so kann man sie wieder durch ihre Zugehörigkeitsfunktionen darstellen. Für die unscharfen Mengen in der Abbildung 2.11 liest man dann beispielweise ab, dass die Geschwindigkeit $v=45\,km/h$ mit einem Zugehörigkeitsgrad von 0.2 zur unscharfen Menge „zu schnell“ gehört, außerdem auch mit einem Zugehörigkeitsgrad von 0.6 zur „schnell“. Dieses Übersetzen eines scharfen Wertes in Zugehörigkeitsgrade zu unscharfen Mengen wird als Fuzzifizierung bezeichnet.

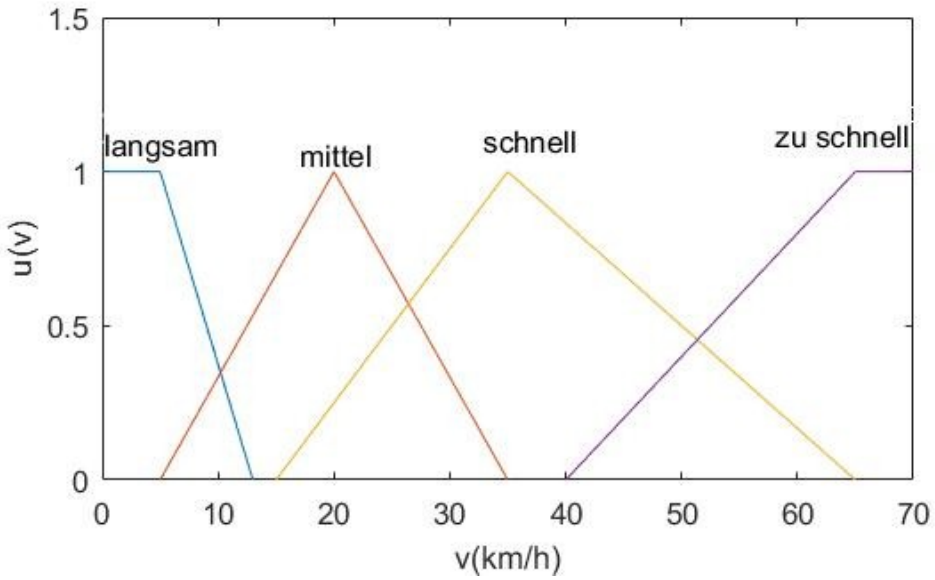


Abbildung 2.11 Linguistische Werte zur „Geschwindigkeit“

## 2.2.4 Wissendarstellung mit linguistischen Regeln

Das Wissen, das über die Regelstrategie bekannt ist, muss in einer bestimmten Form dargestellt werden. Beim Peoplemover können etwa Regeln der Form

„ Wenn der Zug zu schnell fährt, dann stark bremsen“

angegeben werden. Ist $X_1$ die linguistische Variable „Geschwindigkeit“ und $Y$ die linguistische Variable „Beschleunigung“, so könnte die Regel auch als

WENN $X_1=A_{14}$ DANN $Y=B_1$

formuliert werden. Eine solche Regel, die linguistische Aussagen miteinander verknüpft, wird als linguistische Regel bezeichnet. Die Prämisse beschreibt die Eingangssituation der Inferenz, die resultierende Schlussfolgerung ist eine Handlungsanweisung.

### 2.2.5 Wissenverarbeitung, Inferenz

Die Auswertung der in der Regelbasis zusammengefassten linguistischen Regeln zu einer linguistischen Schlussfolgerung wird als Inferenz bezeichnet. Um Schlussfolgerungen angeben zu können, müssen auch für die Ausgangsgrößen linguistische Variablen definiert werden. Im Beispiel des Peoplemovers ist die Ausgangsgröße die Beschleunigung, ihre linguistischen Werte sind in der Abbildung 2.12 dargestellt.

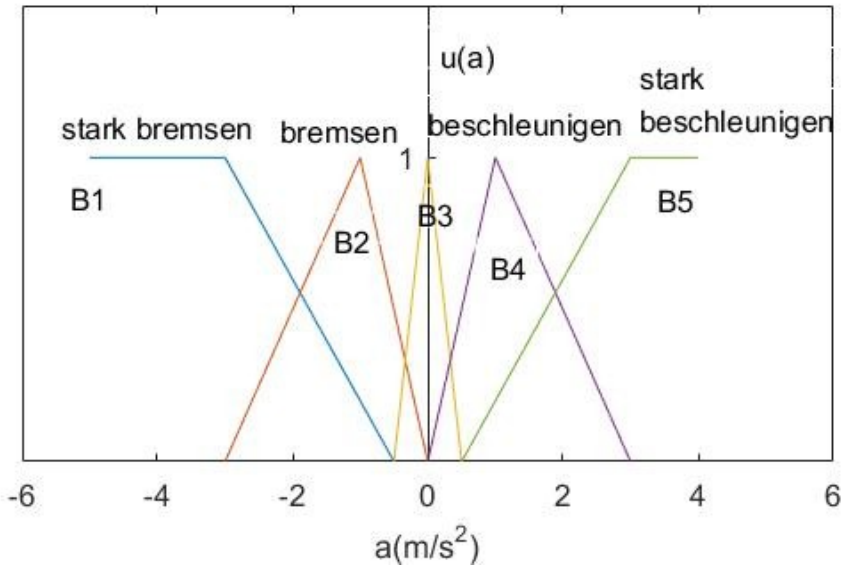


Abbildung 2.12 Linguistische Werte der Ausgangsgröße Beschleunigung

Die Inferenz besteht nun aus den folgenden Schritten. Zunächst wird der Wahrheitswert der Prämisse bestimmt. Anschließend wird damit auf die Aktivierung der Konklusion geschlossen, und schließlich werden alle aktivierten Regeln zu einer Ausgangsgröße zusammengefasst.

Prämissen sollen aus mehre Eingangsgrößen gebildet werden. Im Beispiel wurde bisher lediglich eine Eingangsgröße betrachtete. Diese reicht aber nicht aus. Der Abstand $\Delta s$ zum vorderen Zug wird als eine weitere linguistische Eingangsgröße mit den linguistischen Werten aus der Abbildung 2.13 eingefügt.

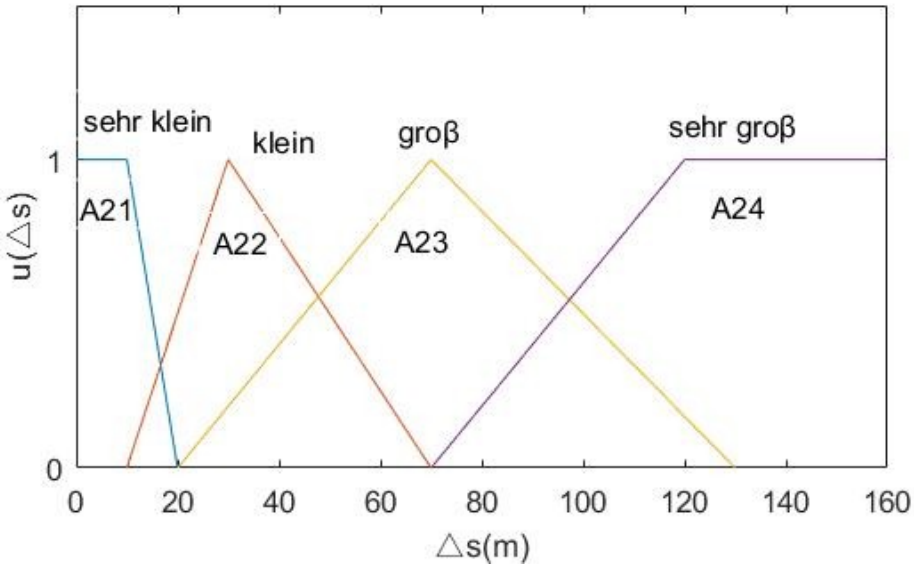


Abbildung 2.13 Linguistische Werte der Abstand zum vorderen Zug

Da jetzt zwei Eingangsgröße vorhanden sind, können in den Prämissen der Regeln auch Aussagen durch UND oder ODER miteinander verknüpft werden,

„ Wenn zu schnell und Δs sehr klein, dann stark bremsen“

beziehungsweise

WENN $X_1 = A_{14}$ UND $X_2 = A_{21}$ DANN $Y = B_1$ .

Die Gesamtheit aller aufgestellten Regeln stellt die Regelbasis dar. Damit beschreibt die Regelbasis das zur Regelung eines Prozesses vorhandene Wissen.

Bei zwei Eingangs- und einer Ausgangsgröße kann die Regelbasis anschaulich in Form einer Matrix dargestellt werden. Für den Peoplemover könnte die Regelbasis etwa wie folgt gegeben sein:

| | A21 | A22 | A23 | A24 |
|---|---|---|---|---|
| A11 | B3 | B4 | B5 | B5 |
| A12 | B2 | B3 | B4 | B5 |
| A13 | B2 | B2 | B3 | B3 |
| A14 | B1 | B1 | B2 | B2 |

Tabelle 2.1 Regelbasis für den Peoplemover

## Prämissenauswertung

In der Prämissenauswertung werden die Wahrheitswert der Prämissen aller Regeln berechnet. Dazu müssen die in den Prämissen verwendeten logischen Verknüpfung mathematisch gefasst werden.

## ODER-Verknüpfung

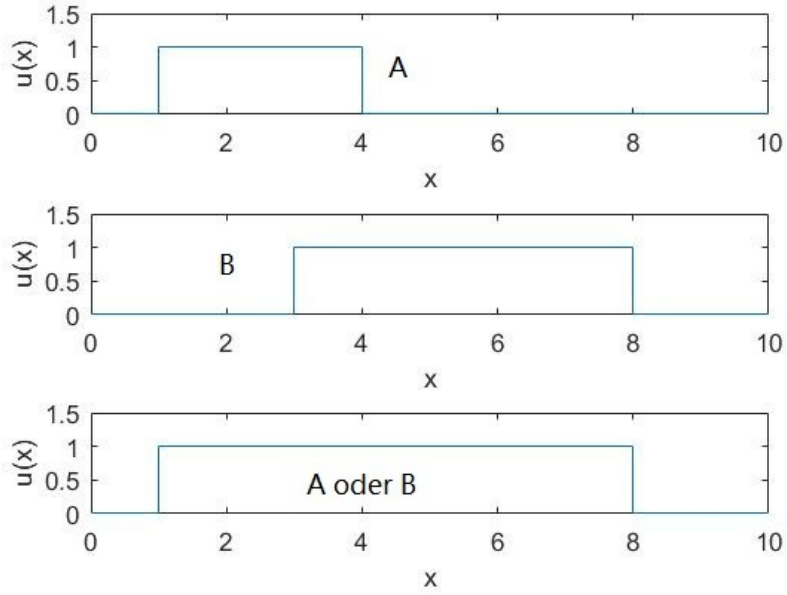


Abbildung 2.14 ODER-Verknüpfung von scharfen Mengen

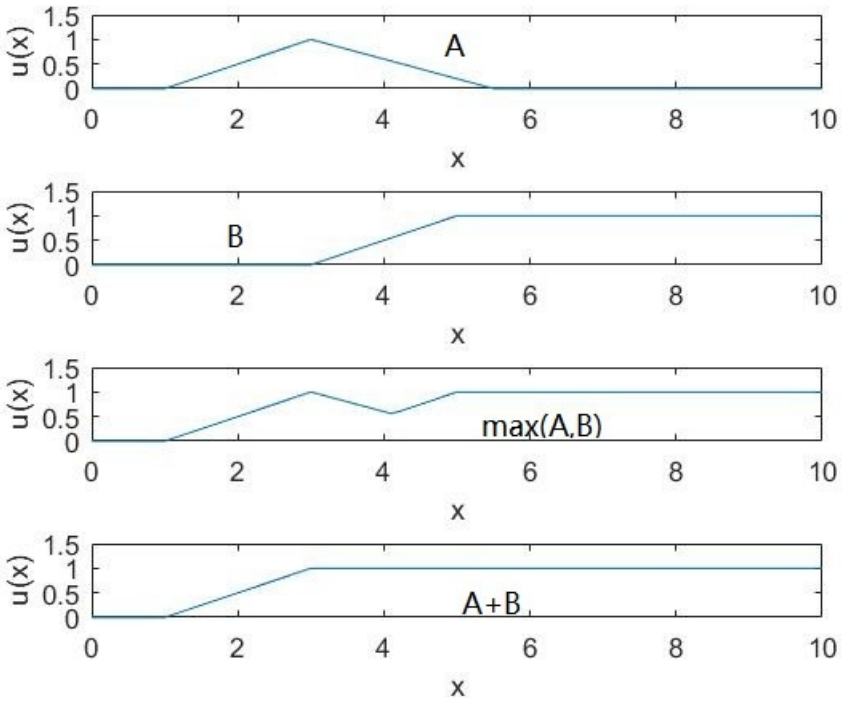


Abbildung 2.15 ODER-Verknüpfung von unscharfen Mengen

Die ODER-Verknüpfung scharfen Mengen ist in der Abbildung 2.14 dargestellt. Da bei der ODER-Verknüpfung nicht beide Bedingungen gleichzeitig erfüllt sein müssen, sind die in der Abbildung 2.15 gezeichnete Maximumbildung und die Addition geeignete ODER-Operator für unscharfe Mengen. In der Theorie unscharfer Mengen können allgemein beliebige Operatoren als ODER-Verknüpfung eingesetzt werden, sofern sie die Eigenschaften einer T-Conorm erfüllen. Diese Eigenschaften sind in [1] zusammengestellt.

Die Maximumbildung ist ein sehr häufig eingesetztes ODER. Aus [1] ist ersichtlich, dass die Addition keine T-Conorm ist. Sie wird wegen ihrer einfachen Handhabbarkeit aber trotzdem häufig als ODER-Operator eingesetzt.

## UND-Verknüpfung

Die UND-Verknüpfung scharfer bzw. unscharfer Mengen ist in den Abbildung 2.16 und 2.17 dargestellt. Bei unscharfen Mengen können die Minimumbildung und das Produkt als UND-Operator eingesetzt werden. Allgemeine UND-Operator werden über die sogenannte T-Norm charakterisiert. Die Eigenschaften einer T-Norm sind in [1] abgegeben. Daraus ist ersichtlich, dass das Produkt keine T-Norm ist. Trotzdem verwendet man es oft als UND-Verknüpfung unscharfer Mengen.

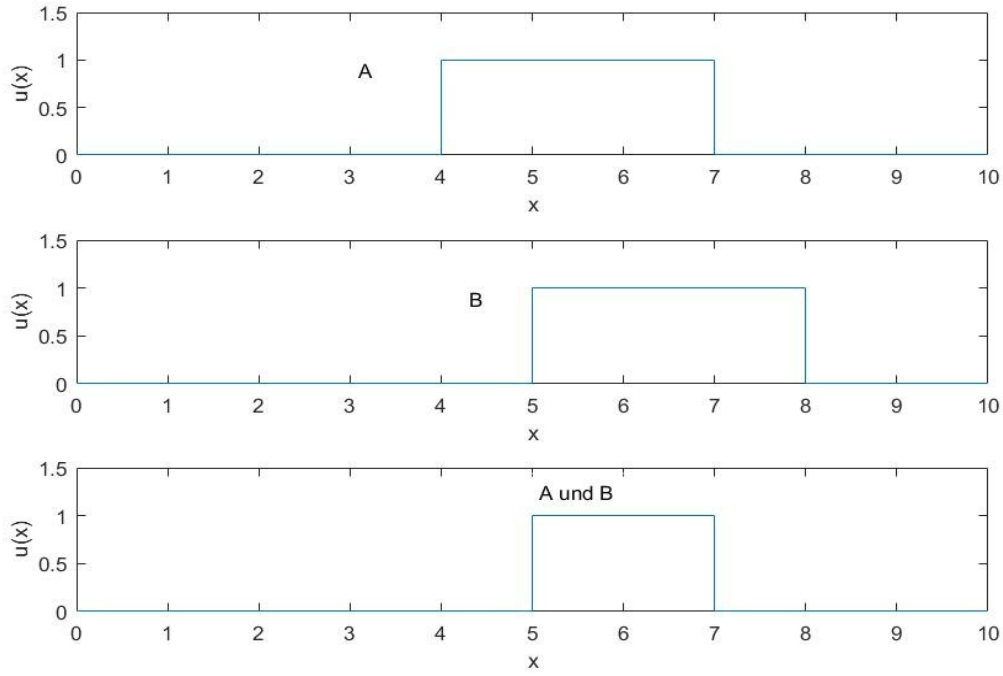


Abbildung 2.16 UND-Verknüpfung scharfer Mengen

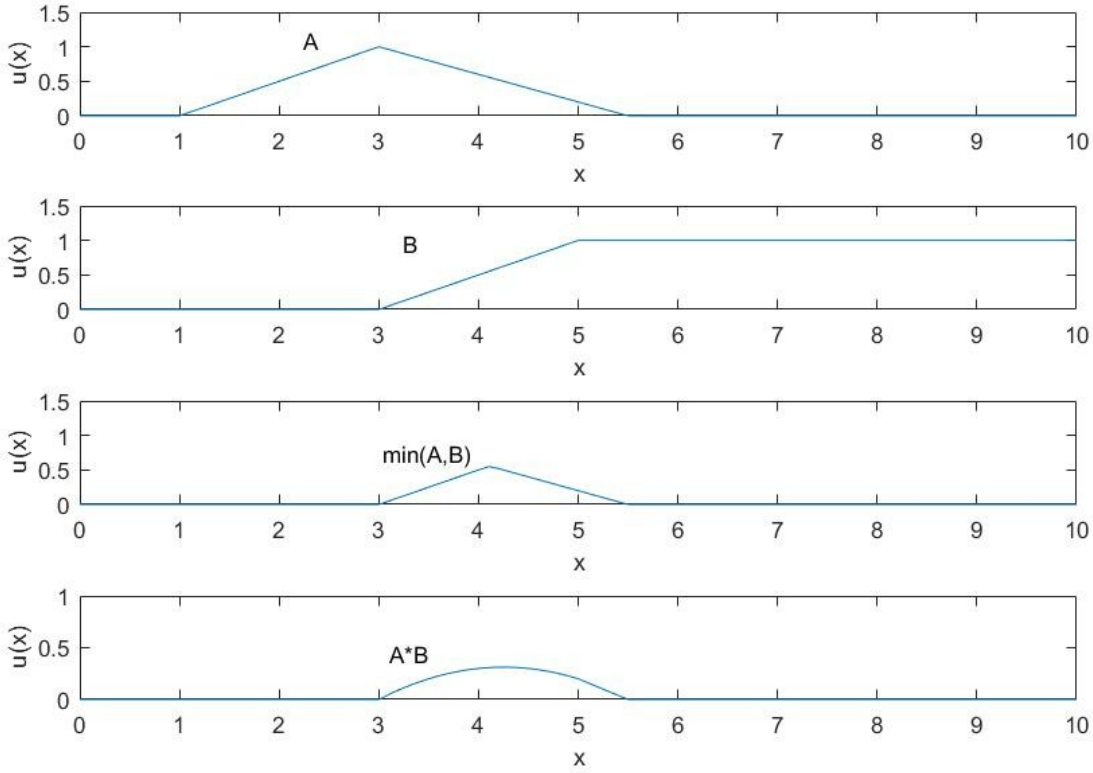


Abbildung 2.17 UND-Verknüpfung unscharfer Mengen

## Zylindrische Erweiterung

Formuliert man die Regeln der Regelbasis, so sind die Elemente der Prämissen meistens auf verschiedenen Grundbereichen definiert. Beispielweise wurde für den Peoplemover die Regel

$$WENN\ X = A_{14}\ UND\ Y = A_{21}\ DANN\ Z = B_1$$

angegeben. Mathematisch betrachtet ist die Auswertung der UND-Verknüpfung zunächst nicht möglich, da die beiden unscharfen Mengen auf verschiedenen Grundbereichen definiert sind. Eine Lösung dafür stellt die zylindrische Erweiterung dar. Dabei werden die Grundbereiche der Zugehörigkeitsfunktionen so erweitert, dass sie miteinander übereinstimmen. Als Beispiel werden zwei unscharfe Menge u und v betrachtet, die auf unterschiedliche Grundbereichen definiert sind. Ihre beiden Zugehörigkeitsfunktionen sind in der Abbildung 2.18 dargestellt.

Um die UND-Verknüpfung dieser beiden unscharfen Mengen berechnen zu können, wird zunächst die Zylindrische Erweiterung durchgeführt. Dabei wird die Zugehörigkeitsfunktion $u_u(x)$ in Richtung des Grundbereichs y der anderen unscharfen Menge erweitert und entsprechend wird die zweite Zugehörigkeitsfunktion in Richtung von x erweitert. Die Ergebnisse sind in der Abbildung 2.19 dargestellt.

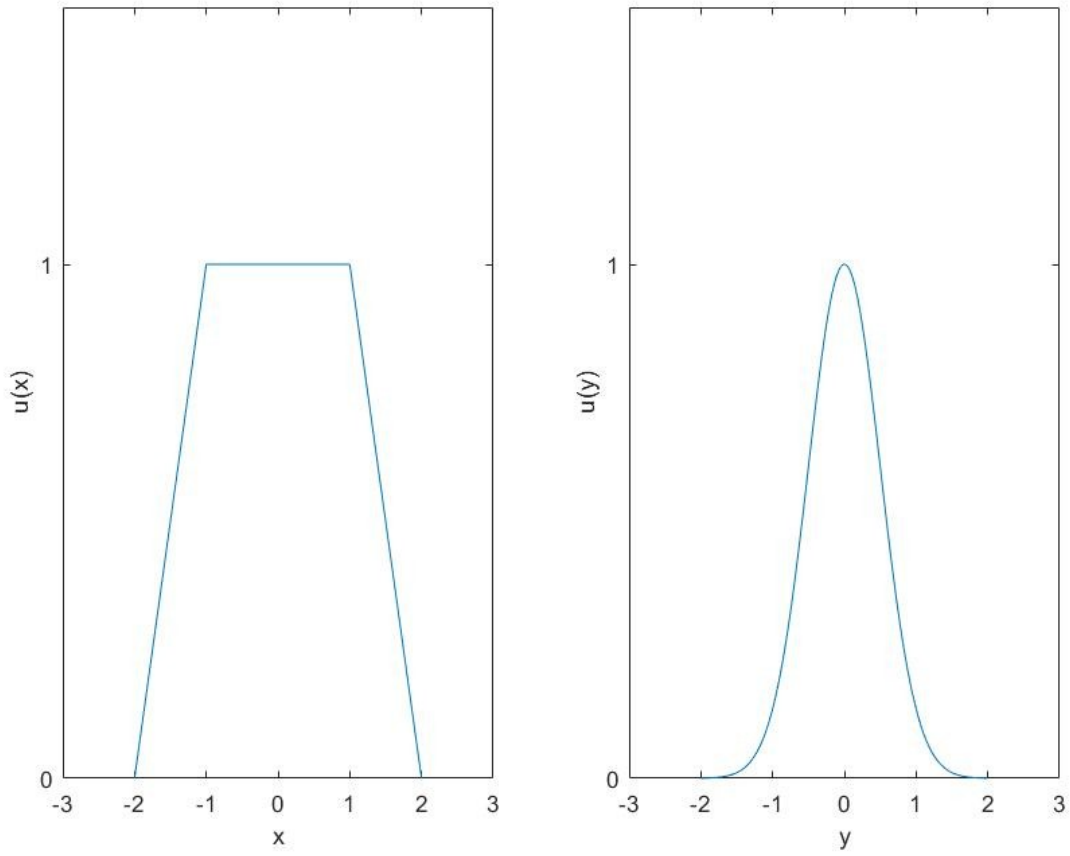


Abbildung 2.18 Zugehörigkeitsfunktionen der Mengen u und v

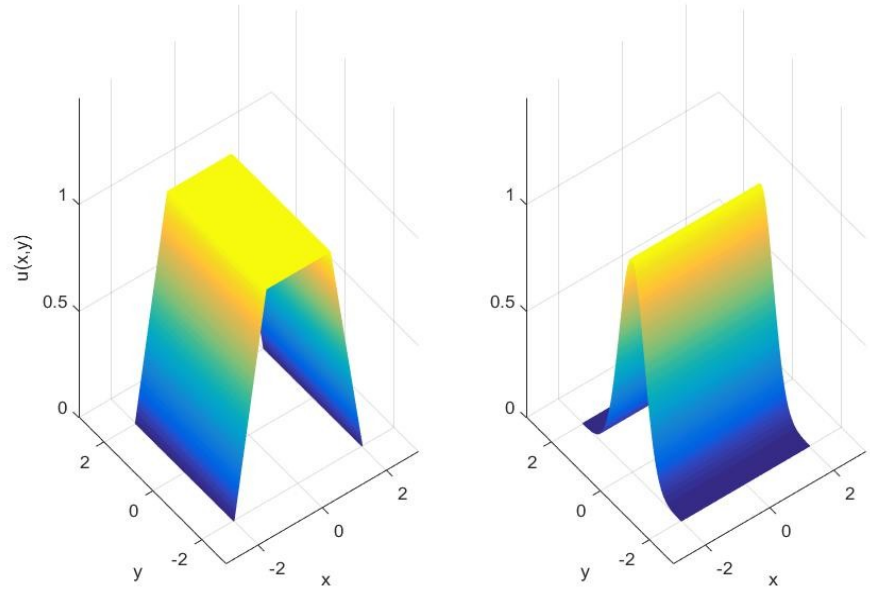


Abbildung 2.19 Zylindrische Erweiterung der Mengen aus Abbildung 2.18

Da die beiden Mengen jetzt auf dem gleichen Grundbereich definiert sind, können sowohl die UND- als auch die ODER-Operation ausgeführt werden. Für die Maximum und die Mi-

nimumbildung sind die Ergebnisse in der Abbildung 2.20 gezeigt.

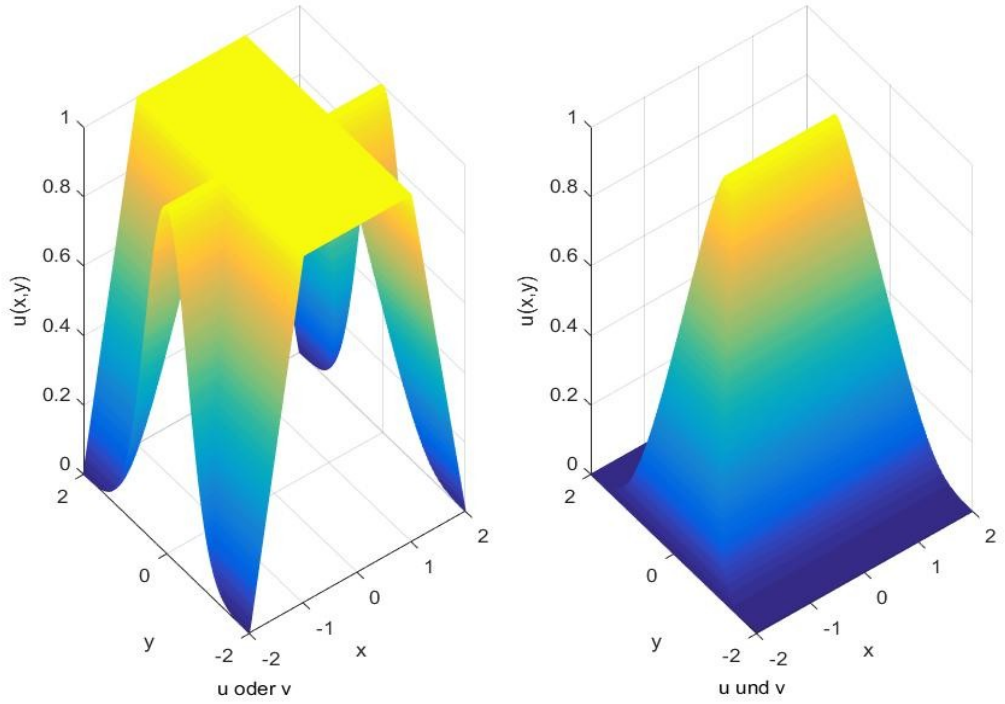


Abbildung 2.20 UND, ODER-Verknüpfung der unscharfen Mengen aus Abbildung 2.19

Der Wahrheitswert oder Erfüllungsgrad der Prämisse für konkrete Eingangsgrößen ist

$$w_p = u_{uv}(x_0, y_0)$$

Zum gleichen Ergebnis kann man auch kommen, indem man die beiden Prämissen getrennt auswertet und anschließend UND-Verknüpft.

## Aktivierung

Nachdem in der Prämissenauswertung die Wahrheitswerte der Bedingungen der Wenn-Dann-Regeln berechnet wurden, müssen die Zugehörigkeitsfunktionen der Konklusionen der Regeln ausgewertet werden. Dies geschieht in der sogenannten Aktivierung. Üblicherweise enthält die Konklusion einer Regel nur eine Handlungsanweisung, also keine logischen Verknüpfungen. Im Folgenden wird die anschauliche „Methode der Aktivierungsgrade“ beschrieben. Bei ihr wird zur Ermittlung der Aktivierung einer Schlussfolgerung die UND-Verknüpfung zwischen Wahrheitswerte $w_p$ der Prämisse und Zugehörigkeitsfunktion der Konklusion gebildet.

Verwendet man als T-Norm bei der Aktivierung den Minimumoperator, so bedeutet das anschaulich, dass die Zugehörigkeitsfunktion der Konklusion auf der Höhe $w_p$ abgeschnitten wird. Für die obige Regel mit dem Erfüllungsgrad 0.2 ist die aktivierte Konklusion in der Abbildung 2.21 gezeichnet.

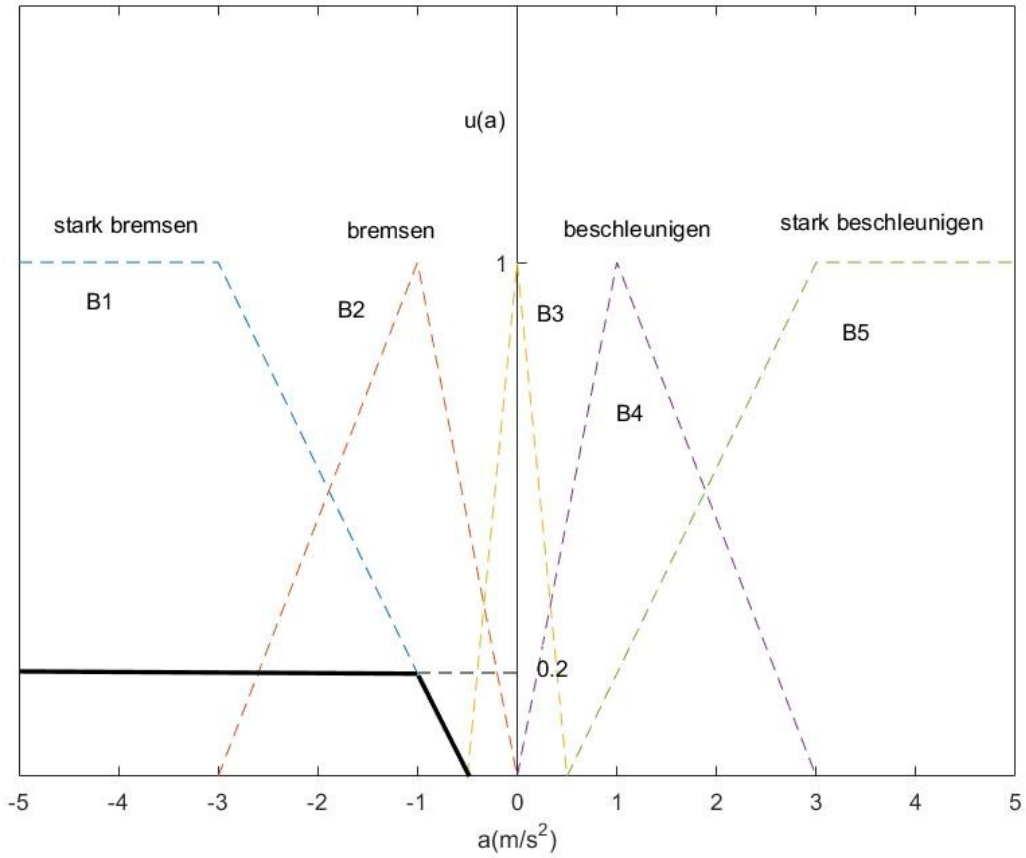


Abbildung 2.21 Aktivierung einer Konklusion mit dem Minimumoperator

## Akkumulation

Aufgabe der Akkumulation ist es, die aktivierten Konklusionen aller Regeln zu einer resultierenden unscharfen Menge am Ausgang zusammenzufügen. Dazu wird eine ODER-Verknüpfung eingesetzt. Als Beispiel werden die folgenden beiden Regeln betrachtet:

R1: „ Wenn v zu schnell UND Δs sehr klein, DANN stark bremsen“

R2: „ Wenn v schnell UND Δs sehr klein, DANN bremsen“

Bei gegebenen Eingangsgrößen sei die erste Regel zu 0.4 erfüllt, die zweite Regel zu 0.6. Mit der Minimum-Aktivierung erhält man die beiden „abgeschnitten“ Konklusionen in der Abbildung 2.22 als aktivierte Konklusionen. Wird aus diesen beiden unscharfen Mengen mit der Maximumbildung eine ODER-Verknüpfung gebildet, so erhält man als unscharfes Resultat die in der Abbildung 2.22 durch die durchgezogenen Linien umgefasst un-scharfe Menge, das sogenannte Attraktivitätsgebirge.

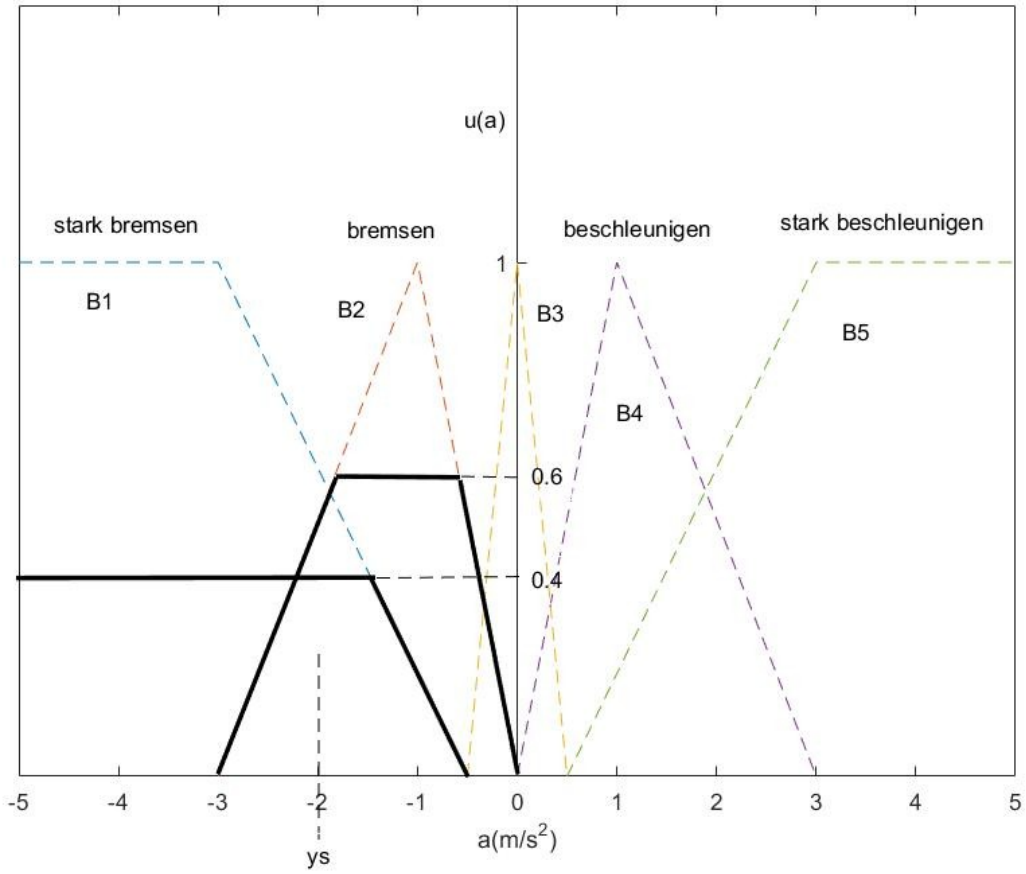


Abbildung 2.22 Aktivierung und Akkumulation zweier Konklusionen

## 2.2.6 Defuzzifizierung

Die Inferenz liefert die Zugehörigkeitsfunktion $u_Y(y)$ der linguistischen Ausgangsvariabl Y. Aus dieser unscharfen Menge der linguistischen Variable muss nun ein scharfer Ausgangswert berechnet werden, da der reale Regler nur scharfe Werte ausgeben kann. Dieser Vorgang wird Defuzzifizierung gennant.

**Maximum-Mittelwert-Methode**

Der scharfe Ausgangswert wird als Mittelwert aller Maxima des Attraktivitätsgebirges der Augangsgröße berechnet.

**Schwerpunktmethode**

Bei diesem Verfahren wird der Schwerpunkt der Fläche unter dem Attraktivitätsgebirge als scharfer Ausgangswert des Reglers verwendet. Ist die Zugehörigkeitsfunktion des Attraktivitätsgebirges durch $u_Y(y)$ gegeben, so berechnet sich der Schwerpunkt $y_s$ aus

$$y_s = \frac{\int y * u_Y(y)\,dy}{\int u_Y(y)\,dy}$$

Für die Zugehörigkeitsfunktionen aus dem Beispiel ergibt sich der in der Abbildung 2.22 bereits eingezeichnete scharfe Ausgangswert $y_s$ .

### 2.2.7 Fuzzy-Regler als Kennfeld

Durch die Auswertung der Regelbasis mit der Methode der Aktivierungsgrade entsteht eine eindeutige Abbildung von Eingangsgrößen auf Ausgangsgrößen. Daher ist es nicht notwendig, bei jeder Berechnung des Ausgangswert den ganzen Rechenweg über Fuzzierung, Aktivierung, Pramissenauswertung, Akkumulation und Defuzzifizierung auszuführen.

In der Praxis wird das durch den Regler definierte nichtlineare Kennfeld häufig durch nichtlineare Interpolationsfunktionen, die schneller ausgewertet werden können, approximiert.

Dazu müssen lediglich die Stützstellen der Interpolation, aber nicht die Fuzzy-Regeln abgespeichert werden.

# 3.Aufgaben

## Aufgabe 1: Perceptron

Skizzieren Sie die Struktur eines künstlichen Neuronalen Netzes, mit dem die Funktion

$$a \,\text{UND}\, (b \leftrightarrow c)$$

nachgebildet werden kann ( $b \leftrightarrow c$ bedeutet b äquivalent c). Stellen Sie dazu zunächst eine Wahrheitstabelle auf.

**Antwort:**

Da die Zahl der Ein- bzw. Ausgangsneuronen jeweils gleich der Zahl der Ein- bzw. Ausgangsgänge ist, gelten hier $n^i = 3$ , $n^o = 1$ .

Die Ausgangsgröße ist als *o* (für *output*) bezeichnet. Zur Vereinfachung wird ein Zwischenwert *d* eingeführt. $d = b \leftrightarrow c = (b \,\text{UND}\, c) \;\text{ODER}\; [\text{NICHT}(b) \,\text{UND}\; \text{NICHT}(c)]$ . Dann bekommen wir:

$$o = a \,\text{UND}\, (b \leftrightarrow c) = a \,\text{UND}\, d \quad .$$

Man kann einfach die Wahrheitstabelle für *d* aufstellen:

| b | c | d |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Tabelle 3.1 Wahrheitstabelle für *d*

Die logische Funktion für *o* ist deshalb durch die folgende Wahrheitstabelle gegeben:

| a | b | c | o |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Tabelle 3.2 Wahrheitstabelle für *o*

Es ist offensichtlich, dass die Funktion mit acht möglichen Eingangstupel nicht linear separabel ist und deshalb von keinem Perceptron direkt nachgebildet werden kann.

Die Struktur der Funktion kann wie so restrukturiert:

$$o = a \,\text{UND}\, (b \leftrightarrow c) = a \,\text{UND}\, d = a \,\text{UND}\, \{(b \,\text{UND}\, c) \text{ ODER } [\text{NICHT}(b) \text{UND } \text{NICHT}(c)]\}$$

$$= (a \text{ UND } b \text{ UND } c) \text{ ODER } [a \text{ UND } \text{NICHT } (b) \text{ UND } \text{NICHT } (c)]$$

Das erste Perceptron kann durch die Funktion beschrieben:

$$y_1 = a \text{ UND } b \text{ UND } c$$

Das zweite durch $y_2 = a \text{ UND } \text{NICHT } (b) \text{ UND } \text{NICHT } (c)$ beschrieben.

Die folgende Struktur entspricht der Formulierung der Funktion für *o* durch lineare separable Funktionen, nämlich

$$o = a \,\text{UND}\, (b \leftrightarrow c) = y_1 \text{ ODER } y_2$$

Hier dienen die Eingangsneuronen der drei Netze jeweils nur als Verteilungspunkte für die entsprechenden Eingangsgrößen, können die Eingangsneuronen von 1. und 2. Perceptron zusammengelegt werden, die Eingangsneuronen von 3. Perceptron können weggelassen werden. Diese führen zu den zweischichtigen Perceptronen, wie es in der Abbildung 3.1 dargestellt ist.

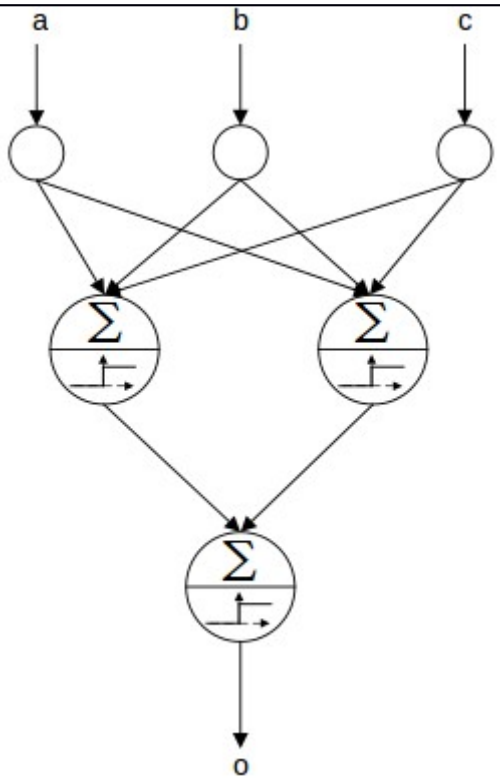


Abbildung 3.1 Struktur des Neuronalen Netzes

## Aufgabe 2: Backpropagation

Bei der Herleitung des Backpropagation-Algorithmus werde die Regel zur Adaption der Gewichtsfaktoren $w_{xy}^{z}$ angegeben, nicht aber für die Adaption der Schwellwerte $\vartheta_i^z$ . Geben Sie eine Möglichkeit an, wie diese ohne großen zusätzlichen Aufwand ebenfalls adaptiert werden können.

**Antwort:**

Ist das Gütemaß durch $J$ gegeben, so wird der Schwellwert $\vartheta$ der Ausgangsschicht oder verborgenen Schicht gemäß

$$\Delta\vartheta = -\eta \frac{\partial J_m}{\partial \vartheta}$$

verändert. Dabei bezeichnet $\eta$ die Lernrate. Verwendet man Gleichung $J_m = \sum_{k=1}^{n^o} J_{m,k}$ zur Darstellung des Gütemaßes, so wird daraus

$$\Delta\vartheta = -\eta \frac{\partial}{\partial \vartheta} \sum_{k=1}^{n^o} J_{m,k} = -\eta \sum_{k=1}^{n^o} \frac{\partial J_{m,k}}{\partial \vartheta} = -\eta \sum_{k=1}^{n^o} \frac{\partial}{\partial \vartheta} \left(\frac{1}{2}\left(y_{m,k} - \hat{y}_{m,k}\right)^2\right) \quad .$$

Der vom Neuronalen Netz berechnete Approximationswert $\hat{y}_{m,k}$ ist gerade der Ausgangswert $o_k^o$ des k-ten Neurons der Ausgangsschicht. Damit ergibt sich also

$$\Delta\vartheta = -\eta \sum_{k=1}^{n^o} \frac{\partial}{\partial \vartheta} \left(\frac{1}{2}\left(y_{m,k} - o_k^o\right)^2\right) = \eta \sum_{k=1}^{n^o} \left(y_{m,k} - o_k^o\right) \frac{\partial o_k^o}{\partial \vartheta_k} \quad .$$

Der Ausgang des Neurons berechnet sich über die Aktivierungsfunktion

$$o_k^o = g^o(a_k^o) \quad .$$

Durch Anwendung der Kettenregel erhält man also

$$\Delta\vartheta = \eta \sum_{k=1}^{n^o} (y_{m,k} - o_k^o) \frac{\partial o_k^o}{\partial a_k^o} \frac{\partial a_k^o}{\partial \vartheta} = \eta \sum_{k=1}^{n^o} (y_{m,k} - o_k^o) \frac{\partial g^o}{\partial a_k^o} \frac{\partial a_k^o}{\partial \vartheta} \quad .$$

Da die Aktivierung $a_k^o$ durch

$$a_k^o = \sum_{j=1}^{n^h} \omega_{kj} o_j^h + \vartheta_k$$

gegeben ist und nur $\vartheta_{k_1}$ berücksichtigt werden muss, erhält man

$$\Delta\vartheta = \eta \sum_{k=1}^{n^o} (y_{m,k} - o_k^o) \frac{\partial g^o}{\partial a_k^o} \cdot 1$$

als die Änderung des Schwellwerts im m-ten Datensatz. Jetzt wird die Summation wieder berücksichtigt und man erhält schließlich die Regel für die Adaption des Schwellwerts vom $k_1$ -ten Neuron der Ausgangsschicht als

$$\Delta\vartheta = \eta \sum_{m=1}^{N} \sum_{k=1}^{n^o} (y_{m,k} - o_{m,k}^o) \frac{\partial g^o}{\partial a_{m,k}^o} \quad .$$

## Aufgabe 3: Identifikation mit einem Neuronalen Netz

1. Welche Parameter stehen bei der Identifikation einer Regelstrecke mit einem Neuronalen Netz zur Verfügung, um die Approximationsfähigkeit des Netzes zu beeinflussen?

**Antwort:**

- Anzahl der Perceptronen und Schichten (Netztyp)
- die Struktur des Neuronalen Netz
- Typ von der Aktivierungsfunktion
- die Gewichtungsfaktoren $\omega_{kj}$ und die Schwellwerte $\vartheta$

2. Wie gestaltet sich die Vorgehensweise bei der Identifikation mit einem Neuronalen Netz? Entwerfen Sie ein einfaches Ablaufdiagramm. Machen Sie sich insbesondere auch Gedanken über die Validation. Vergleichen Sie hierzu den Fehler, den Sie bei Anlegen der Lerndaten an den Eingang erhalten, mit dem, den Sie bei Anlegen von Eingangswerten erhalten, die in den Lerndaten nicht enthalten sind.

**Antwort:**

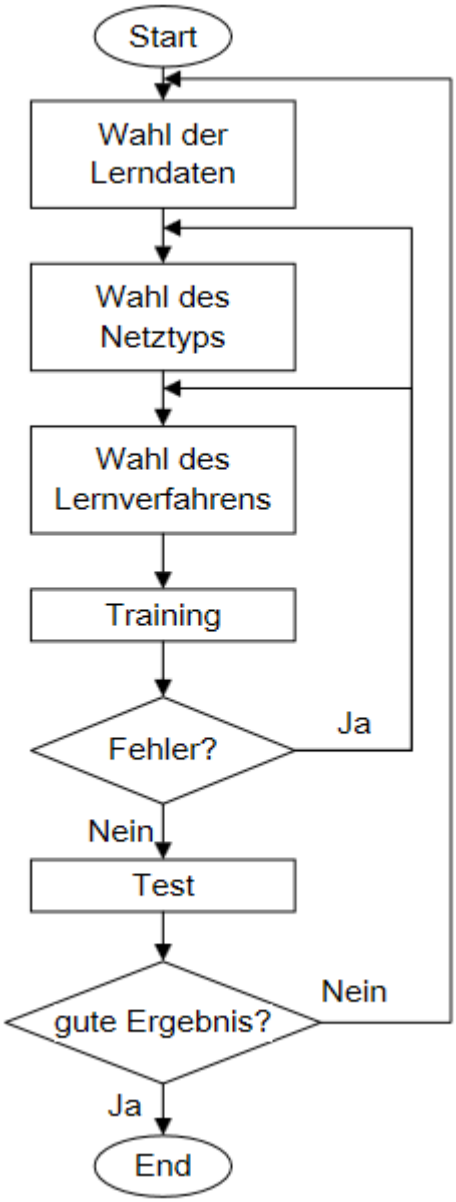


Abbildung 3.2 Ablaufdiagramm der Identifikation

## Aufgabe 4: Neuronale Netze als Regler

Das Dreitanksystem kann mit ein wenig Übung gut durch einen Menschen geregelt werden. Welche Ein- und Ausgangsgrößen würden Sie bei einem Neuronalen Netz berücksichtigen, das den Menschen als Regler des 3-Tank-Systems nachbilden soll? Welchen Netztyp würden Sie wählen? Wie viele Eingangs- und Ausgangsneuronen würden Sie verwenden?

**Antwort:**

Bei dem Neuronalen Netz werden die 2 Füllhöhenabweichungen vom Tank 1 und Tank 2 ( nämlich $\Delta h_1$ und $\Delta h_2$ ) als Eingangsgrößen betrachtet, während die Volumenströme der 2 Pumpen ( nämlich $q_1$ und $q_2$ ) als Ausgangsgrößen betrachtet werden. Als Netztyp wird das Multiplayer-Perceptron gewählt. Da die Zahl der Ein- bzw. Ausgangsneuronen jeweils gleich der Zahl der Ein- bzw. Ausgangsgänge ist, gelten hier $n^i$ =2, $n^o$ =2. Des-

halb werden 2 Eingangsneuronen und 2 Ausgangsneuronen verwendet.

## Aufgabe 5: Fuzzy-Logik

Ermitteln Sie grafisch die Stellgröße beim Peoplemover, wenn die aktuelle Geschwindigkeit $v = 35\,km/h$ ist, der Abstand zum vorderen Zug $\Delta s = 50\,m$ . Verwenden Sie dazu die Regelbasis und die Zugehörigkeitsfunktionen.

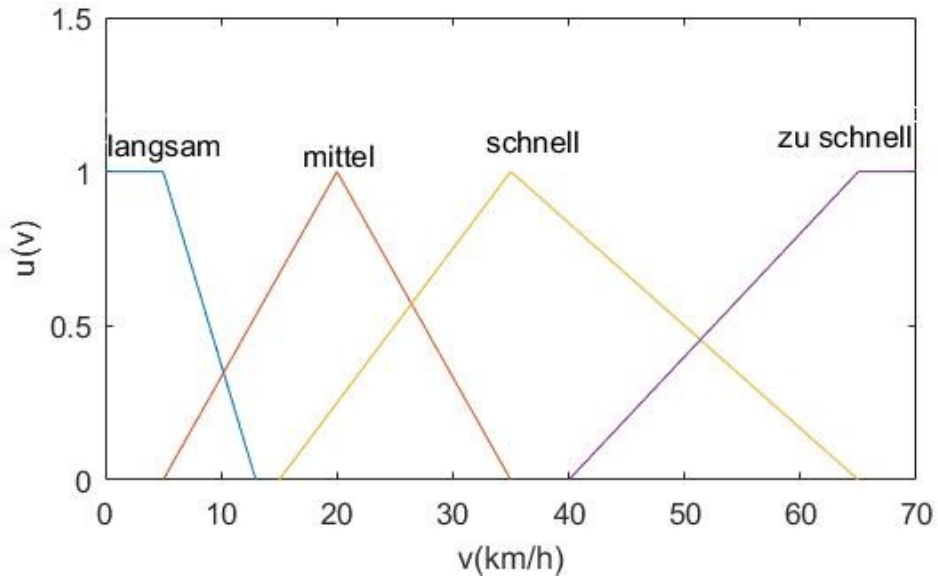


Abbildung 3.3 Linguistische Werte zur linguistischen Variable „Geschwindigkeit“

**Antwort:**

Mit $v = 35\,km/h$ kann man aus dem Bild lesen, dass die Geschwindigkeit einen Zugehörigkeitsgrad 1 von Schnell besitzt.

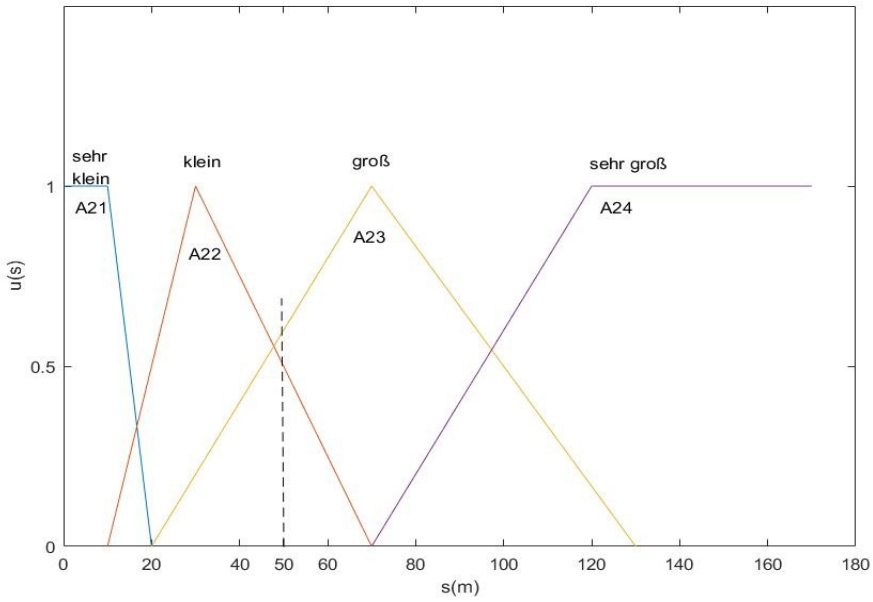


Abbildung 3.4 Linguistische Werte der Eingangsgröße „Abstand zum vorderen Zug“

Aus dem Bild kann man lesen, dass der Abstand einen Zugehörigkeitsgrad 0.6 von groß

und einen Zugehörigkeitsgrad 0.5 von klein besitzt. Deswegen haben wir jetzt 2 Prämissen:

1. Schnell und groß mit(1*0.6)

2. Schnell und klein mit(1*0.5)

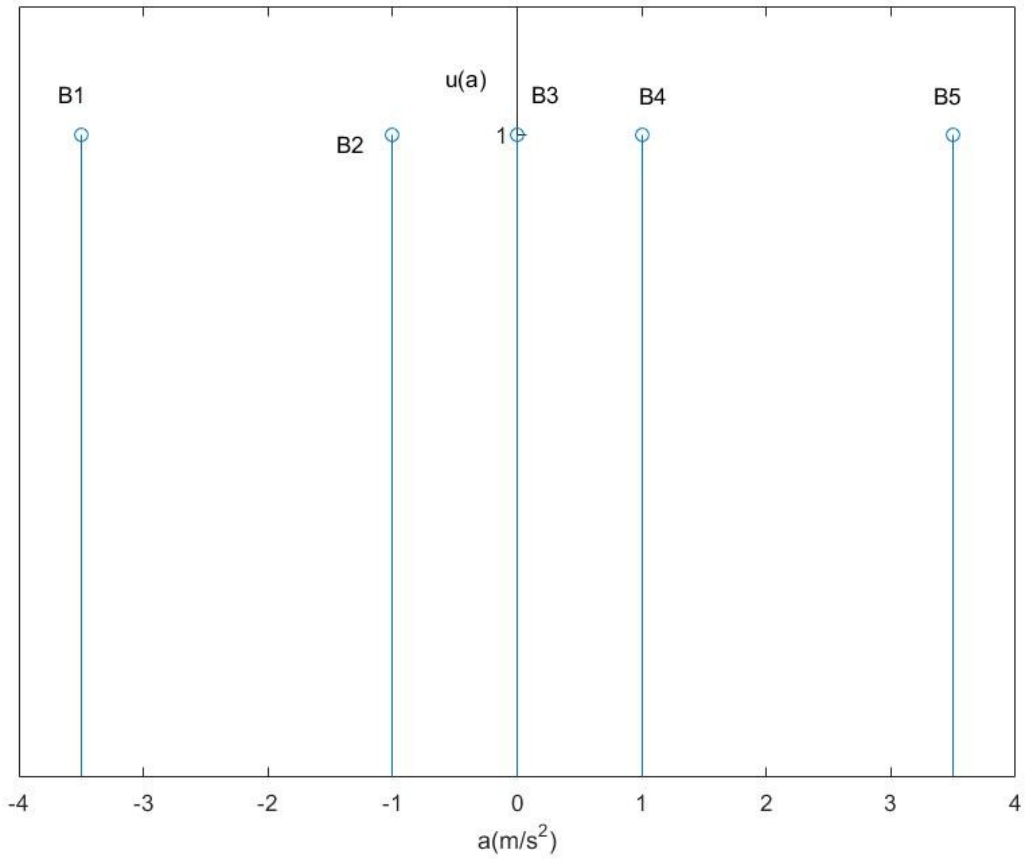


Abbildung 3.5 Zugehörigkeitsfunktion zum Regelgesatz des Peoplemoves

1. Schnell und groß entspricht B3

2. Schnell und klein entspricht B2

Dann machen wir die Akkumulation. Bekommen wir die Zugehörigkeitsfunktion

$$u(a)=0.6\delta(a)+0.5\delta(a+1) \quad .$$

Dann machen wir die Defuzzifizierung mit der Gleichung

$$a_s=\frac{\int a*u_Y(a)\,da}{\int u_Y(a)\,da} \quad .$$

Bekommen wir $a_s=\frac{-0.5}{1.1}$ .

## Aufgabe 6: Aufbau einer Fuzzy-Regelung

1. Welche Parameter stehen beim Entwurf eines Fuzzy-Reglers zur Verfügung, um die Dy-

namik des Reglers zu beeinflussen?

**Antwort:**

- Regelbasis
- Linguistische Werte
- Anzahl der Eingangs- und Ausgangsgröße
- Zugehörigkeitsfunktionen

2. Wie gestaltet sich die Vorgehensweise beim Entwurf eines Fuzzy-Reglers? Entwerfen Sie ein einfaches Ablaufdiagramm. Machen Sie sich auch Gedanken über die Validation.

**Antwort:**

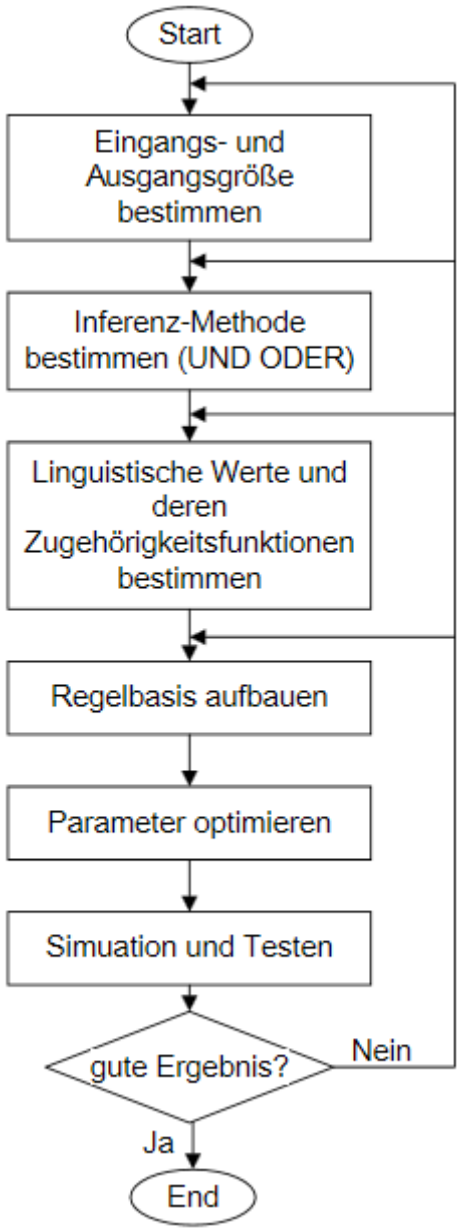


Abbildung 3.6 Ablaufdiagramm des Entwurfs eines Fuzzy-Reglers

## Aufgabe 7: Eigenschaften Neuronaler Netze

Bevor Sie ein neuronales Netz aufbauen und trainieren bzw. einen Neuro-Regler für das 3-Tank-System entwerfen, sollten Sie sich einige Eigenschaften Neuronaler Netze klar machen. Das MATLAB-Programm /NeuroDemo1/Lernrate.m veranschaulicht die Minimierung des Gütemaßes und den Einfluss der Lernrate beim Backpropagation-Algorithmus. Das Programm /NeuroDemo2/Approx.m demonstriert den Einfluss der Zahl der Neuronen auf die Approximationsfähigkeit eines Neuronalen Netzes.

**Antwort:**

Vor allem wird der Einfluss der Lernrate beim Backpropagation-Algorithmus untersucht.

Die Optimierung im Backpropagation-Algorithmus wird mittels eines Gradientenabstiegsverfahrens durchgeführt. Die Lernrate beschreibt prinzipiell die Konvergenzgeschwindigkeit des Lernverfahrens. Ist die Lernrate geeignet gewählt, wird das Minimum schnell gefunden. In Abbildung 3.7 und 3.8 erzielt das Verfahren das Ziel nur nach ungefähr 52 Epochen.

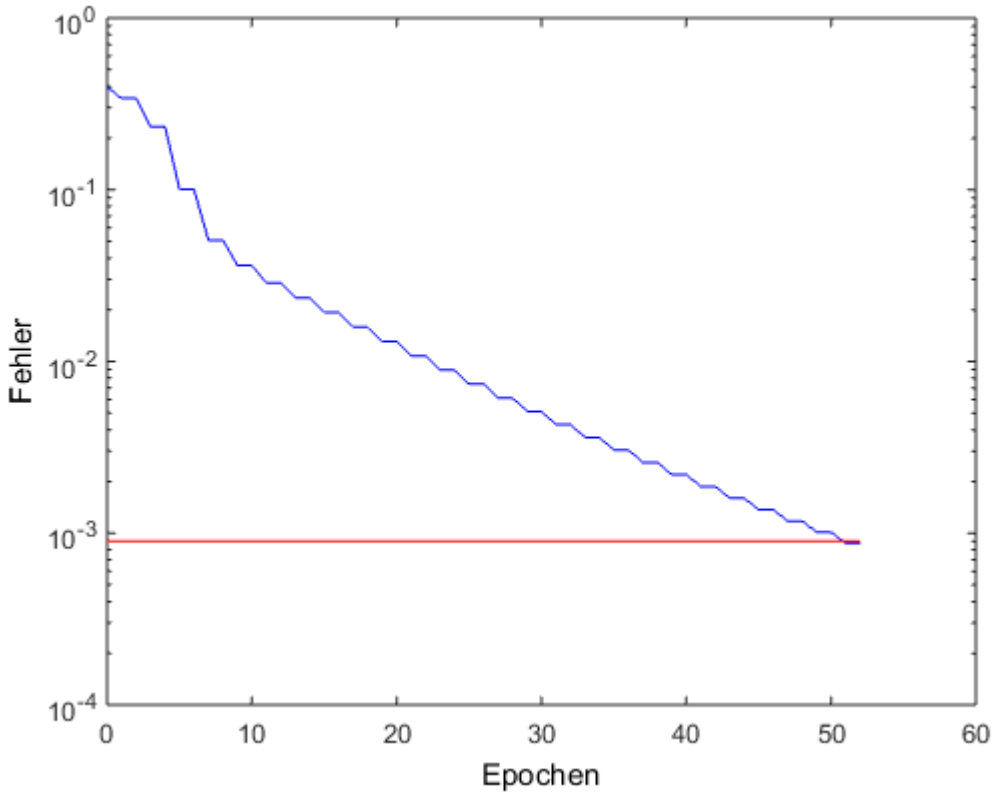


Abbildung 3.7 Fehler des Neuronalen Netzes mit geeigneter Lernrate

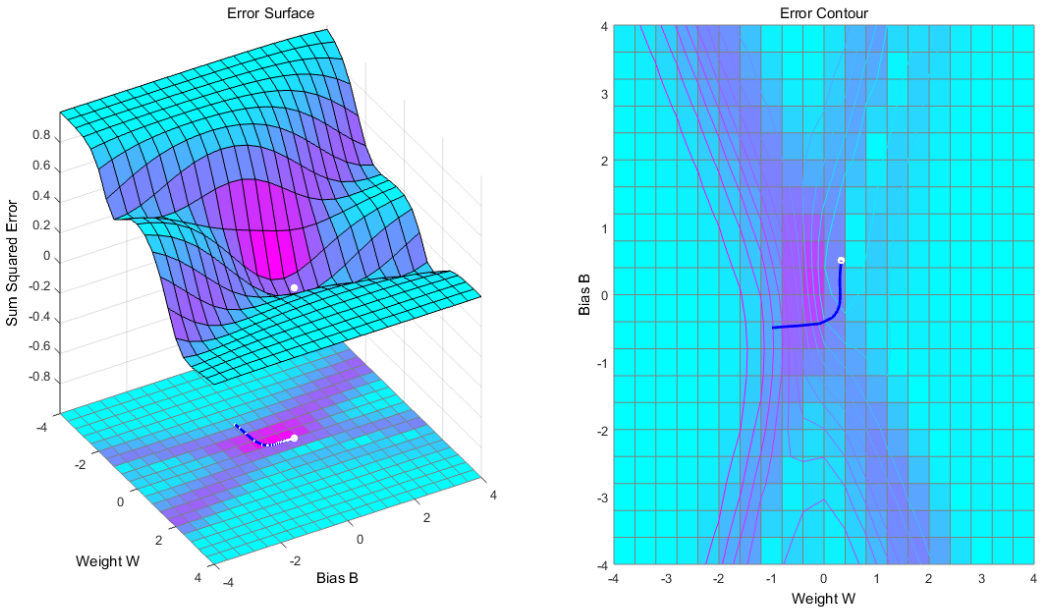


Abbildung 3.8 Weight des Neuronalen Netzes mit geeigneter Lernrate

Wenn die Lernrate verhältnismäßig groß ist, treten die laufenden Schwingungen auf. Die Ergebnisse sind in Abbildung 3.9 und 3.10 gezeigt. Nach 100 Epochen gibt es noch anhaltende sich verändernde Fehler, dabei erreicht das Ergebnis das globale Minimum nicht, weil der Wert des Gewichts das Minimum jedesmal überquert.

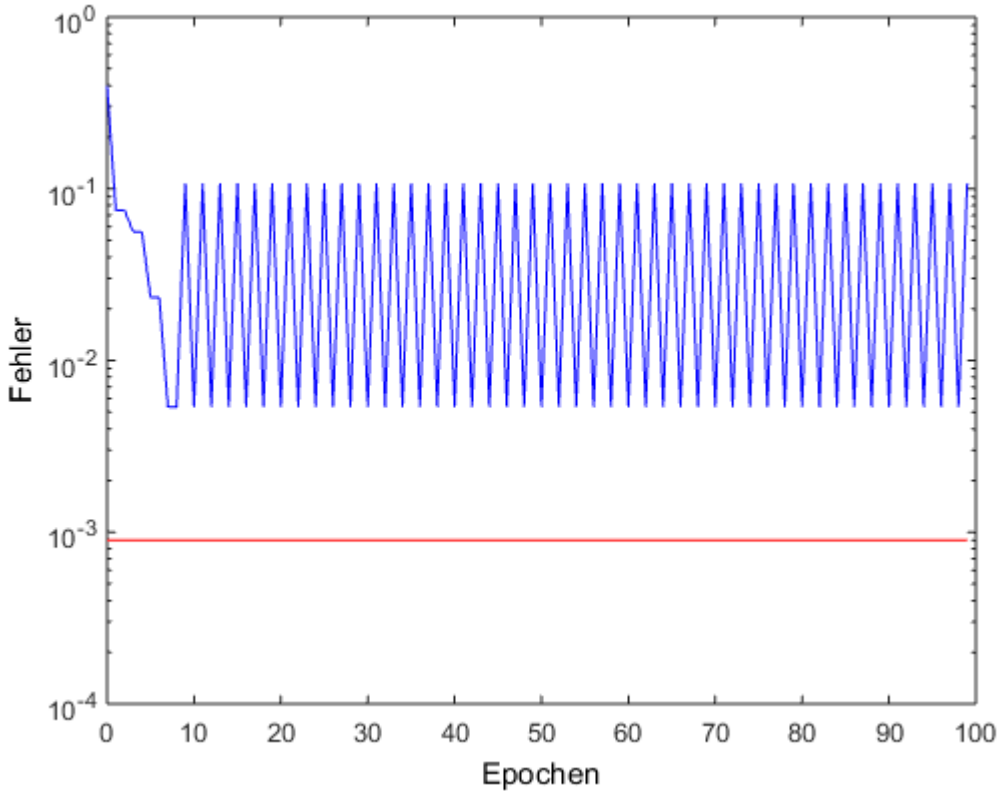


Abbildung 3.9 Fehler des Neuronalen Netzes mit großer Lernrate

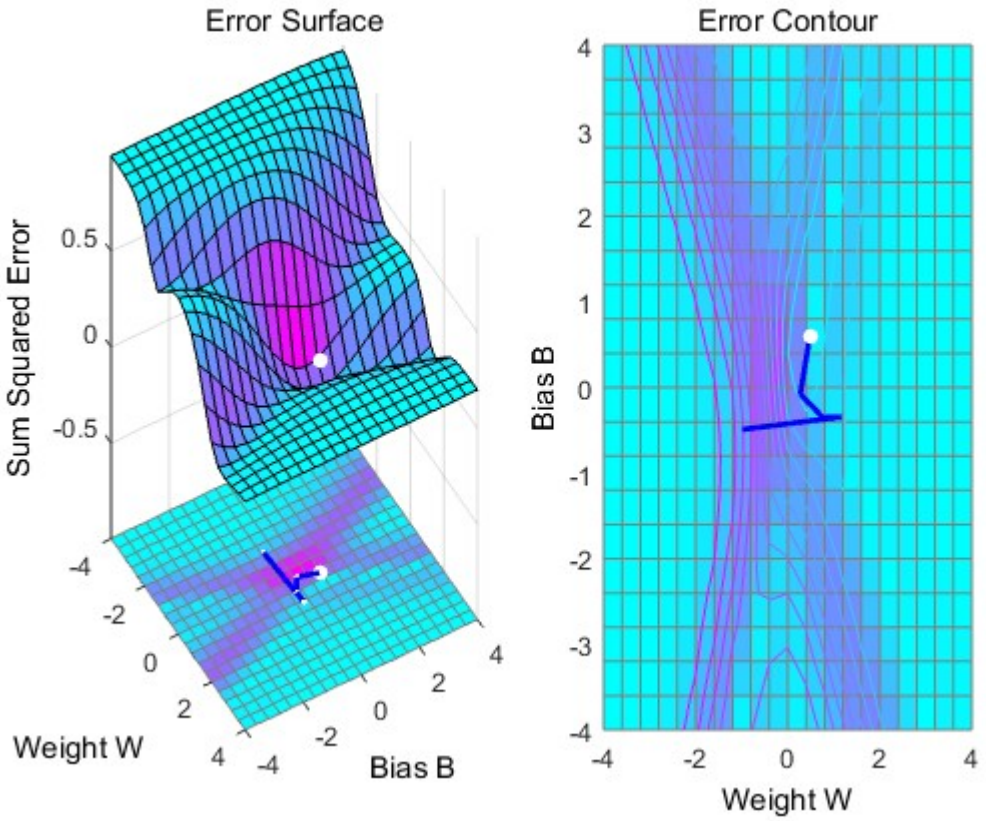


Abbildung 3.10 Weight des Neuronalen Netzes mit großer Lernrate

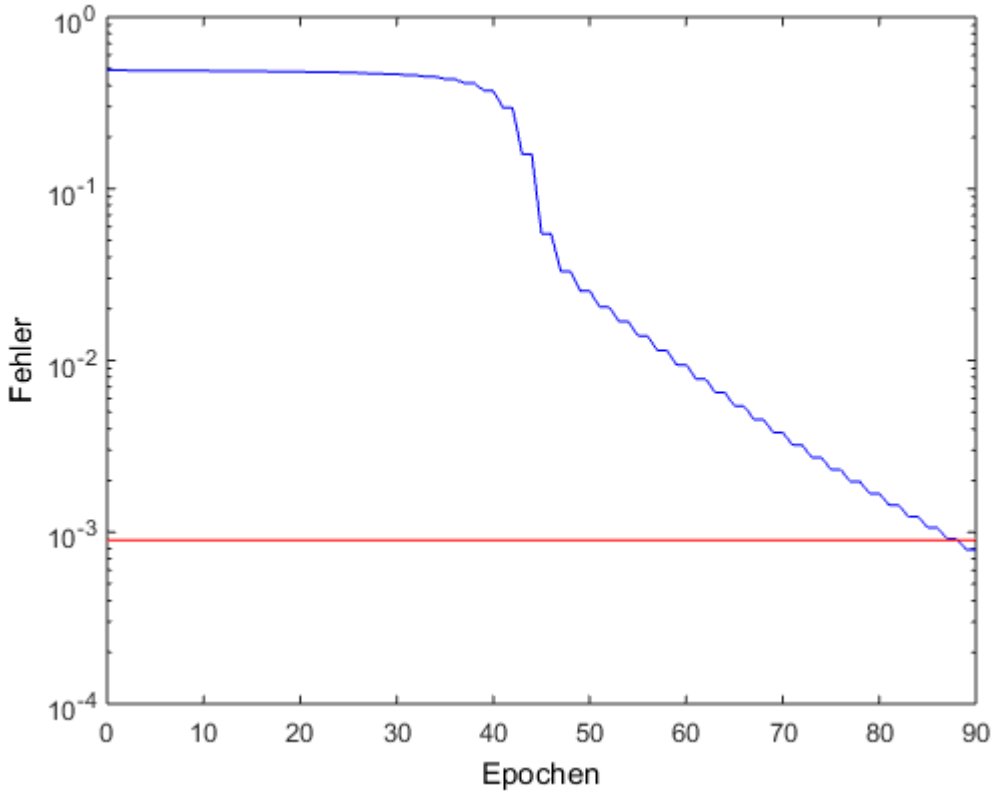


Abbildung 3.11 Fehler des Neuronalen Netzes mit kleiner Lernrate

Allerdings läuft ein Verfahren mit kleiner Lernrate auch nicht gut. Wie es in Abbildung 3.11

zeigt läuft der Prozess am Anfang sehr langsam. Die Veränderung ist sehr klein. Zudem hängt das Ergebnis mit solcher kleinen Lernrate noch von dem Anfangspunkt.

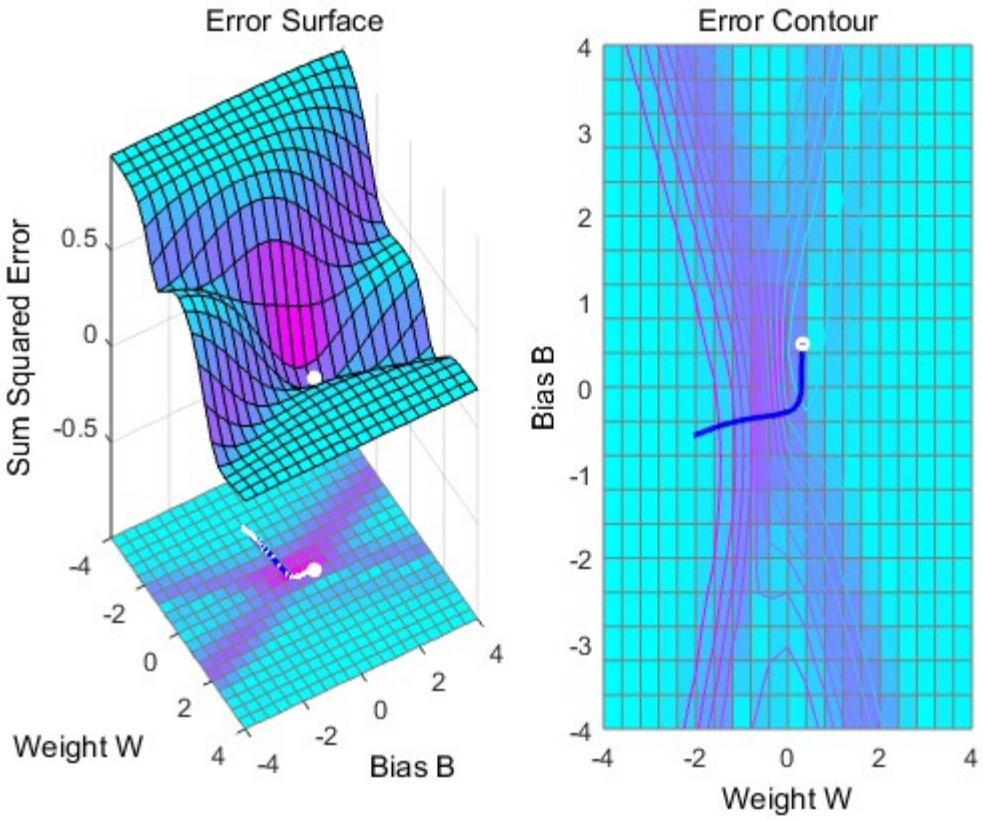


Abbildung 3.12 Weight des Neuronalen Netzes mit kleiner Lernrate

Hier wird der Einfluss der Zahl der Neuronen auf die Approximationsfähigkeit eines Neuronalen Netzes untersucht.

Der Difficulty Index beschreibt die Schwierigkeit, um eine Funktion dieser Daten nachzubilden. Von den Ergebnisse kann man sagen, dass die Anzahl der Neuronen die Schwierigkeit überstimmen muss. Abbildung 3.13 und 3.14 zeigen ein gutes Ergebnis. Die Zahl der Neuronen ist so ausgewählt, dass das Neuronale Netz die Punkte gut approximiert. Eine große Zahl der Neuronen bedeutet eine hohe Ordnung des Approximationsergebnisses des Neuronalen Netzes. In Abbildung 3.15 ist die Zahl der Neuronen der verdeckten Schicht gleich 9 und der Grad der Schwierigkeit gleich 1. Davon kann man sehen, dass die Approximation sehr präzise die vorhandenen Punkte approximiert. Allerdings ist der Trend dieser Punkte nicht gut nachgebildet wegen so hoher Ordnung. Andererseits ist die Approximationseingenschaft nicht gut mit einer kleinen Zahl der Neuronen. Wie in Abbildung 3.16 und 3.17 gezeigt kann das Neuronale Netz die Funktion nicht gut approximieren. Die Approximationsfähigkeit ist nicht genügend.

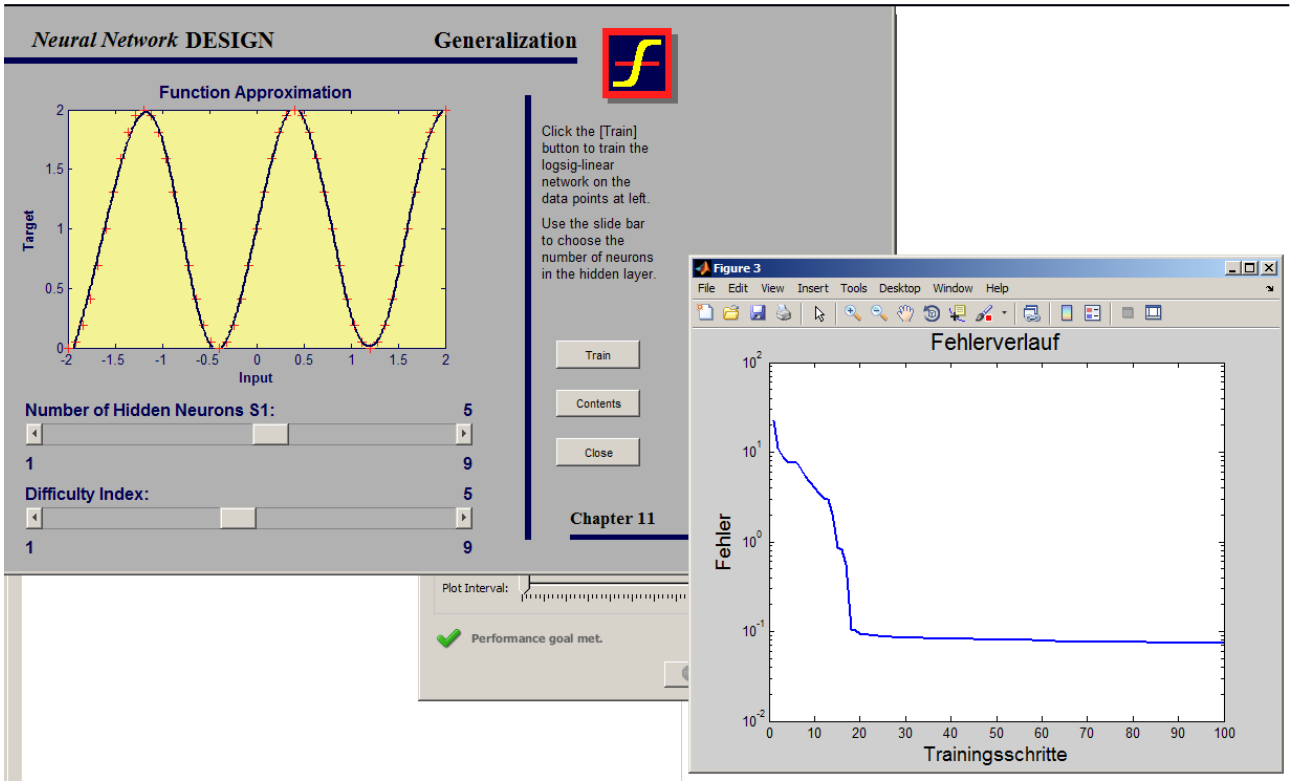


Abbildung 3.13 Approximation des Neuronalen Netzes

(Zahl der Neuronen: 5; Grad der Schwierigkeit: 5)

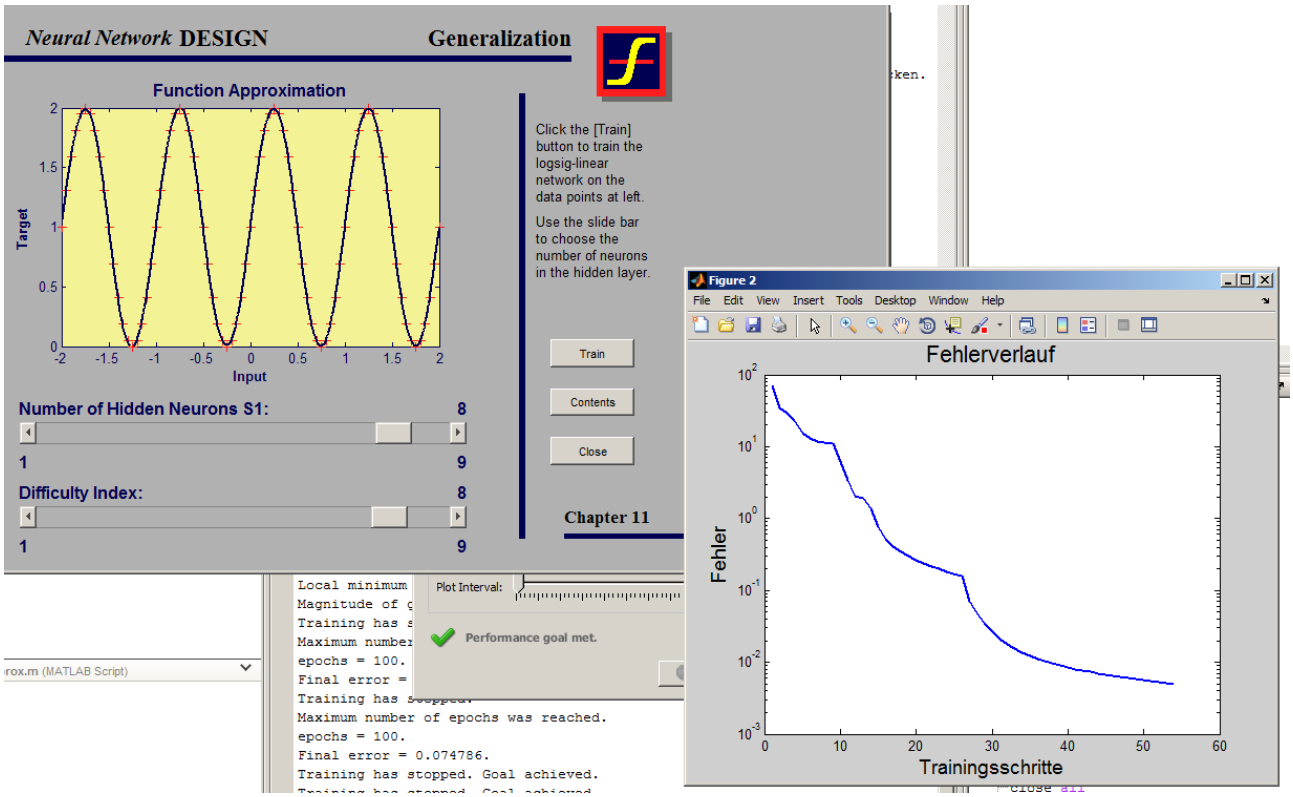


Abbildung 3.14 Approximation des Neuronalen Netzes

(Zahl der Neuronen: 8; Grad der Schwierigkeit: 8)

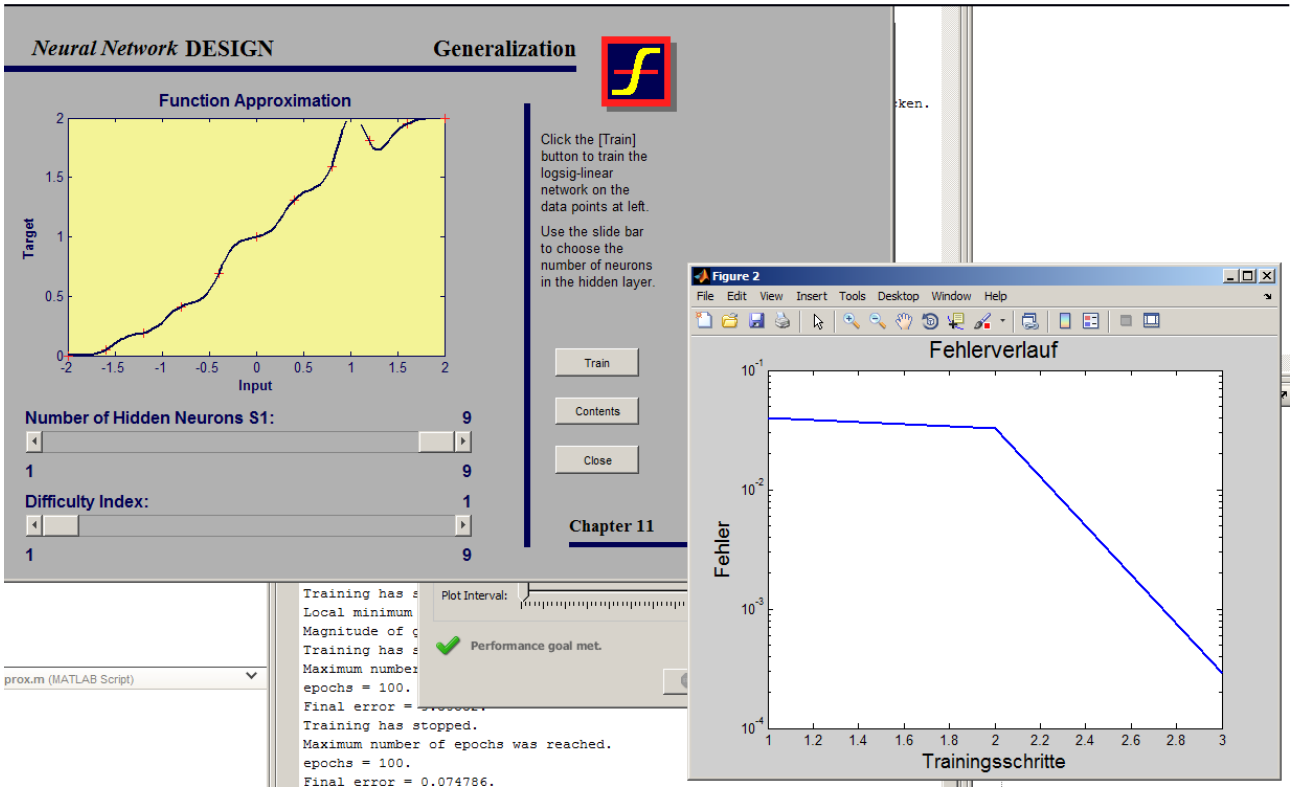


Abbildung 3.15 Approximation des Neuronalen Netzes

(Zahl der Neuronen: 9; Grad der Schwierigkeit: 1)

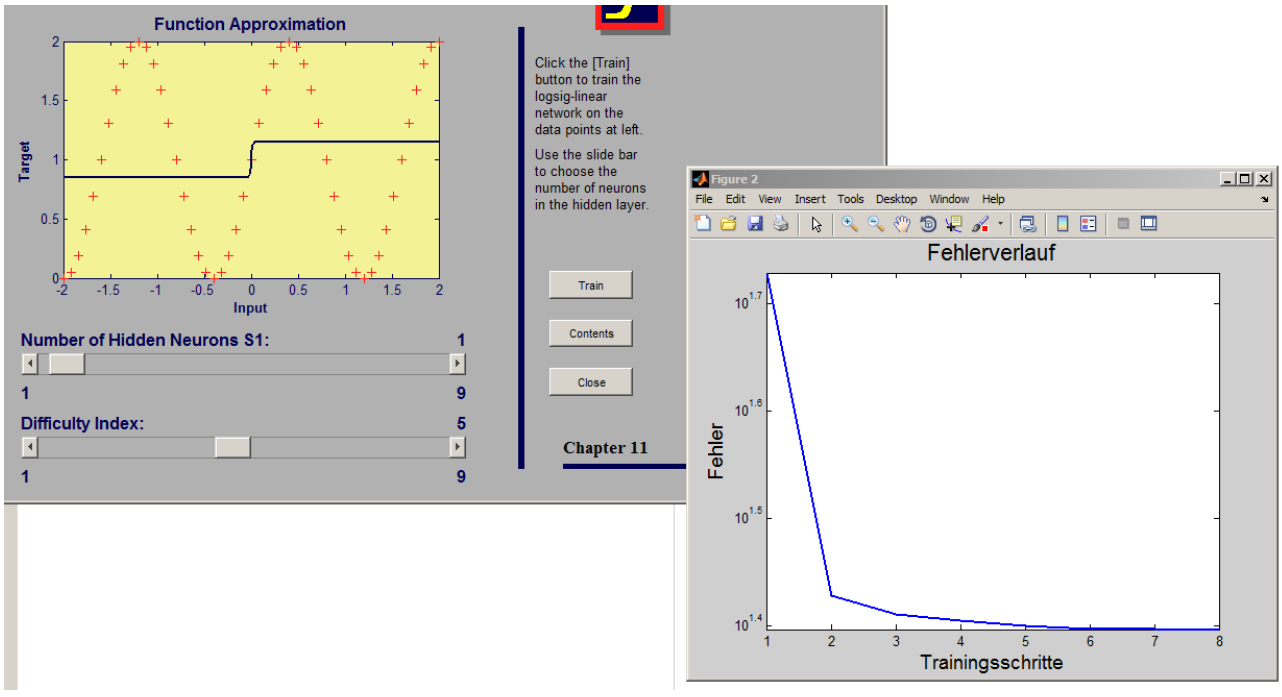


Abbildung 3.16 Approximation des Neuronalen Netzes

(Zahl der Neuronen: 1; Grad der Schwierigkeit: 5)

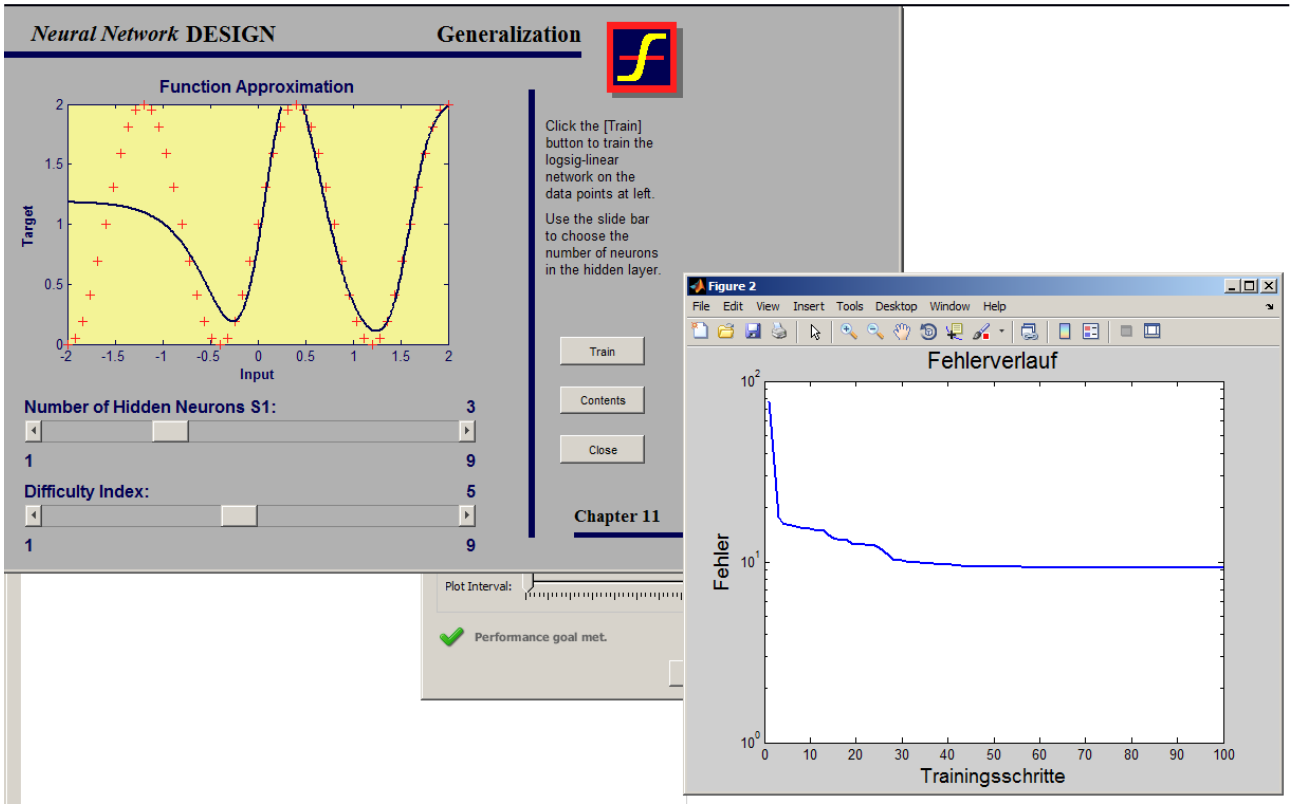


Abbildung 3.17 Approximation des Neuronalen Netzes

(Zahl der Neuronen: 3; Grad der Schwierigkeit: 5)

## Aufgabe 8: Aufbau und Training eines Neuronalen Netzes

Implementieren Sie nun das in Aufgabe 1 strukturell entworfene neuronale Netz zur Nachbildung der beschriebenen logischen Funktion und trainieren Sie dieses Netz mehrfach mit den möglichen Zeilen der Wahrheitstabelle. Erläutern Sie, was die Konfu-sionsmatrix angibt. Beschreiben Sie Ihre Beobachtungen bei mehreren Trainingsdurchläufen. Testen Sie auch, welche Werte ihr Netz ausgibt, wenn Sie als Eingangsdaten der Simulation die folgenden 3-Tupel verwenden.

| a | b | c |
|---|---|---|
| 0.1 | 0.1 | 0.9 |
| 0.1 | 0.9 | 0.1 |
| 0.9 | 0.1 | 0.1 |
| 0 | 0 | 2 |
| 0 | 1 | 3 |
| 1 | 3 | 3 |

**Antwort:**

Das Programm in MATLAB ist folgenderweise geschrieben:

```
u=[ 0 0 0;
    0 0 1;
    0 1 0;
    0 1 1;
    1 0 0;
    1 0 1;
    1 1 0;
    1 1 1]';
y=[ 0;0;0;0;1;0;0;1]';
t=[ 0.1 0.1 0.9;
    0.1 0.9 0.1;
    0.9 0.1 0.1;
    0 0 2;
    0 1 3;
    1 3 3]';
net=feedforwardnet(2);
net.trainparam.goal=1e-10;
net.divideFcn='';
net_train=train(net,u,y);
Ausg_train=sim(net_train,u);
Ausg_test=sim(net_train,t)
plotconfusion(y,Ausg_train);
```

Die Konfusionsmatrix wird folgenderweise geplottet. Es gibt 8 Daten und insgesamt 2 Kategorien als Ausgang mit einem Wert 0 bzw. 1 in den Trainingsdaten. Die Elemente in der Zeile entsprechen den Approximationsergebnisse aus dem Neuronalen Netz. Die Elemente in der Reihe entsprechen den präzisen Ausgänge aus den Trainingsdaten. In dieser Abbildung zeigen die ersten beiden diagonalen Zellen die Anzahl und Prozentsatz der korrekten Klassifizierung durch das ausgebildete Neuronale Netz. 6 Ausgangswerte 0 und 2 Ausgangswerte 1 sind beziehungsweise korrekt klassifiziert. Dies entspricht bzw. 75% und 25% der alle 8 Daten. Außerdem zeigen die rosa Zellen die Anzahl und Prozentsatz der falschen Klassifizierung. Es gibt weder Ergebnis 0 noch Ergebnis 1, das inkorrekt approximiert wird. Aus 6 Vorhersagen mit dem Wert 0 100% sind korrekt. Aus 2 Vorhersagen mit dem Wert 1 auch 100% sind korrekt. Aus 6 Fälle mit ursprünglichem Ergebnis 0 sind 100% richtig getippt als 0. Aus 2 Fälle mit ursprünglichem Ergebnis 1 sind auch 100% als 1 vorhergesagt. Zusammenfassend kann man sagen, dass 100% der Vorhersagen korrekt sind.

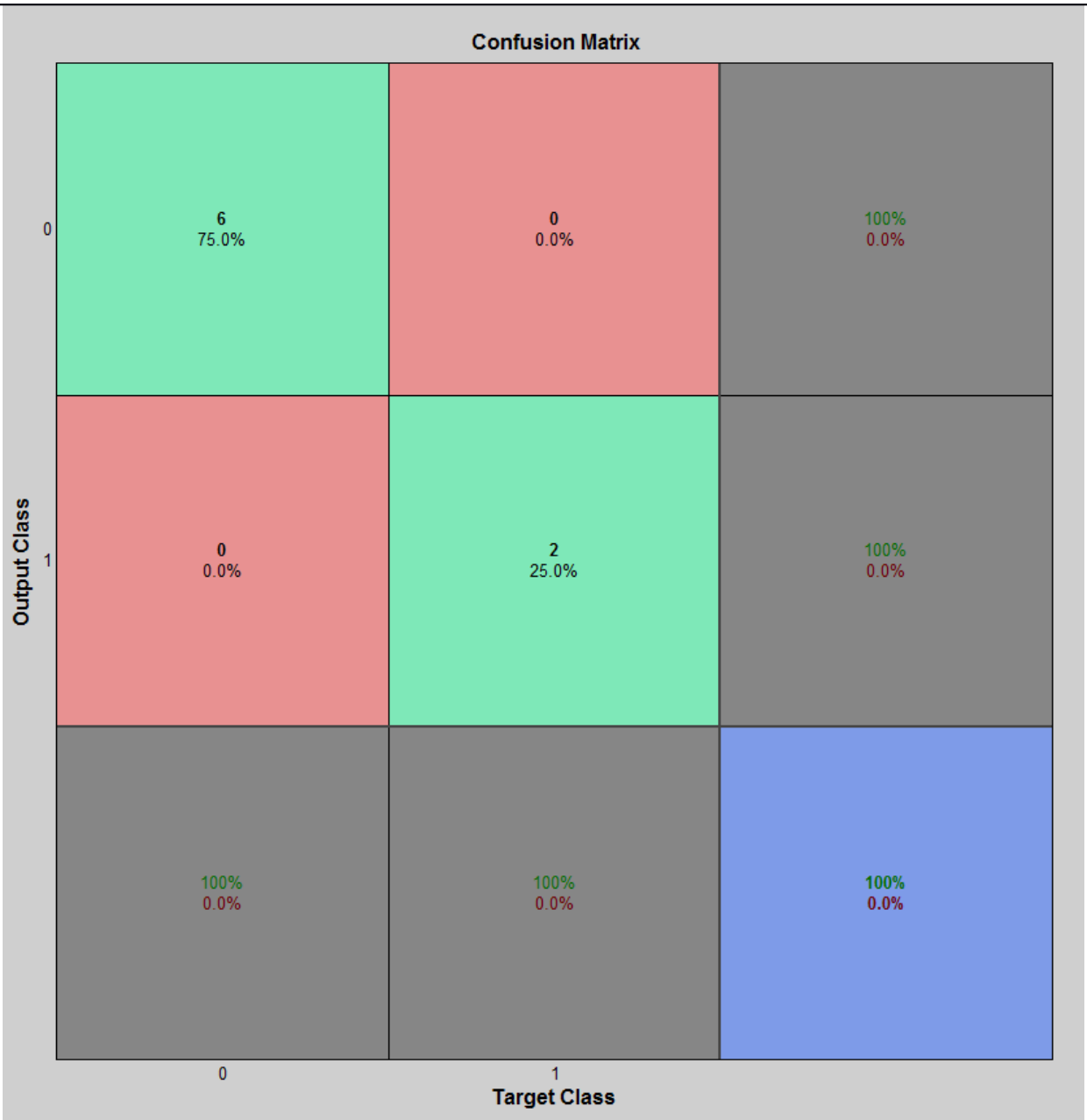


Abbildung 3.18 Konfusionsmatrix

Das Ergebnis des Tests mit dem Netz ist in folgender Abbildung aufgelistet. Die ersten drei 3-Tupel ([a,b,c] = [0,1, 0,1, 0,9],[0,1, 0,9, 0,1],[0,9, 0,1, 0,1]) sind innerhalb des Bereiches der Trainingsdaten. Die Ergebnisse dieser Tupel sollen nach der grundlegenden Funktion circa 0, 0, 1 sein. Nach dem Vergleich sind die Testergebnisse noch akzeptierbar. Die restlichen drei 3-Tupel ([a,b,c] = [0,0,2],[0,1,3],[1,3,3]) sind außer dem Bereich der Trainingsdaten. Die entsprechenden Ausgangswerte sollen nach der Funktion genau 0, 0, 1 sein. Aber es ist deutlich, dass die Ergebnisse des fünften und sechsten Tupels gleich sind. Deshalb können wir sagen, dass die Verwendung des Neuronalen Netzes außerhalb des Bereiches der Trainingsdaten problematisch ist.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | -7.4910e-04 | -7.4914e-04 | 0.2924 | 0.2414 | 1.3371 | 1.3371 |

Abbildung 3.19 Ausgangsergebnis der Testdaten

## Aufgabe 9: Kalibrieren der Anlage

Die Differenzdruck-Sensoren zur Messung der Füllhöhen sollten vor der Versuchsdurchführung kalibriert werden. Verwenden Sie dazu das Programm /Matlab/T3Kalibrieren.m. Machen Sie sich klar, auf welche Parameter im Simulink-Modell sich diese Kalibrierung auswirkt.

**Antwort:**

Im Simulink-Modell existiert ein Bestandteil, der in Abbildung 3.20 dargestellt ist. Dieser Teil beschreibt die Verwandlung von den gemessenen Signalen in die richtigen Werte der Höhe. Diese Kalibrierung wirkt sich auf Steigung und Offset aus, die in Abbildung als Faktorh und Offseth dargestellt. Durch die Kalibrierung werden diese zwei Koeffizienten geändert, damit man die Beziehung zwischen den richtigen Füllständen h und den Signalen der Sensoren festlegen kann.

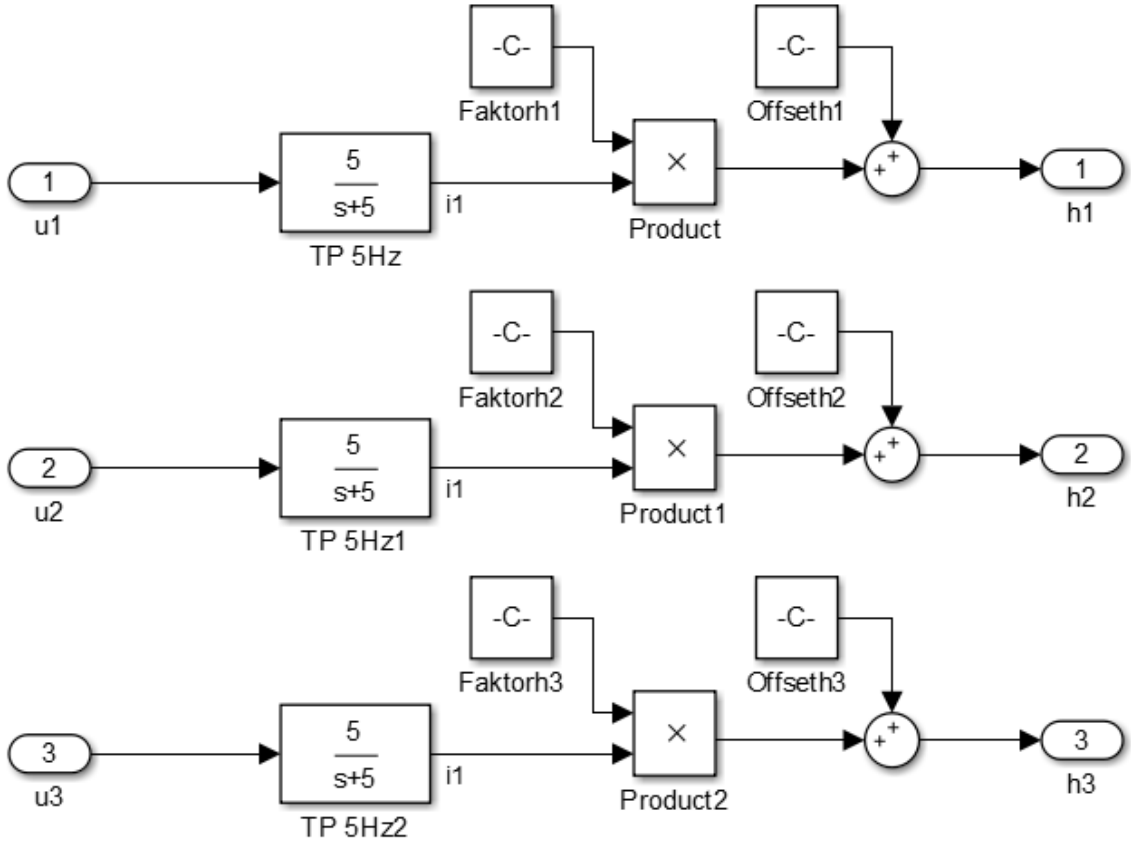


Abbildung 3.20 Verwandlung der Signale im Simulink-Modell

## Aufgabe 10: Trainieren eines Neuro-Reglers

Verwenden Sie ein Multilayer-Perceptron zur Approximation der Trainingsdaten. Überlegen Sie sich, ob das Neuronale Netz im für die Regelung relevanten Bereich das gewünschte Verhalten zeigen wird. Nimmt das Netz das gewünschte Verhalten an? Überlegen Sie sich die Ursachen für die gemachten Beobachtungen.

Setzen Sie nun das von Ihnen trainierte Netz als Regler am realen System ein. Fahren Sie die Tanks in Anfangshöhen, die von Ihnen zuvor trainiert wurden und starten Sie die Rege-

lung. Arbeitet der Regler zufriedenstellend? Charakterisieren Sie das Regelverhalten. Untersuchen Sie das Verhalten auf Störungen.

Stoppen Sie den Versuch und stellen Sie die Füllstände auf Anfangshöhen, die nicht von Ihnen trainiert wurden. Starten Sie die Regelung erneut. Arbeitet der Regler zufriedenstellend? Charakterisieren Sie das Regelverhalten. Was sind die Ursachen für dieses Verhalten?

Überlegen Sie sich, wie Sie das Neuronale Netz verändern können, um eine bessere Regelung zu erzielen.

**Antwort:**

Im Versuch erfassen wir zuerst durch Handregelung 8 Trainingsdaten und speichern wir die Daten in den Variablen Te und Ta ab. Das Programm um ein Netz zu trainieren ist folgenderweise geschrieben:

```
netgrund=feedforwardnet(10);
netgrund.divideFcn='';
netgrund.layers{1}.transferFcn='tansig';
netgrund.layers{2}.transferFcn='purelin';
netgrund.trainparam.goal=0.01;
netgrund.trainparam.epochs=100;
netgrund.trainparam.show=1;
net_train=train(netgrund,Te,Ta);
%net_ttrain=train(net_train,Te,Ta);
netzanzeige(net_train);
```

Das Kennfeld des trainierten Neuronalen Netzes ist in Abbildung 3.21 und 3.22 gezeigt. Diese Abbildungen sind die Ergebnisse nach nur 100 Epochen. Wenn die Regeldifferenz des Tanks gleich Null ist, was bedeutet, der Istwert der Füllhöhe ist gleich Sollwert, muss die Leistung der Pumpe Null sein. Aber in den Kennfelder sind die Leistungen der Pumpen im Punkt, wo die Regeldifferenzen Null sind, nicht gleich Null. Der Grund liegt darin, dass die Lerndaten nicht sehr gut sind wegen unserer Handregelung.

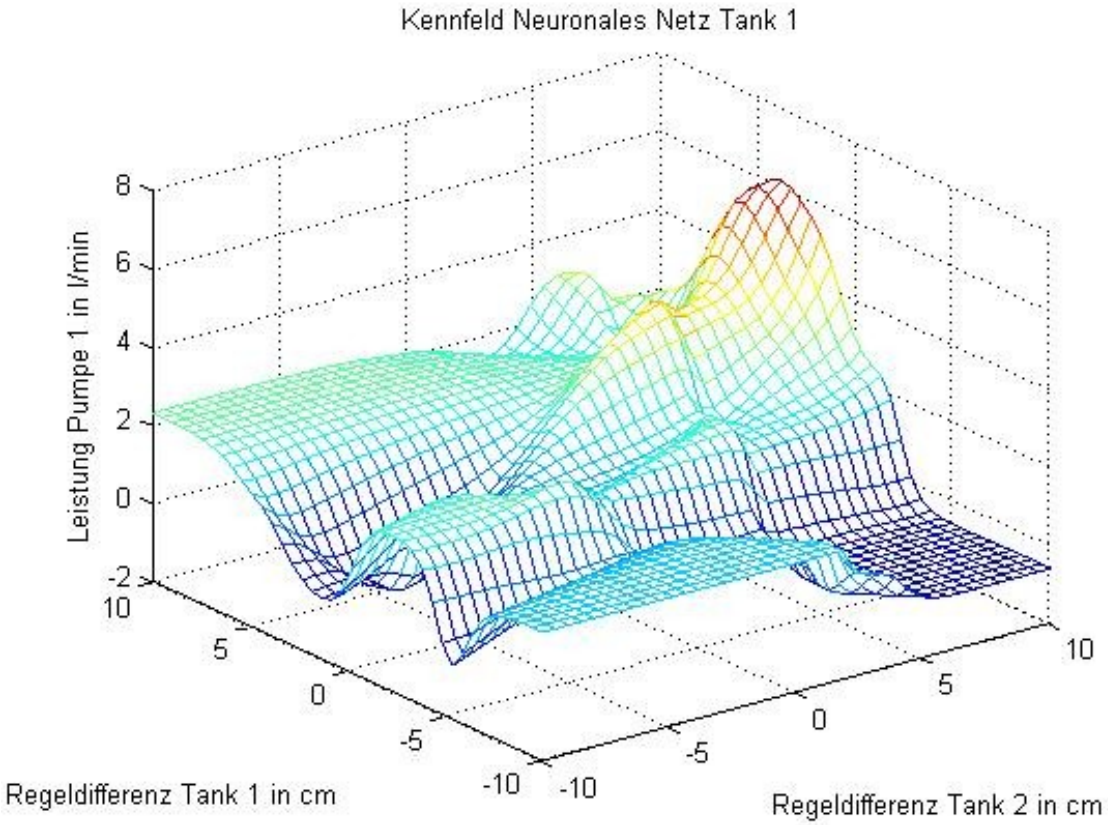


Abbildung 3.21 Kennfeld der Leistung von der Pumpe 1

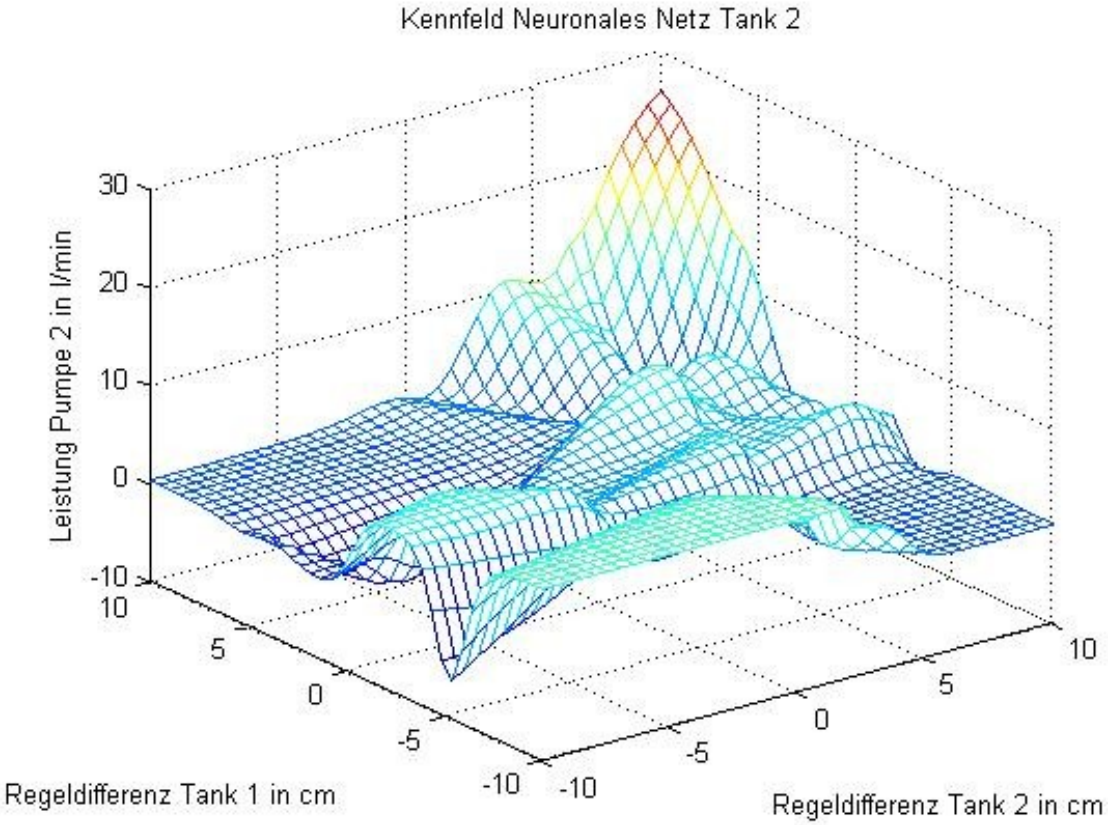


Abbildung 3.22 Kennfeld der Leistung von der Pumpe 2

Das trainierte Neuronale Netz wird jetzt als Regler am realen System eingesetzt. Das endgültige Regelungsergebnis mit den Anfangshöhen, die von uns zuvor trainiert wurden, ist in Abbildung 3.23. Der Regler kann sich gut verhalten und schnell die Sollhöhen durch die Veränderung der Pumpenleistungen erreicht. Obwohl existieren kleine Fehler zwischen dem Istwert und Sollwert, ist das Ergebnis noch zufriedenstellend und akzeptierbar.

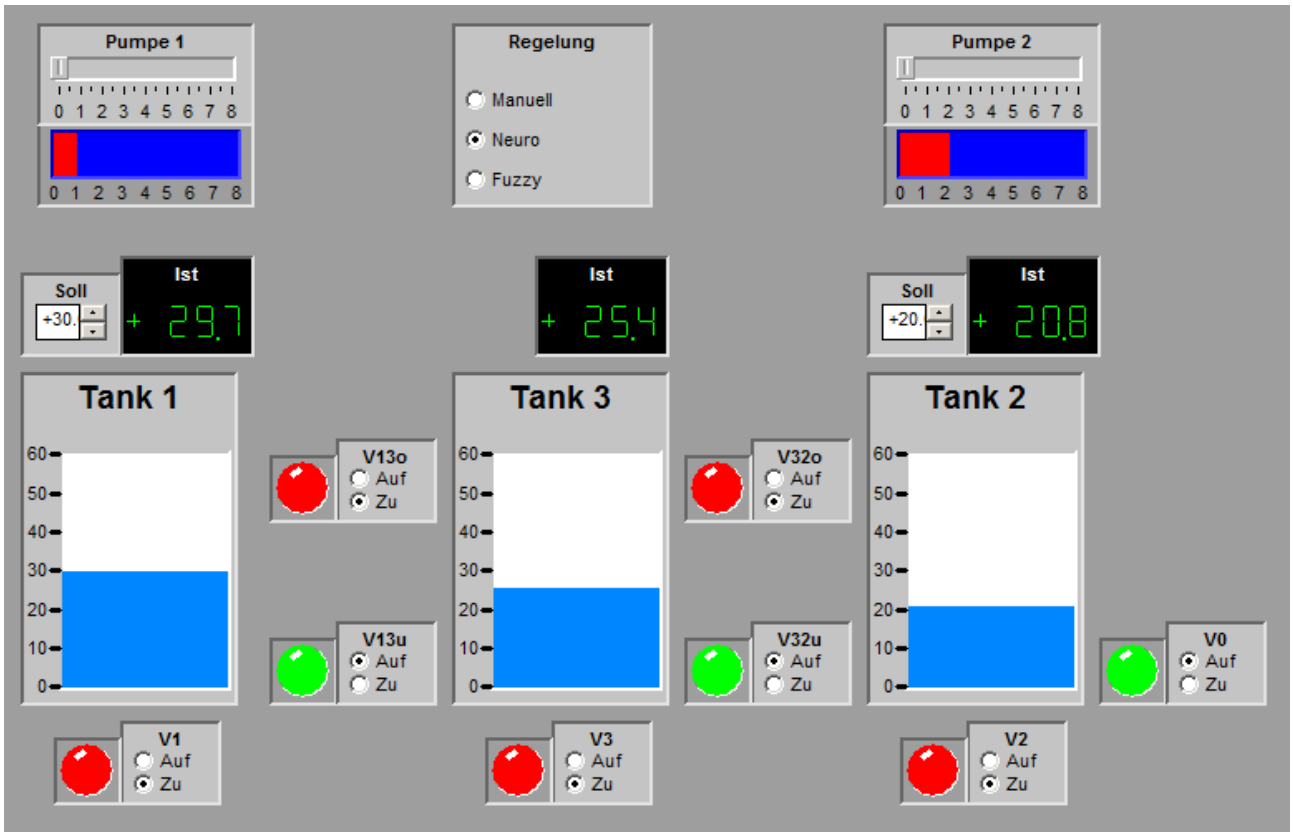


Abbildung 3.23 Füllstände nach der Regelung

Wenn eine Störung wie in Abbildung 3.24 tretet auf, kann der Regler deutlich nicht mehr gut funktionieren. Wegen des Öffnens des Ventils V1 sinkt die Höhe des Tanks 1. In beiden Tanks kann das Wasser die Sollhöhe nicht erreichen. Solche niedrige Fähigkeit gegen die Störungen liegt daran, dass diese Störungen nicht bei der Training berücksichtigt werden.

Wenn wir die Füllstände auf Anfangshöhen stellen, die nicht von uns trainiert wurden, arbeitet der Regler nicht zufriedenstellend. Weil die Anfangswerte außerhalb Trainingsbereichs, ist  die Regelung nicht unbedingt zuverlässig.

Verändern wir nun die Sollwerte der Höhen. Wie in Abbildung 3.25 gezeigt sind die Sollhöhe jetzt 35cm und 15cm. Letztlich erreicht die Höhe vom Tank1 und Tank2 einen Wert, der sehr nahe an dem Sollwert, trotzdem arbeitet der Regler noch problematisch. Die Fehler sind ein bissen groß. Das Neuronale Netz kann als Regler mit einen Sollwert, die nicht trainiert werden, nicht zufriedenstellend funktionieren.

Das Neuronale Netz kann durch mehr zuverlässige Trainingsdaten, deren Bereich größer sind, trainiert werden, um eine bessere Struktur erhalten zu können.

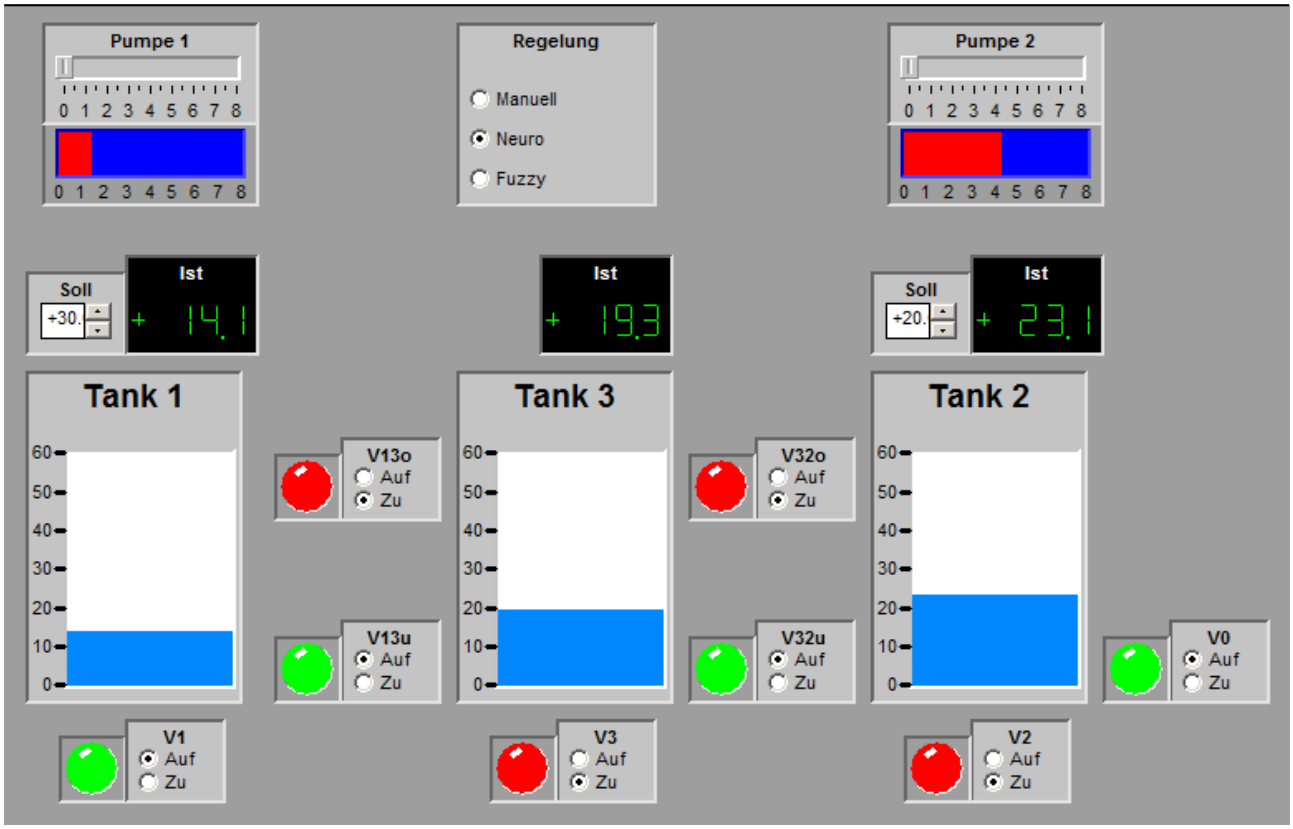


Abbildung 3.24 Füllstände nach der Regelung mit Störung

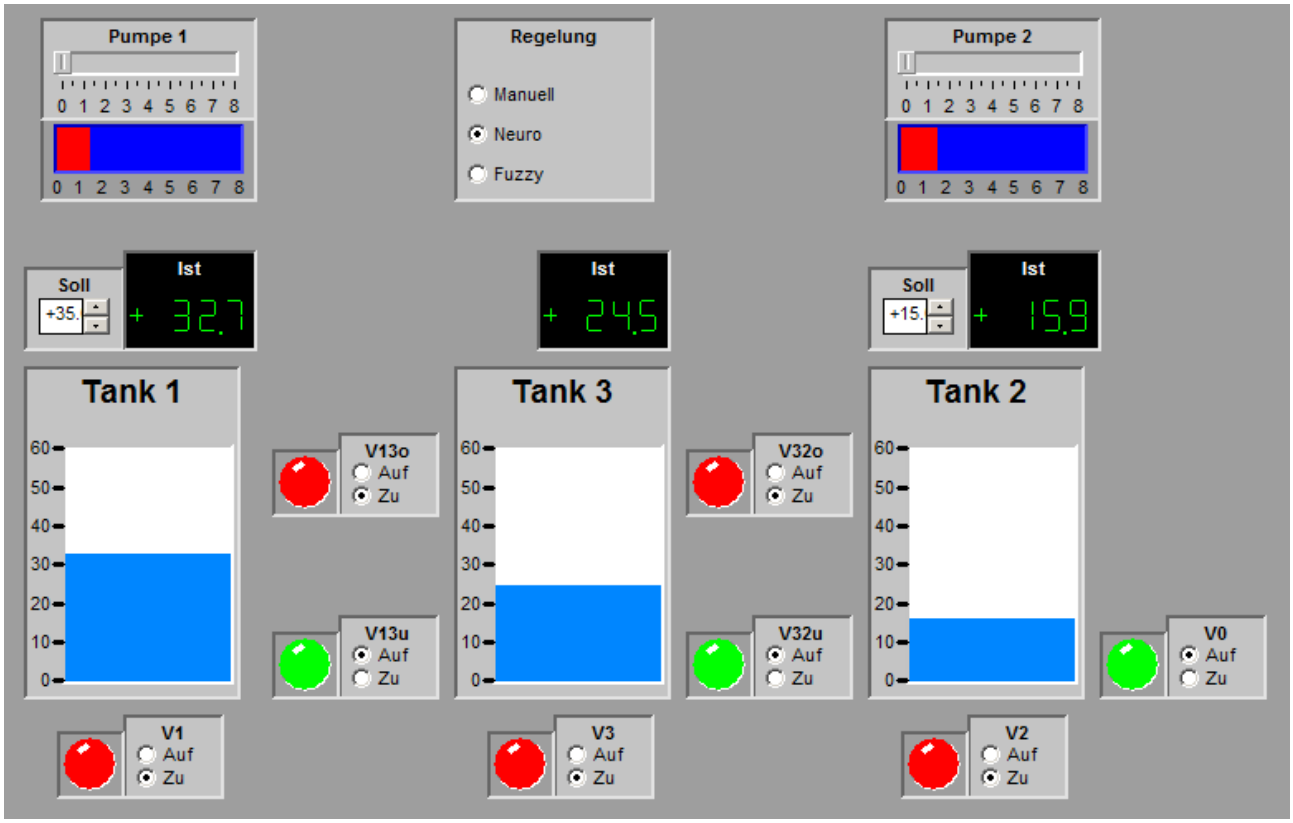


Abbildung 3.25 Füllstände nach der Regelung mit anderen Sollhöhen

## Aufgabe 11: Entwurf einer Fuzzy-Regelung

Entwerfen Sie einen Fuzzy-Regler für das 3-Tank-System. Die Füllhöhe des ersten Tanks soll auf $h_1$ =30cm und die Füllhöhe des zweiten Tanks auf $h_2$ =20cm geregelt werden. Die Regelung soll möglichst genau und möglichst schnell sein.

Sie können dabei davon ausgehen, dass das Verhalten des mittleren Tanks nicht berücksichtigt werden braucht. Verwenden Sie als Eingangsgrößen des Reglers die Differenz von Sollwert und Istwert $e = h_{soll} - h_{ist}$ . Geben Sie die Zugehörigkeitsfunktionen der verwendeten Ein- und Ausgangsgrößen sowie die zur Regelung benötigten Wenn-Dann-Regeln an.

**Antwort:**

Für den Entwurf von Fuzzy-Reglern steht die leicht zu bedienende Oberfläche zur Verfügung, welche durch die Eingabe von *fuzzy* in das MATLAB-Fenster gestartet werden kann.

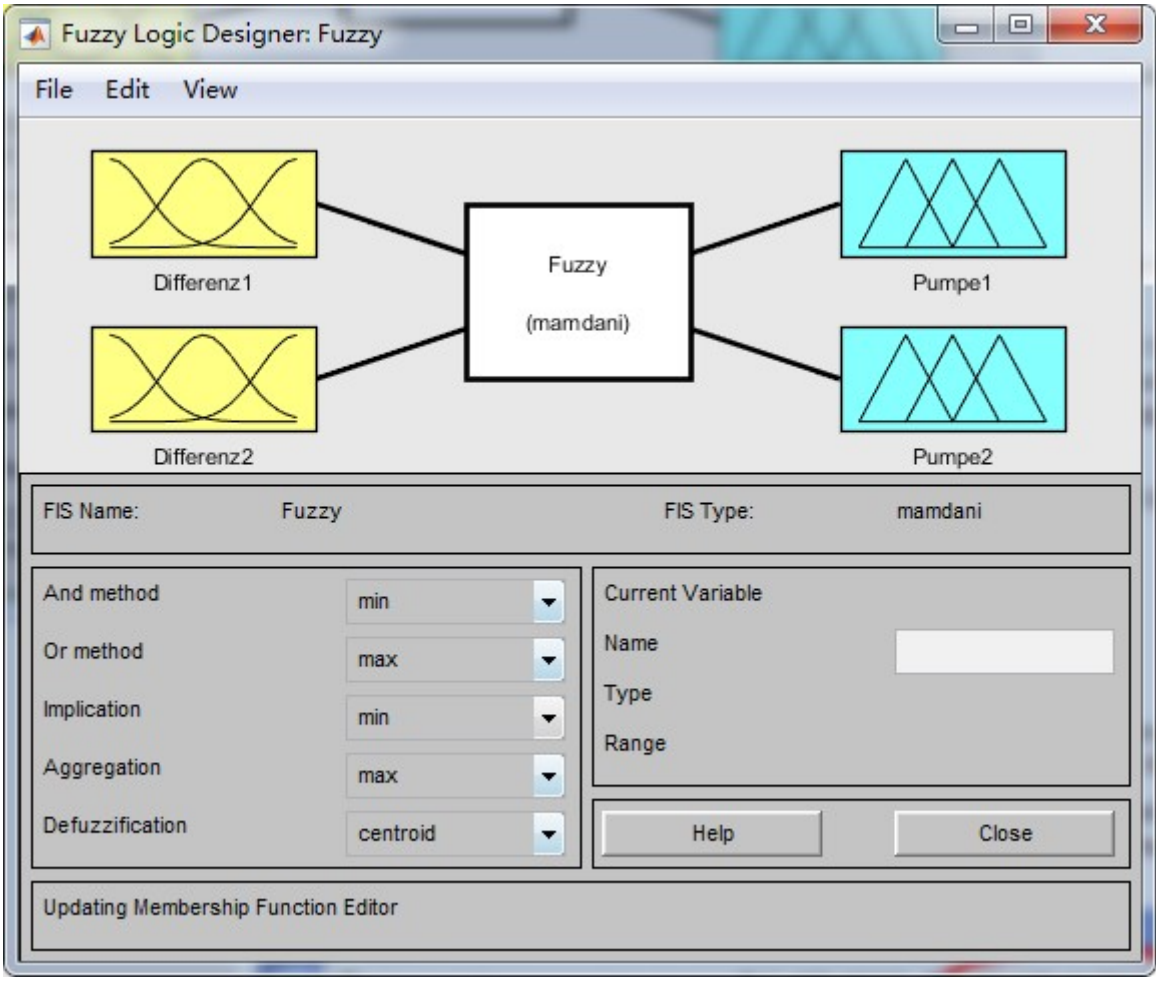


Abbildung 3.26 Oberfläche des Fuzzy-Reglers

Beim Entwurf des Fuzzy-Reglung werden die 2 Füllhöhenabweichungen vom Tank 1 und Tank 2 ( nämlich $e_1$ und $e_2$ ) als Eingangsgrößen betrachtet. Hier werden sie „Differenz1“ und „Differenz2“ genannt. Die Volumenströme der 2 Pumpen ( nämlich $q_1$ und $q_2$ ) werden als Ausgangsgrößen betrachtet. Hier werden sie „Pumpe1“ und „Pumpe2“ genannt.

Unter dem Menüpunkt *Edit* stehen Befehle zum Hinzufügen weitere Ein- und Ausgangsgrößen zur Verfügung. In dem **Membership Function Editor** werden die Zugehörigkeitsfunktionen der Ein- und Ausgangsvariablen definiert. Im Versuch werden die 3 linguistischen Werte „zuviel“, „mittel“ und „zuwenig“ bei der Eingangsvariablen genommen. Bei der Ausgangsvariablen werden die 3 linguistischen Werte „aus“, „halten“ und „ein“ genommen.

Beim 1. Versuch wird der Grundtyp als Dreieck gewählt. Die Breite von „mittel“ wird sehr groß gesetzt (hier $\pm 5$ cm ). Und die Zugehörigkeitsfunktionen von der Ausgangsvariablen werden wie folgende gesetzt.

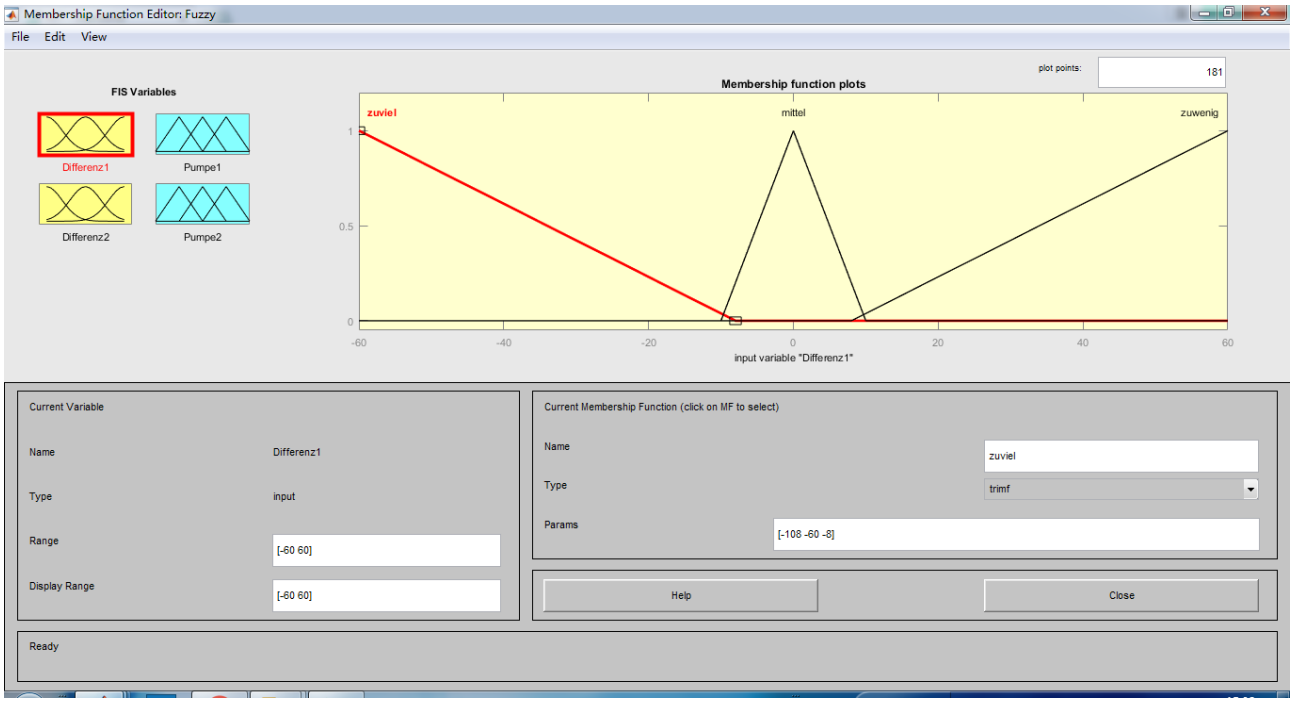


Abbildung 3.27 Zugehörigkeitsfunktion der verwendeten Eingangsgrößen (Versuch 1)

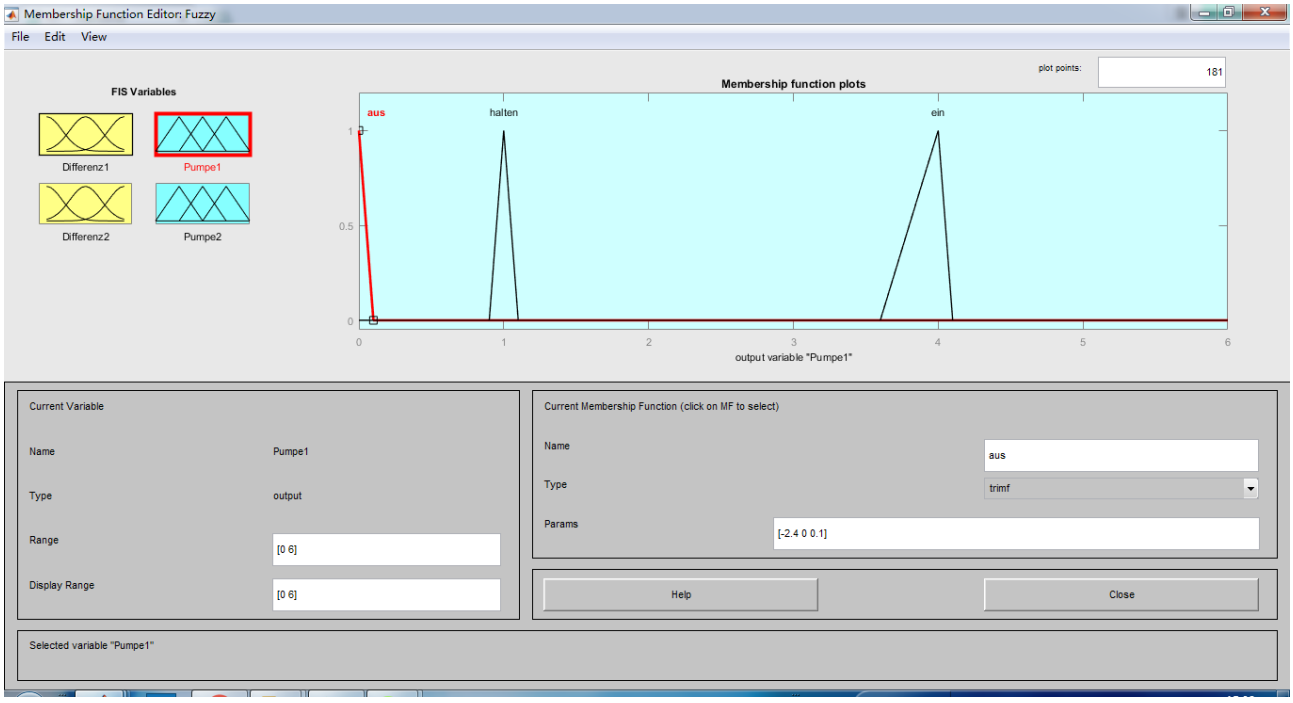

Abbildung 3.28 Zugehörigkeitsfunktion der verwendeten Ausgangsgrößen (Versuch 1)

In dem Rule Editor werden die Regeln in Form von Wenn-Dann-Regeln wie folgende editiert. Hier gelten 6 Regeln.

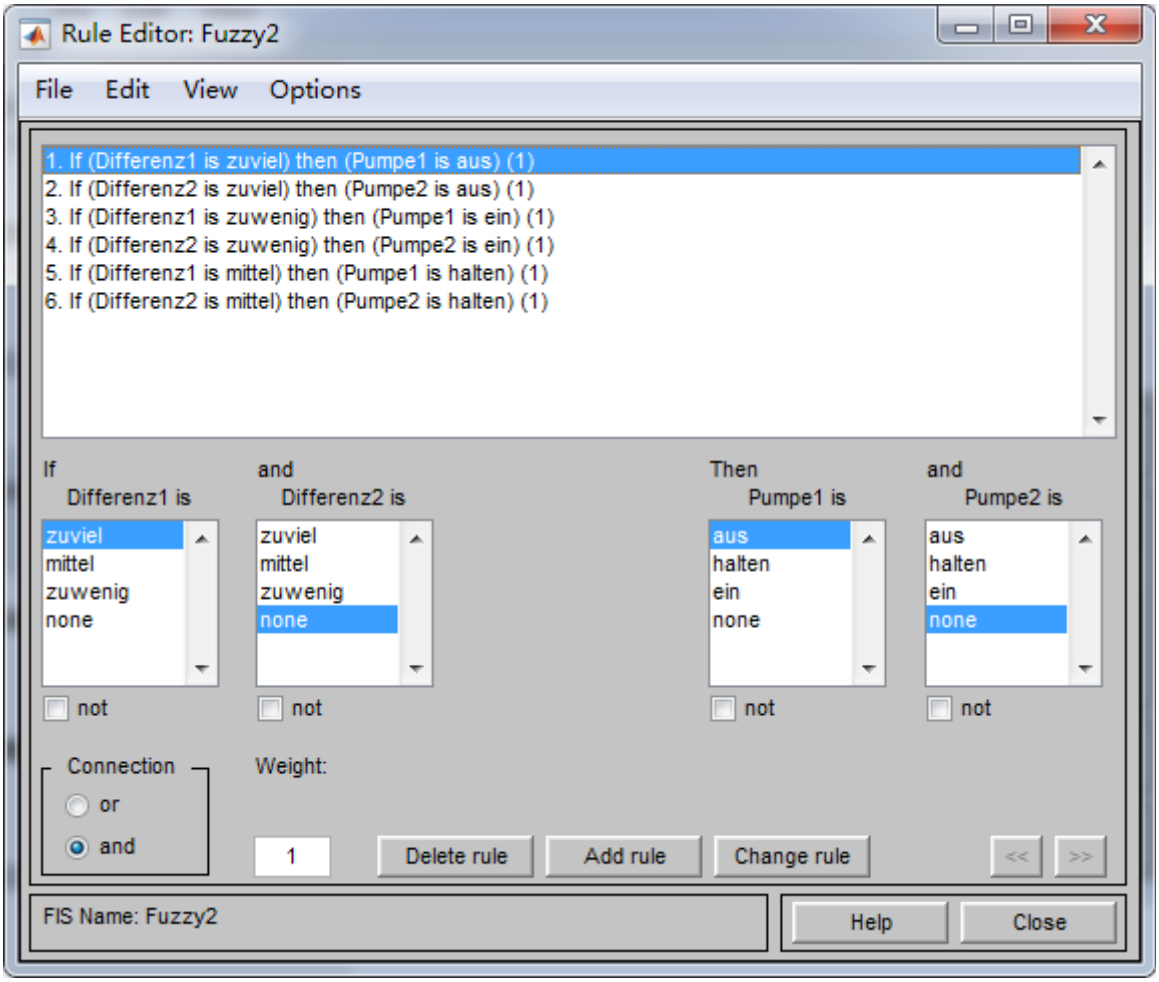


Abbildung 3.29 Wenn-Dann-Regeln

## Aufgabe 12: Fuzzy-Regler

Implementieren Sie den von Ihnen in Aufgabe 11 entworfenen Fuzzy-Regler und laden Sie ihn auf die **dSPACE**-Karte.

Arbeitet der Fuzzy-Regler zufriedenstellend? Nehmen Sie gegebennenfalls Änderungen an Ihrem Regler vor. Wie reagiert der Regler auf Störungen?

Der Regler wurde von Ihnen für bestimmte Sollhöhen entworfen. Testen Sie den Regler, wenn Sie andere Sollhöhen vorgeben. Arbeitet der Fuzzy-Regler zufriedenstellend? Ist der Regler stationär genau? Welche Veränderungen könnte man am Regler vornehmen, um das Regelverhalten zu verbessern? Versuchen Sie, eine solche Verbesserung zu implementieren.

**Antwort:**

Nach dem Entwurf wird das Programm des Fuzzy-Reglers auf die **dSPACE**-Karte geladen. Das Ergebnis ist allerdings sehr schlecht, denn die Differenz von „mittel“ ist zu groß

gesetzt. Die Füllhöhen schwanken sehr stark, ( beim Tank 1 zwischen 25cm und 35cm, beim Tank 2 zwischen 15cm und 25cm).

Deshalb haben wir die Zugehörigkeitsfunktionen beim 2. Versuch wie folgende verändert. Die Breite von „mittel“ wird sehr groß gesetzt (hier ± 5cm ). Aber beim Variablen „zuviel“ und „zuwenig“ werden die Trapez-Funktion verwendet. Und die Zugehörigkeitsfunktionen von der Ausgangsvariablen werden wie folgende gesetzt.

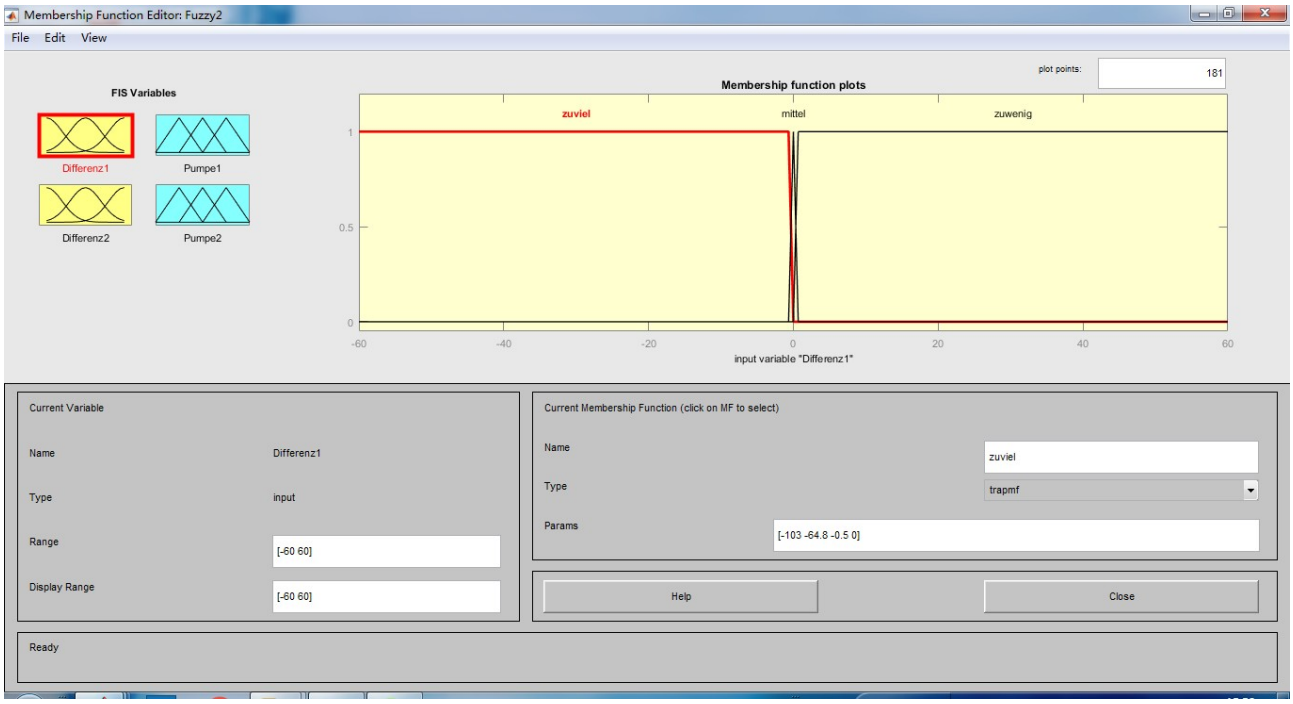


Abbildung 3.30 Zugehörigkeitsfunktion der verwendeten Eingangsgrößen (Versuch 2)

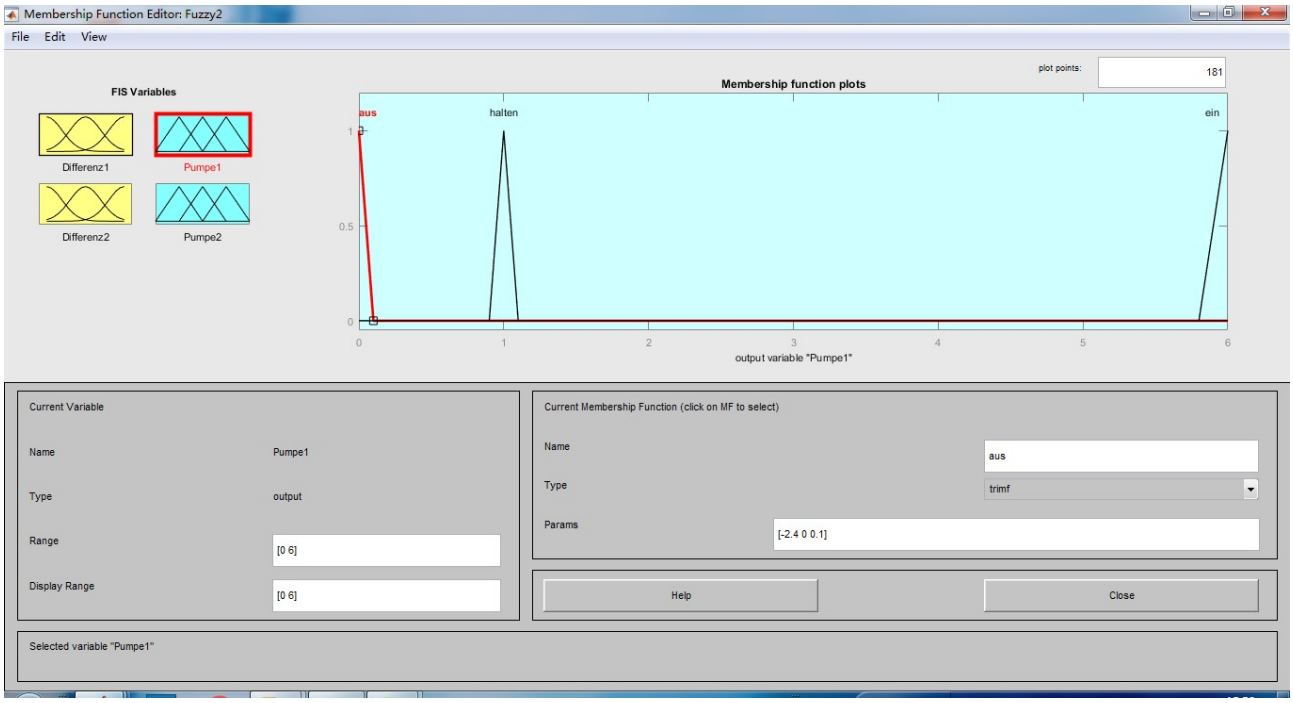


Abbildung 3.31 Zugehörigkeitsfunktion der verwendeten Ausgangsgrößen (Versuch 2)

Danach wird das neue Programm des Fuzzy-Reglers auf die **dSPACE**-Karte geladen. Das

Ergebnis wird offensichtlich verbessert. Beim im Voraus bestimmte Sollhöhe wird der stabile Zustand schnell erreicht. Aus den Unterschied zwischen die 2 Versuchen können wir das Fazit ziehen, dass die Unscharfe Menge bei weniger Anzahl von Stufen nicht zu breit gesetzt werden soll. Ansonsten findet Schwankung bei jeweiliger Stufe statt.

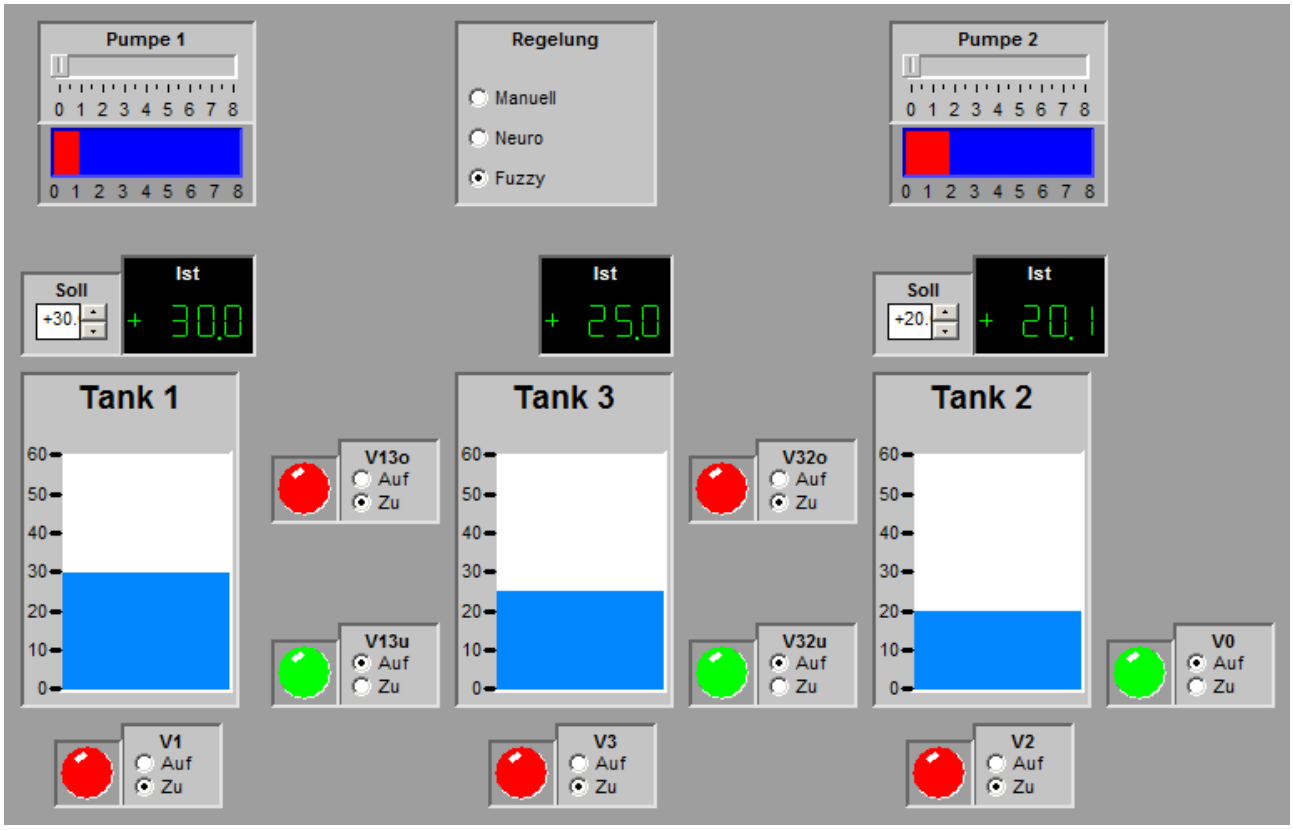


Abbildung 3.32 Füllstände der Fuzzy-Regelung (Versuch 2)

Danach wird der Fuzzy-Regler bei der Vorgabe anderen Sollhöhen getestet (hier beim Tank 1 40cm und beim Tank 2 10cm). Das Ergebnis wird wie folgende gezeigt. Offensichtlich funktioniert der Regler nicht so zufriedenstellend. Ein stabile Zustand wird erreicht, aber das System bei dieser Situation ist nicht stationär genau.

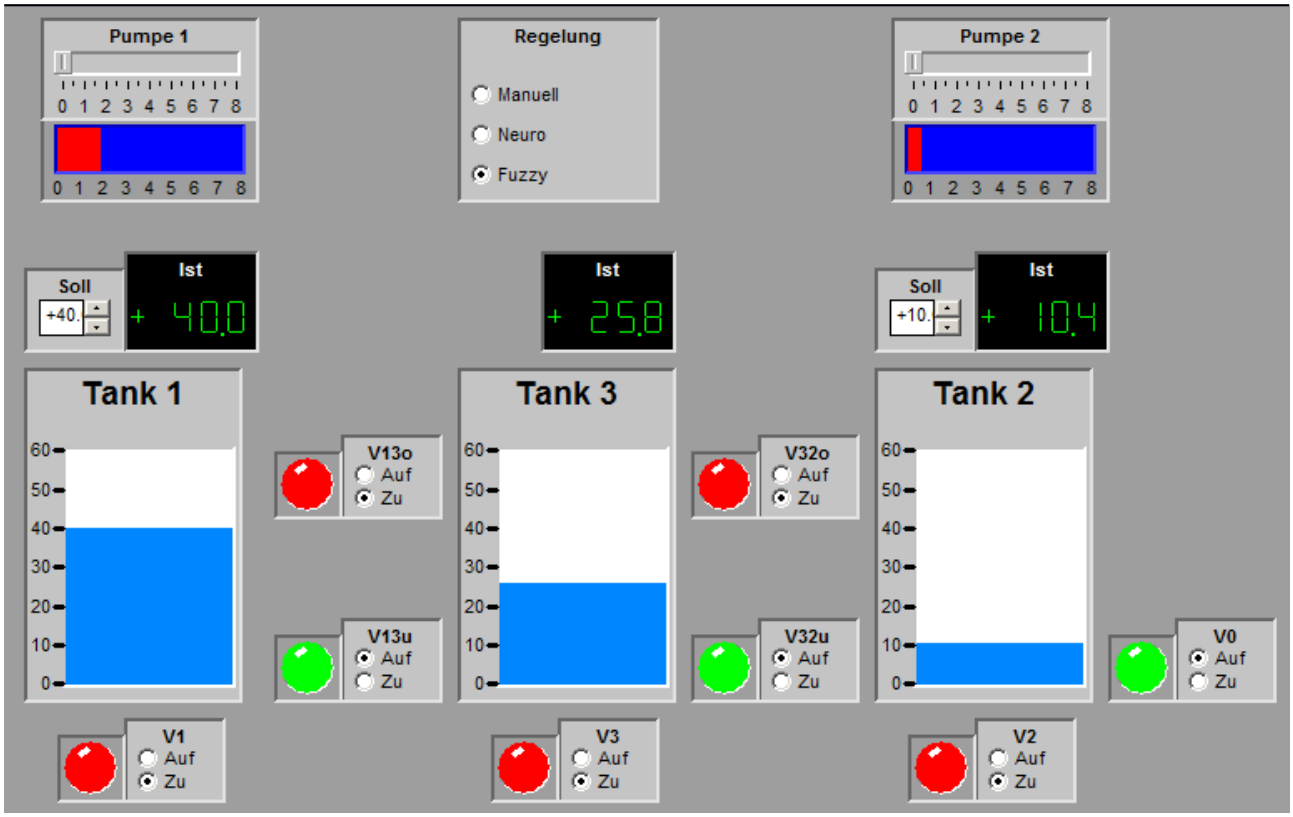


Abbildung 3.33 Füllstände der Fuzzy-Regelung mit anderen Sollhöhen

Zudem wird die Situation mit einer Störung getestet, funktioniert der Regler deutlich nicht. Hier wird das Öffnen des Ventil V1 als Störung betrachtet. Bei dieser Situation arbeitet der Regler auch nicht so zufriedenstellend.

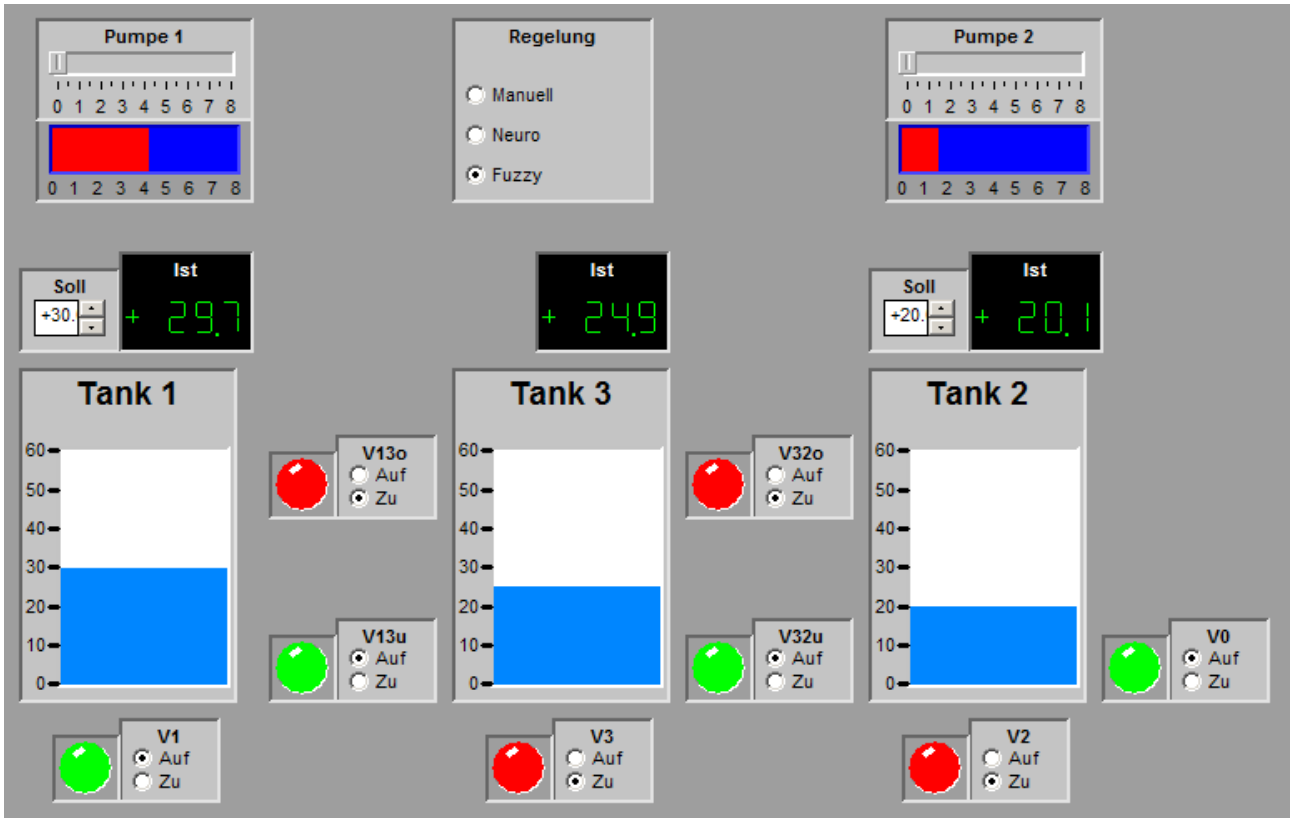


Abbildung 3.34 Füllstände der Fuzzy-Regelung mit Störung (Öffnen des Ventil V1)

Um das Problem zu lösen, stehen 2 Methoden zur Verfügung:

1. Der Hysterese Mode/ Zweipunktregelung (englisch Bang–bang control)

Zweipunktregelung ist die einfachste aller möglichen Regelungen. Der Zweipunktregler achtet lediglich darauf, dass die Ausgangsgröße sich innerhalb der vorgegebenen Grenzen befindet und schaltet die Regelung entsprechend ein oder aus. Der Regler ist sehr robust.

2. Ein zusätzlich integrierender Regler

Der Fuzzy-Regler ist stationär ungenau. Im Vergleich dazu ist der I-Regler durch seine unendliche Verstärkung ein genauer Regler. Es wird die stationäre Abweichung beseitigen.

In unserem Versuch wird die 2. Methode, also ein Fuzzy-Regler mit zusätzlichem I-Regler verwendet. Der I-Glied wird nach dem Fuzzy-Regler gesetzt.

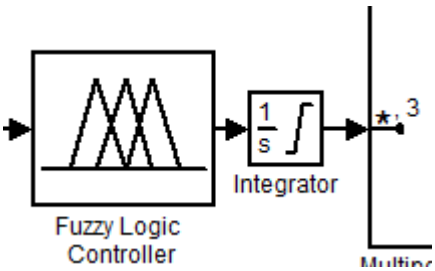


Abbildung 3.35 Struktur der Reglern

Denn die Abweichung wird durch dem I-Glied integriert, müssen wir die Zugehörigkeitsfunktion der verwendeten Ausgangsgrößen wie folgende verändern.

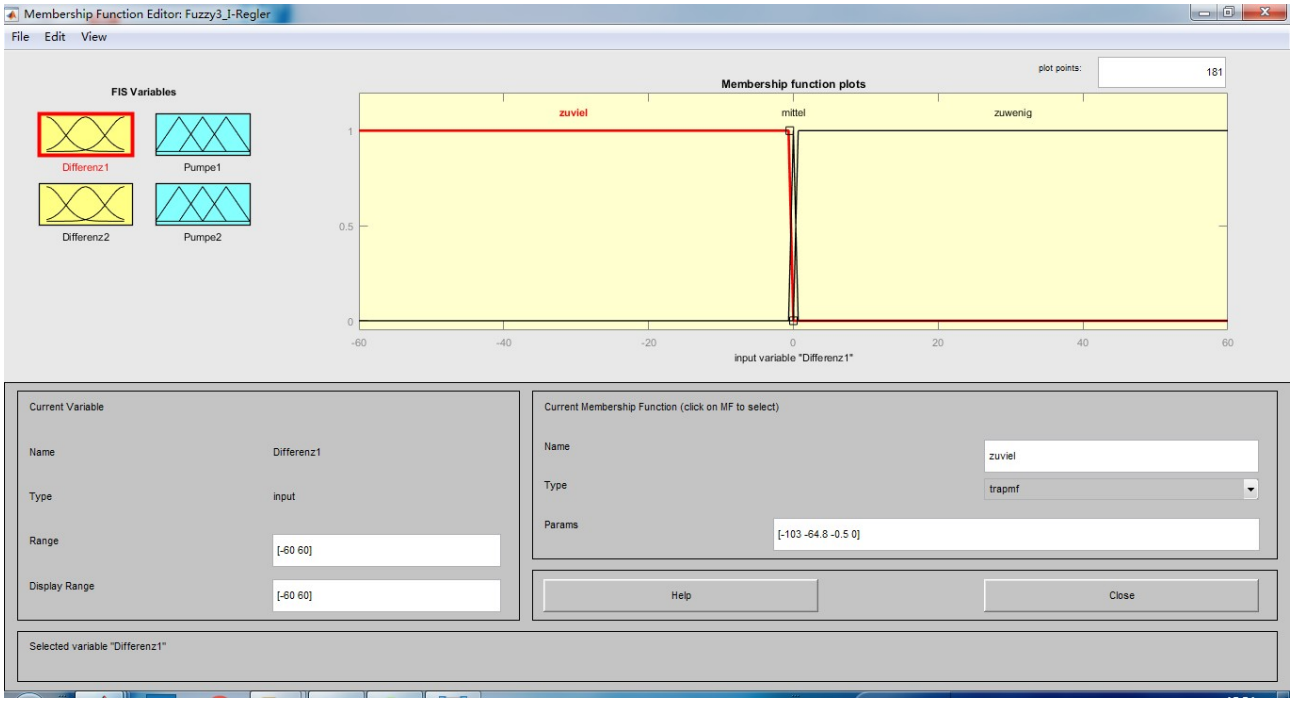


Abbildung 3.36 Zugehörigkeitsfunktion der verwendeten Eingangsgrößen (Fuzzy-Regler mit zusätzlichem I-Regler)

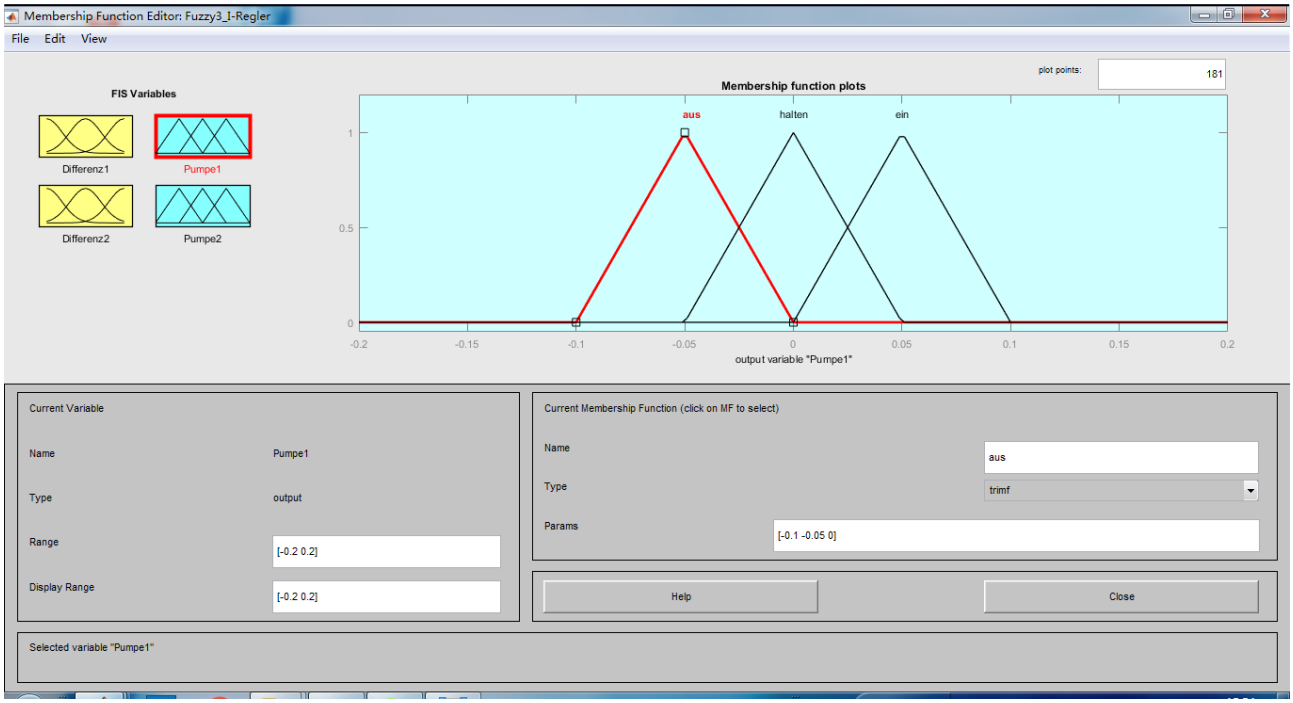


Abbildung 3.37 Zugehörigkeitsfunktion der verwendeten Ausgangsgrößen (Fuzzy-Regler mit zusätzlichem I-Regler)

Danach wird das neue Programm des Fuzzy-Reglers mit zusätzlichem I-Regler auf die **dSPACE**-Karte geladen. Es wird bei der Situation mit der Störung getestet. Im Vergleich zum Fuzzy-Regler ohne zusätzlichem I-Regler wird das Ergebnis verbessert. Denn jetzt ist der Regler stationär genau. Aber wegen des I-Reglers wird der Übergang ganz lang, es dauert viel Zeit, um den stationäre Zustand zu erreichen.

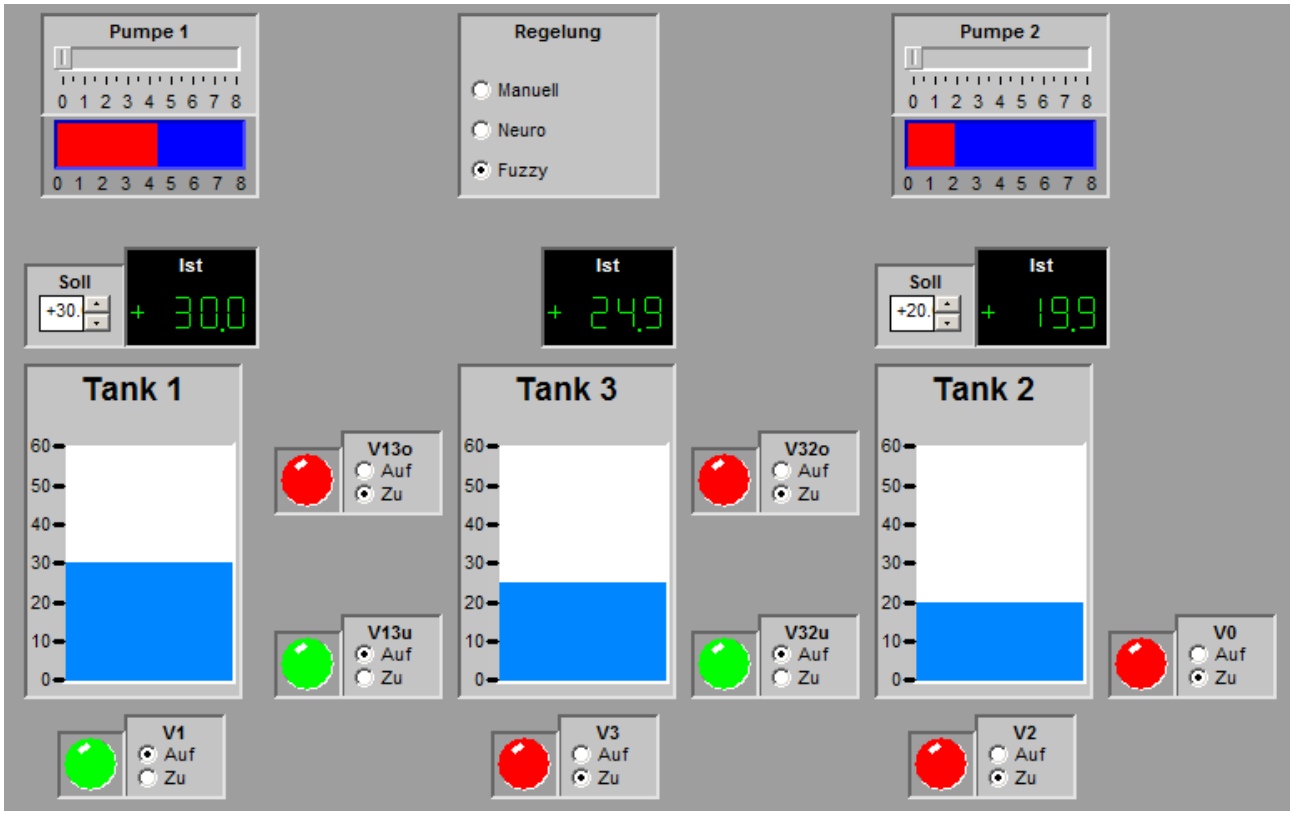


Abbildung 3.38 Füllstände der Fuzzy-Regelung mit zusätzlichem I-Regler

# 4. Literatur

[1] Material zum Versuch 5: 3-Tank-System, Praktikum Automatisierungstechnik, IRS