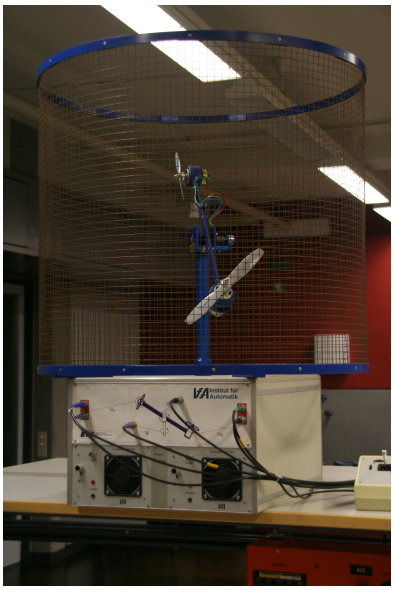
Automatic Control Laboratory, ETH Zürich
Prof. J. Lygeros

Manual revised by: A. Karapetyan, X. Guidetti, K. Koutroumpas, A. Milias-Argeitis, S. Richter, A. Zgraggen, M.Quack
Revision from: March 7, 2022

IfA Fachpraktikum - Experiment 1.4 :

Helicopter I – Fuzzy Logic

In this experiment the student will apply fuzzy logic to a nonlinear control problem. The idea of fuzzy control will be explained and experienced with a practical example. The student will learn the basic concept of fuzzy control and apply this to a model helicopter.



- **Preparation:**
  Study the theory of fuzzy control and calculate a simple example controller.
- **Lab session:**
  You will work with a fuzzy controller for a model helicopter and study its work principle. Various aspects of a fuzzy controller are analyzed.

For the preparation of this experiment at home you don't need any files to download.

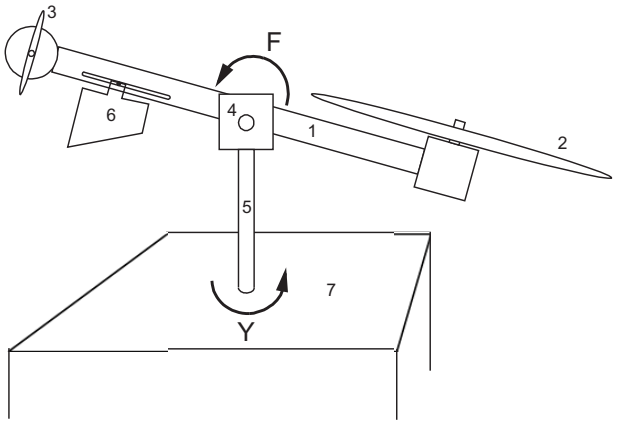
During the lab session you will need additional files that are provided locally on your machine:

`Fuzzy_Heli_RTW.mdl`	Simulink model of the control structure
`fuzzyPhi.fis`	Fuzzy rules for Φ control
`fuzzyPsi.fis`	Fuzzy rules for Ψ control

Contents

Chapter 1

Problem Setup and Notation



1: Crossbeam
2: Large Rotor
3: Small Rotor
4: Axis of Rotation F
5: Axis of Rotation Y
6: Counter Weight
7: Base

Figure 1.1: Sketch of helicopter

A model helicopter has been developed at the Automatic Control Laboratory with two rotors to control the vertical and horizontal movement. Each rotor has one input (motor), with the vertical (Φ) and horizontal (Ψ) angular positions measured. Due to its two input, two output structure, the system is called a MIMO-System (Multiple-Input-Multiple-Output). The objective of this experiment is to control the horizontal and vertical angular movements of the helicopter using a Fuzzy Logic controller. The students shall become familiar with the Fuzzy-Logic Control technique and learn about its advantages and disadvantages.

Since there is no lecture at ETH that covers the design of Fuzzy-Logic Controllers, no prior knowledge in this domain is assumed, and a theoretical introduction is provided in chapter 2. A description of the experimental setup and control structure is provided as well. Details of editing the Fuzzy parameters can be found in the appendix A. In chapter 2 you are also asked to complete an example which is necessary for the lab session. A series of practical exercises are then provided in chapter 3 to give an insight into Fuzzy Logic Controllers.

The experiment can be divided into two parts:

- In the first part, you will become familiar with the model and the software used to create fuzzy controllers. You will also design controllers for the horizontal and vertical movements

of the helicopter, and discover the effect of changing the controller's parameters on the system performance.

- The second part involves the design of a controller for the vertical movement of the helicopter model. This controller will be evaluated using performance criteria given in the document.

In practice Fuzzy Control is mainly used to control complex systems which are often difficult or impossible to describe mathematically. Often little is known about the internal functioning of these systems and only the input-output behavior is modeled (Black-Box Model). Such an approximate model will therefore be used in the experiments with the Fuzzy Controller. For a more detailed approach to the modelling of the helicopter the reader is referred to the instructions of the Fachpraktikum experiment 2.6 (Helicopter II - Lead-Lag)[1]. The Fuzzy Controller will be implemented in Simulink using the Fuzzy Toolbox. The communication between the controller and the helicopter model is established via a D/A-A/D converter.

[1] `http://people.ee.ethz.ch/~ifa-fp/wikimedia/images/2/2d/IfA_2-6_manual.pdf`

Chapter 2

Preparation@Home

2.1 Fuzzy-Logic Control

Classical control theory assumes that the process (plant) can be partially or completely mathematically modelled . Even though such mathematical descriptions are never exact, they are often sufficient to construct a standard controller (e.g. a PID). However there exist plants that cannot be modelled mathematically, (or only at prohibitive expense) which are difficult to control using classical techniques. Nevertheless it is possible describe the plant qualitatively and quantitatively, even in the absence of a precise mathematical model. An example using natural speech would be:

It is pretty cold and rains heavily.

Such statements are, by their nature, unprecise or so called *fuzzy*. Despite their fuzzyness, humans use words, such as little somewhat, quite, strongly, very, ..., frequently. In every day life these words can be quite useful, despite their lack of mathematical precision.

Having read the above statement about the weather no one would leave the house wearing a T-Shirt and sun glasses. Thus for choosing the appropriate clothing the above statement is precise enough. A more accurate statement, including exact temperature and the rainfall intensity per area and time, is unlikely to affect the choice of clothing strongly.

This ability to cope with fuzzy statements allows humans to act as controllers in situations that are not mathematically fully described. Furthermore, despite the fuzzy statements, the actions of humans taken in response to them are (in most cases) precise. An example of this shall be given to show how a human controller might work:

If it is cold and rainy, then the inside of the house has to be heated.

In industrial applications a human is often too slow or/and too expensive to control a process directly. The concept of fuzzy control evolved from the desire to use a similar approach implemented upon a computer. To achieve this, exact measurements have to be converted into fuzzy statements (fuzzification). This is done by using a set of "if then" rules. Theses statements are mapped to a corresponding reaction (inference), and finally the verbal reaction (e.g. to heat) has to be mapped to a precise manipulated variable.

2.1.1 Fuzzying human height measurements: an example of fuzzification

Fuzzification is the process of converting a *crisp* input value into a *fuzzy* one. The definition of a crisp set A is the one given by classical set theory: it consists of a finite or infinite number of elements belonging to some pre-specified superset called the *universe of discourse* X. By defining a *characteristic function* $\mu(x)$ membership of an element x from set X in the set A can be expressed:

$$\mu_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

On the contrary, a *fuzzy* set A has no "sharp boundaries": its characteristic function is generalized to a *membership function* and is allowed to take values between 0 and 1:

$$\mu_A : X \to [0, 1]$$

This means that an element x can have a certain *degree of membership* other than 0 and 1, e.g. equal to 0.4. In other words, in the case of fuzzy sets, the degree of membership expresses "how much" a certain x belongs to a given set A.

A *linguistic variable*, such as "human height", takes *linguistic* (or *fuzzy*) values, such as "*small*", "*medium*","*tall*" etc., which are fuzzy terms by nature. Thus, we can consider each of these fuzzy values as a certain fuzzy set.

Returning to the process of fuzzification of human height measurements, we can now see how it works: on the set of all possible input values (all possible human heights – our universe of discourse), we define a number of fuzzy sets, which correspond to the linguistic values of the variable "human height", with each set being represented by its membership function (see Fig. 2.1). A given crisp input value is fuzzified by determining its degree of membership in each of these fuzzy sets. For example, a 180cm man can be labeled to be 0.4 "medium" and 0.6 "tall".

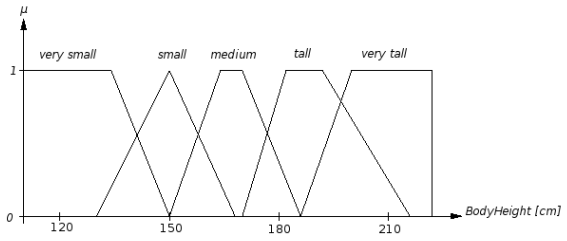


Figure 2.1: Fuzzyfication of the variable body height

Every membership function reaches in at least one point the maximum value 1. Commonly all membership functions of an input signal are drawn into one diagram. For every input signal such a diagram is generated. Figure 2.1 shows such a fictitious diagram for the variable "body height".

Theoretically, membership functions can be of arbitrary shape, however only four standard function shapes are commonly implemented.

Shape of the Membership Function (MF)	Label in MATLAB	compare to x in Fig. 2.1
Z	zMF	'very small'
S	sMF	'very tall'
Λ	triMF	'small'
Π	trapMF	'medium'

2.1.2 Inference: The use of fuzzy If-Then statements

Firing of a single rule

At the heart of fuzzy control lies the notion of *approximate reasoning*, which is used to represent and reason with knowledge which is expressed not in quantitative and exact form, but in a natural language, "fuzzy", form. The "elements" that we use to express this knowledge are called *atomic fuzzy propositions*. For example:

The controller input is big.

is such an atomic proposition involving the linguistic variable "controller input" and the value that this variable takes ("big"). The meaning of this proposition is defined by a fuzzy set that corresponds to the linguistic value "big" (which we suppose has already been defined), or – equivalently – a membership function μ_{big}, defined on the physical domain of the physical variable "controller input".

Complex (or compound) fuzzy propositions can be constructed from atomic propositions and *linguistic connectives* such as 'and', 'or', 'not' and 'if-then'. For example:

X is A and X is B
X is A or X is B
X is not A

The meaning of these compound propositions is given by interpreting the connectives 'and', 'or' and 'not' as logical operators acting on the fuzzy sets corresponding to the atomic propositions they connect. For example, the atomic statement X is A corresponds a fuzzy set defined on the domain of variable X, which we shall represent by its membership function $\mu_A(x)$. The same holds for the statement X is B. Then, one of the possible (and widely used) interpretations of X is A or X is B is to define a new fuzzy set with membership function $\mu_{A\cup B} = \max\{\mu_A, \mu_B\}$, for all x belonging to the domain of the variable X. Similarly, X is A and X is B is defined as $\mu_{A\cap B} = \min\{\mu_A, \mu_B\},\ \forall x$.

The following compound fuzzy statement:

if X is A, then Y is B

where A and B can be compound statements themselves, is called a *fuzzy if-then statement* and holds a central position in fuzzy control theory, since it is used to represent the rules governing the operation of a fuzzy controller. It is an example of a *fuzzy binary relation*, i.e., a fuzzy statement that defines a correlation between the values taken by the two variables X and Y.

Consider now a fuzzy if-then statement *if X is A, then Y is B* and a certain "crisp" input value for X (e.g. $x = 3$). What is the value of the output Y? Certainly the answer is not a simple number, since the if-then statement is not a classical relation, but a fuzzy one. Thus, we expect Y to belong to a certain fuzzy set (that is, we expect a "blurry" output value instead of a "crisp" one). How do we calculate this fuzzy set? First we have to *fuzzify* the crisp input, so that we can apply the operations that we know between fuzzy sets. The fuzzification is done as explained in the first paragraph, i.e. by determining the degree of membership of 3 in A (so we find $\mu_A(3)$). Suppose this is equal to 0.75. We also assume that a certain fuzzy set with membership function $\mu_B(y)$ corresponds to the statement *Y is B*. Then, we form a modified ("clipped") version of μ_B (which we shall call μ_{CB}) as follows:

$$\mu_{CB}(y) = \begin{cases} \mu_B(y) & \text{, if } \mu_B(y) \leq 0.75 \\ 0.75 & \text{otherwise} \end{cases}$$

,$\forall y$ in the domain of Y.

This procedure can be represented graphically as seen in Fig. 2.2:

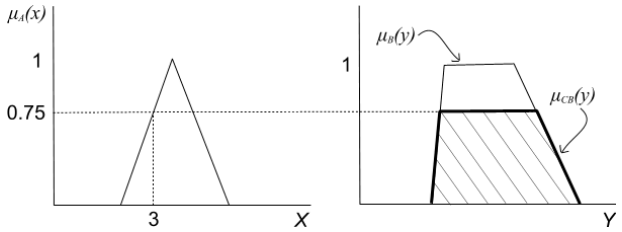


Figure 2.2: Clipping of membership function aka rule firing

The procedure described above, called *rule firing*, results in a fuzzy set μ_{CB}, which actually is a fuzzy form of "output". For example, consider the statement *if the room temperature is low, the radiator's temperature should be set high.* Given a crisp measurement of the room temperature, how much exactly is "*high*"? The answer in this case is not a crisp number, but a fuzzy set obtained by combining the input information with the fuzzy set that represents the value "*high*" of the linguistic variable "radiator's temperature". One of the possible ways to do this was presented above. In any case, we expect the membership function of the output to be higher at high rather than low temperatures. This is an example of how fuzzy reasoning works, but in actual problems the fuzzy set we just obtained is useless, unless we can extract a crisp output value from it. After all, even a human would have to decide eventually at which temperature to set the radiator. The process of obtaining this crisp output value is called *defuzzification* and will be explained later.

Firing of a set of rules

In the previous subsection we considered a single rule *if X is A, then Y is B* and a crisp input, in combination with a specific method of inference. The statements *X is A* and *Y is B* may be complex fuzzy statements, but this does not affect the method we presented. However, in a fuzzy controller we do not have only one rule, but a *set of rules*, which determine the output of the controller under various possible situations. In this case, instead of finding the result of a single rule, we need to know the combined result that the firing of many rules will have on the output of the controller.

As an example, consider the pair of rules:

R1: If the error e is negative big (NB), the output u is positive big (PB)
R2: If the error e is close to zero (Z), the output u is close to zero (Z)

Here the output variable u is affected by both rules. Also, each rule has the same input (e). In this case, we define the fuzzy set of the output as follows:

1. We fire each rule separately and obtain a fuzzy set corresponding to that rule. Let us call these sets $\mu_{R1}(u)$ and $\mu_{R2}(u)$.

2. We apply the 'or' operator to these sets to obtain the output of the set of rules. Interpreting the 'or' as the *max* operator (see above), this implies that we simply take the maximum of the two membership functions.

We could do exactly the same thing in the case u was affected not only by e, but another variable too. If, for example R1 was the same and the second rule was:

R2': If the change-of-error Δe is large, the output u is negative small

then we would proceed as before, firing each rule separately and taking the maximum of the two resulting membership functions. Of course, the same procedure generalizes easily when we have more than two rules affecting an output variable. A graphical representation of this procedure for the rules R1 and R2 mentioned above, is the following and can be seen in Fig. 2.3:

Graphical representation of the fuzzy inference procedure for a set of rules: the crisp input value of the error (e^{crisp}) is fuzzified and each rule is fired separately, giving the "clipped" membership functions μ_{R1} and μ_{R2} (indicated with the thick lines). Afterwards, the 'max' operator is applied to these functions to obtain the output membership function (shaded area on the graph).

2.1.3 Defuzzification: computation of a single crisp output

A fuzzy controller is a system that implements the fuzzy inference procedure presented above, in order to calculate crisp output values from crisp input values. Up to this point, we know that the firing of a set of rules results in a fuzzy set. During the defuzzification step, a crisp output

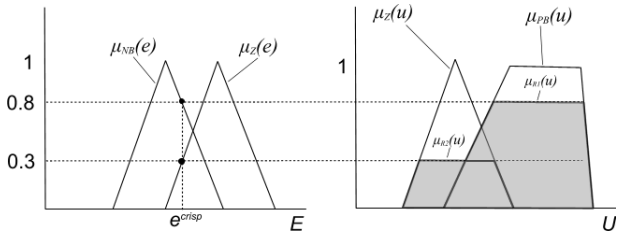


Figure 2.3: Fuzzy inference procedure

value is extracted from this fuzzy set. This is the controller output value, which will be applied to the system (there is also an intermediate step of scaling, which we ignore here).

There are many defuzzification methods (center-of-area, center-of-sums, first-of-maxima, mean-of-maxima etc.). They can be classified into two broad categories:

1. Algorithms based on the "most plausible solution".
2. Algorithms that make a compromise among different solutions.

Representative of the first category is the "*mean-of-maxima*" method. According to this, one first locates the x-axis point(s) where the output membership function is maximum, and takes their mean value (in case they are more than one).

The best-known method from the second category is the "*center-of-gravity*". In this case the area underneath the resulting output membership function is taken into account. The crisp output value is obtained by calculating the center of gravity of this area (you can think of it as a 2-D mass on the plane and apply the well-known formula from physics) and keeping only the x-coordinate of the resulting point. This method is used in our experiment. (In order to increase the speed of the computation, Matlab calculates the weighted average of a few points rather than integrating across the resulting output membership function to find the center of gravity.)

The two procedures are illustrated graphically below for the fuzzy set obtained from the previous example:

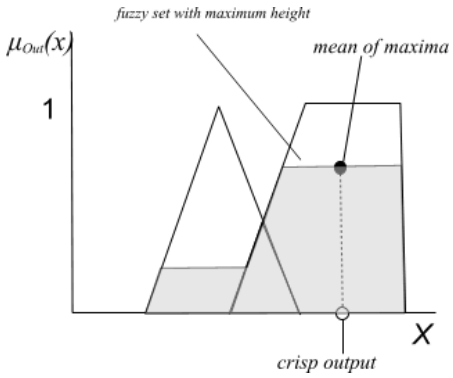


Figure 2.4: Illustration of mean-of-maxima method

Illustration of mean-of-maxima method in Fig. 2.4: this is the algorithm that we use in the experiment. The output is easily calculated by locating the "highest points" of the resulting output fuzzy set and finding their "middle point".

Illustration of center-of-gravity method in Fig. 2.5: The output is calculated by finding the center of gravity of the shaded area. This involves calculating two integrals, which makes the operation quite complex and not suitable for fast inference cycles.

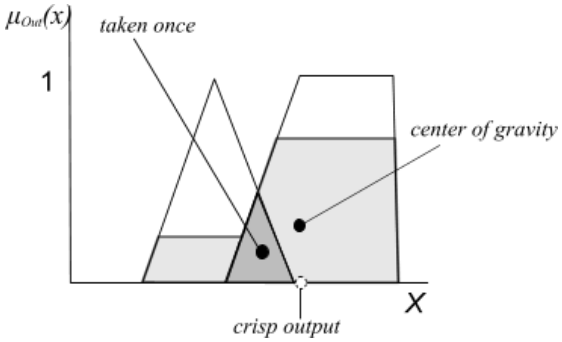


Figure 2.5: Illustration of center-of-gravity method

2.1.4 Example

The example in Fig. 2.6 demonstrates the steps that have to be taken. It describes a Fuzzy-Controller of an air conditioning system. The example has been taken from a documentation of P. Stegmaier. A more detailed example can be found in App. A.2.

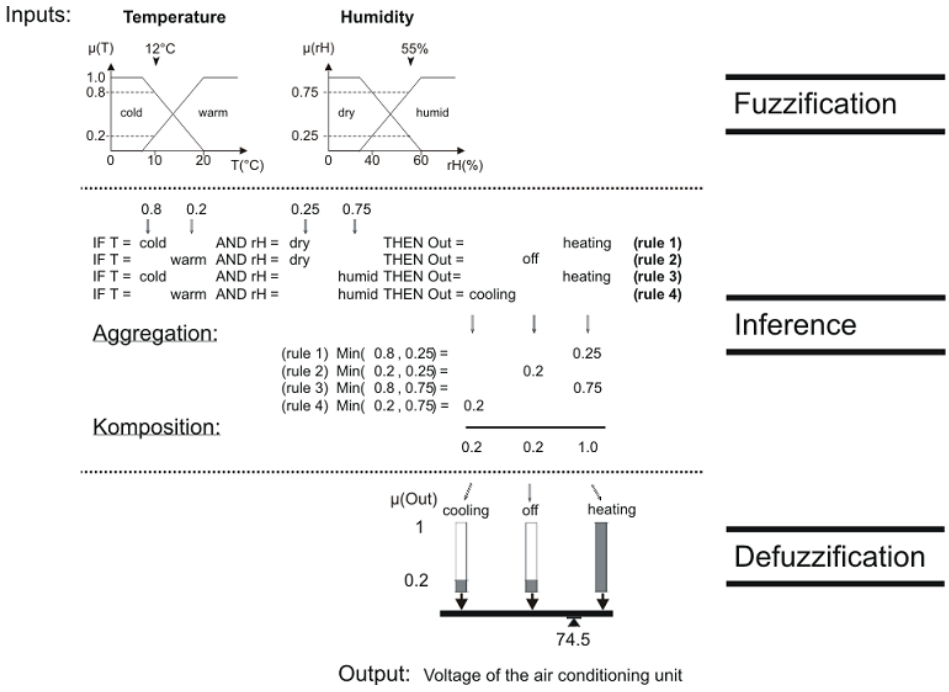


Figure 2.6: Fuzzy-Controller Example

2.2 The Helicopter-Model

The helicopter model consists of one crossbeam with two rotors and a vertical pole (Fig. 2.7). The crossbeam is mounted on top of the vertical pole in such a way that the main propeller (2) can move around a pivot point (4), which in turn can rotate around the base (7).

The model has two output signals y_Φ and y_Ψ (controlled variables). They are given as measurements (Voltages between -10 and $+10$V) of the two angular positions Φ and Ψ. Three

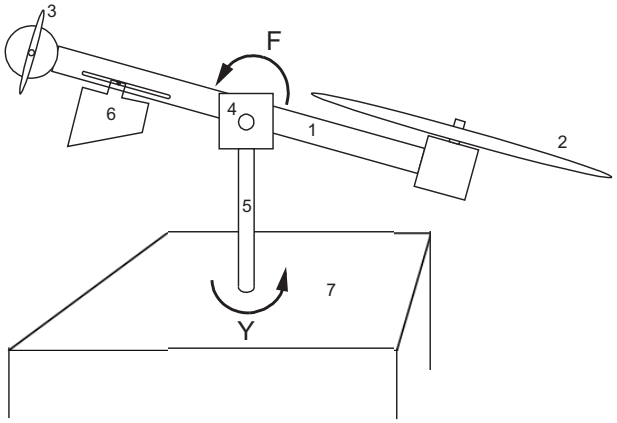


Figure 2.7: Model of the helicopter

typical measurement values are given in each of the following two tables:

vertical angle Φ		control variable y_Φ
$-57°$ (large rotor down)	$\Longrightarrow$	-10.0 Volt
$0°$ (crossbeam horizontal)	$\Longrightarrow$	0.0 Volt
$+57°$ (large rotor up)	$\Longrightarrow$	+9.4 Volt

horizontal angle Ψ		control variable y_Ψ
$-172°$ (beam at the left limit)	$\Longrightarrow$	-10.0 Volt
$0°$ (beam in central position)	$\Longrightarrow$	0.0 Volt
$+172°$ (beam at the right limit)	$\Longrightarrow$	+10.0 Volt

In a similar fashion, the helicopter model also has two input signals u_Φ and u_Ψ (manipulated variables) which control the the motors for the large and small rotors respectively. Both rotor motors are controlled by voltages ranging from -10 to +10V. The relationships between the input signals and the angular positions are as follows:

manipulated variable u_Φ:		influence on the angular position Φ:
positive	$\Longrightarrow$	positive torque (large rotor moves upward)
negative	$\Longrightarrow$	negative torque (large rotor moves downward)

manipulated variable u_Ψ:		influence on the angular position Ψ:
positive	$\Longrightarrow$	negative torque (beam moves clockwise)
negative	$\Longrightarrow$	positive torque (beam moves counterclockwise)

Due to physics of gyroscopes, the system is coupled, ie changes to the subsystems for Φ and Ψ effect each other. For example, if the large rotor moves quickly upwards, then the crossbeam is affected by a negative perturbation moment which causes the beam to turn clockwise in the horizontal plane. Such unwanted couplings are given in the following tables:

manipulated variable u_Φ:		coupling w.r.t. the angular position Ψ:
positive	$\Longrightarrow$	negative distortion (beam moves clockwise)
negative	$\Longrightarrow$	positive distortion (beam moves counterclockwise)

manipulated variable u_Ψ:		coupling w.r.t. the angular position Φ:
positive	$\Longrightarrow$	positive distortion (large rotor moves upward)
negative	$\Longrightarrow$	negative distortion (large rotor moves downward)

Luckily the above couplings are relatively weak, so it can be assumed that the MIMO System is effectively decoupled. Thus, the controllers will be designed for two independent SISO Systems. In that sense the couplings are modelled as disturbances that have to be rejected by the individual controllers.

For the stationary hovering position with the operating point $\Phi = 0°$ and $\Psi = 0°$ the following motor currents have been measured [1]:

$\boxed{\Phi_0 = 1.95 \text{ Volt}}$ $\boxed{\Psi_0 = \text{-}2.9 \text{ Volt}}$

These voltages have to be added to the output of the controller as so called offset-voltages for the manipulated variables (see Fig. 2.8). This means, for controller outputs of $U_{\Phi_{out}}$ and $U_{\Psi_{out}}$ of both 0 Volts, the rotors are not stopped but turn with the speed that is required to keep them in the operating point.

2.3 Structure of the control

Although the two movement directions Φ and Ψ affect each other, they are assumed to be decoupled and will be controlled separately. The two feedback loops are conventionally designed (compare Fig. 2.8) with the measured, A/D-converted control variables y_Φ (y_Ψ) subtracted from the reference value r_Φ (r_Ψ). The result is the control error e_Φ (e_Ψ) which is used as an input for the controller.

From the modelling of the system (compare with IfA Fachpraktikum experiment 2.6 (Helicopter II - Lead-Lag)[2].) it can be concluded that the time constant of the small rotor is negligible. Therefore the control of the Ψ-movement with a PD-controller is sufficient. The time constant of the large rotor is about ten times greater, which requires the controller to have an additional zero-point. Therefore the controller of Φ movement has a PD^2 structure. These principal considerations are also relevant for the fuzzy logic control. The consequence is that the first and second derivative of e_Φ and the first derivative of e_Ψ have to be taken as inputs.

The values of the input signals are multiplied with proper factors g_i, $i = 1..5$, to map the variables accurately to the domain of the membership function. Therefore the actual fuzzy controller systems has either two (e_Ψ and $\dot{e}_\Psi$), or three inputs (e_Φ, $\dot{e}_\Phi$ and $\ddot{e}_\Phi$) with one output. Each of the output signals are multiplied with appropriate factors h_i, $i = 1, 2$, with the offset values Φ_0 and Ψ_0 for the stationary flight added, before being D/A-converted and passed to the helicopter model as manipulated variables u_Φ and u_Ψ.

2.4 Homework

Before starting the practical experiments, it is necessary to complete the following example. You are requested to check your results with the solution-sheet during the lab session and correct them if necessary.

Task 1: Simplified Fuzzy Controller

> The example deals with the control of the Ψ axis. Your task is to compute the output of the Fuzzy controller for that axis, given measurements and membership functions for the inputs and outputs. Based on section 2.1 enter the missing values and thereby calculate the output of the fuzzy controller for time t_1.

[1] With the a distance of 9.3 cm between the weight on the beam and the vertical pole.
[2] `http://people.ee.ethz.ch/~ifa-fp/wikimedia/images/2/2d/IfA_2-6_manual.pdf`

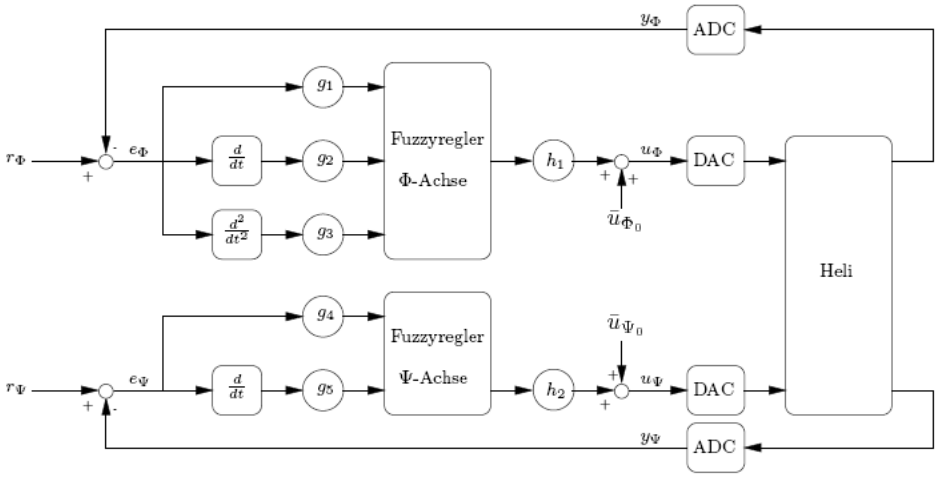


Figure 2.8: Structure of the control system (derivatives are discrete)

1. Two adjacent measurement values of y_Ψ are given:

$y_\Psi(t_1 - T_s)$	=	-4.7 Volt
$y_\Psi(t_1)$	=	-4.8 Volt

   In this case $T_s = 0.05\ s$ stands for the sampling time of the interval. The reference r_Ψ is given as $0\ Volt$.

2. Fuzzification:

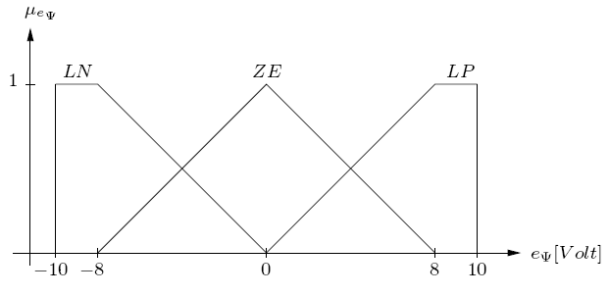


Figure 2.9: Fuzzification of the control error e_Ψ

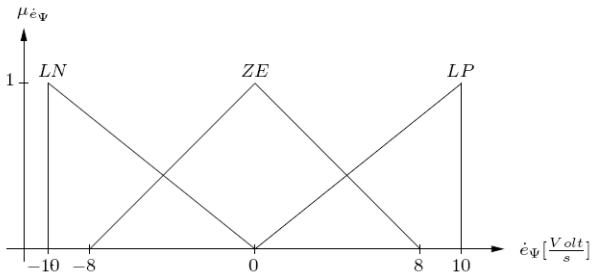


Figure 2.10: Fuzzification of the first derivative of the control error $\dot{e}_\Psi$

Control error	$e_\Psi(t_1) =$ $= \_\_\_\_$ Volt	$\mu_e('\text{LN}') = \_\_\_\_$ $\mu_e('\text{ZE}') = \_\_\_\_$ $\mu_e('\text{LP}') = \_\_\_\_$
control error derivative	$\dot{e}_\Psi(t_1) = \frac{e_\Psi(t_1) - e_\Psi(t_1 - T_s)}{Ts}$ $= \_\_\_\_$ Volt/s	$\mu_{\dot{e}}('\text{LN}') = \_\_\_\_$ $\mu_{\dot{e}}('\text{ZE}') = \_\_\_\_$ $\mu_{\dot{e}}('\text{LP}') = \_\_\_\_$

3. Rules:

1.	If (e_Ψ is LN) and ($\dot{e}_\Psi$ is none) then out_Ψ is LN	weight 1
2.	If (e_Ψ is LP) and ($\dot{e}_\Psi$ is none) then out_Ψ is LP	weight 0.5
3.	If (e_Ψ is LN) and ($\dot{e}_\Psi$ is LP) then out_Ψ is ZE	weight 0.5
4.	If (e_Ψ is LP) and ($\dot{e}_\Psi$ is LN) then out_Ψ is ZE	weight 0.5
5.	If (e_Ψ is ZE) and ($\dot{e}_\Psi$ is ZE) then out_Ψ is ZE	weight 0.5

Note: the *none* keyword is used when the state of an input is not relevant in a rule. It sets the degree of satisfaction for the corresponding input in the rule to 1.

4. Aggregation:

Rule 1:	min(____ , ____)	=	____
Rule 2:	min(____ , ____)	=	____
Rule 3:	min(____ , ____)	=	____
Rule 4:	min(____ , ____)	=	____
Rule 5:	min(____ , ____)	=	____

5. Composition:

$\mu_{out}('\text{LN}') = \_\_\_\_\_$
$\mu_{out}('\text{ZE}') = \_\_\_\_\_$
$\mu_{out}('\text{LP}') = \_\_\_\_\_$

6. Defuzzification:

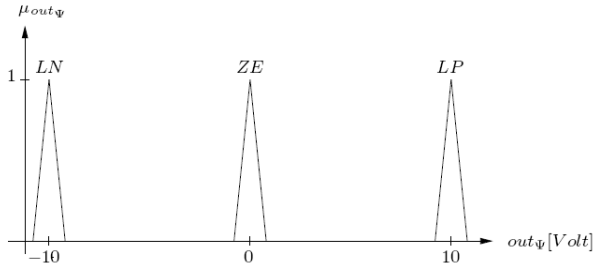


Figure 2.11: Defuzzification for out_Ψ

$out_\Psi = \_\_\_\_$ Volt

Chapter 3

Lab Session Tasks

To start the exercise, it is necessary to open the text file "1.4 Helicopter I" on the Desktop and follow instructions. The experiment files will be retrieved and will be stored in the following directory structure: **C:\Scratch\Fuzzy_Heli**.
Note: Sometimes it seems that the Φ controller is not working properly. A possible reason is that the manual control joystick generates a noisy signal. So, **make sure that manual control joystick is switched off**, when you are testing the fuzzy control structure.

Task 2: Introduction to the Fuzzy-Editor

Before creating a fuzzy controller, it is necessary to familiarize yourself with the fuzzy editor and its functions in App. A. Furthermore you should get to know the membership functions and the fuzzy rules of the Ψ-controller. The FuzzyPsi diagram should already be open, however, if it is not, type *fuzzy fuzzyPsi* to the Matlab command prompt. This will open the fuzzy editor with the Ψ-controller. The Ψ-controller has two inputs:

Control error	$e_\Psi(t_1)$	Epsi
Control error derivative	$\dot{e}_\Psi$	Epsid

Now, test all functions that were described in the previous sections. Before proceeding with the next exercise save the Ψ-controller to the Workspace (*File→Export→Save to workspace*, or press *Ctrl+t*) . Then open the Φ-controller (*fuzzy fuzzyPhi*) and also save it to the workspace.

Task 3: Control of the Ψ-axis

Your next task is to try to understand the functioning of a simple fuzzy controller. This is done by examining the fuzzy controller of the Ψ-axis. Next, go to the Simulink window, or in case it is not visible, open it by typing *Fuzzy_Heli _RTW* . This model contains the fuzzy controller and the interface to the helicopter implemented. The Simulink Model uses the Windows Real Time Target toolbox, which means that you have to compile the program first by pressing *Incremental Build* next to the *Start* button, or by using the shortcut *Ctrl+b* when inside the simulink window. Turn on the helicopter model. You may start the control of the helicopter by pressing first *connect to target* and then the *Start* button of Simulink.

The helicopter will now move to the operating point $\Phi = 0°$ and $\Psi = 0°$. This may be monitored on the three scopes on the PC screen: The first scope shows the Ψ-axis

with the reference and the control variable. The second one shows the Φ-axis. The third scope keeps track of the two manipulated variables. The noisy manipulated variable in that scope is u_Φ. Now you may add step changes on the reference value of the Ψ-controller, by changing *Psi-ref* on the left side. The changed reference value is sent automatically to the helicopter. Next, enter step changes from -8 to 8 and monitor the behavior of the controller. You can stop the controller with the *Stop* button of Simulink. If you don't do that it can happen that the motors get full power until you are done with entering the value. So please avoid that.

We want to investigate the influence of each fuzzy rule to the system. In the editor turn off all controller rules by setting their weight to 0 and then turn on one after another. Unfortunately these changes do not become active automatically. After each change you will have to save the fuzzy controller to the workspace and start the Simulink model afterwards! After each change monitor the behavior of the controller in the scopes. Try to interpret the effects and reasons of each of the rules. Afterwards complete the following table. You may use the explanations from previous sections and Fig. 2.8 for your argumentation:

Nr.	e_Ψ	$\dot{e}_\Psi$	out_Ψ	observed angular position Ψ		manipulated variable u_Ψ
1	LN	ANY	LN	too large	⇒	strong negative
2	LP	ANY	LP	.....	⇒	.....
3	ANY	LP	LP	.....	⇒	.....
4	ANY	LN	LN	.....	⇒	.....
5	ZE	ZE	ZE	.....	⇒	.....

Task 4: The usage of a *none*-term in a rule

You may use *none*, if the state of an input is not relevant in a rule. *None* will set the degree of satisfaction for the corresponding input in this rule to be 1. In this exercise it should be attempted to replace *none* by other terms:

In the fuzzy controller of the Ψ-axis replace the first rule (LP none ⇒ LP) with the following three rules:

- LP LN ⇒ LP
- LP ZE ⇒ LP
- LP LP ⇒ LP

Again, enter large step changes of the reference value (from −8 V to 8 V and the other way around) of the Ψ-controller. Afterwards answer the following Questions:

- Do the three substituted rules have the same effect as the original rule? Reasons? .....
  .....
  .....

- Do you notice differences in the controller behavior? Which? .....
  .....
  .....

- Which version would you prefer? Why? .....
  .....
  .....

After answering these questions please revoke the changes that you made on the Ψ-controller in this exercise. Namely, replace the three new fuzzy rules by the original rule: 'LP none ⇒ LP'.

Task 5: Control of the Φ-position

Next you should learn about a fuzzy controller for the Φ-position. If you do not have the Φ-controller open any longer please open it again by entering *fuzzy fuzzyPhi*. The Φ-controller has the following three inputs:

Control error	e_Φ	Ephi
First Derivative (discrete) of the control error	$\dot{e}_\Phi$	Ephid
Second derivative (discrete) of the control error	$\ddot{e}_\Phi$	Ephidd

Start as in task 3 and fill out the following table. Note: make sure that the relevant rule is activated with the current initial position of the helicopter. You can move the helicopter manually using the joystick next to the power switch.

Nr.	e_Φ	$\dot{e}_\Phi$	$\ddot{e}_\Phi$	out_Φ	angular position Φ		manipulated variable u_Φ
1	LN	ANY	ANY	LN	too large	⇒	strong negative
2	LP	ANY	ANY	LP	.....	⇒	.....
3	ANY	LN	ANY	LN	.....	⇒	.....
4	ANY	LP	ANY	LP	.....	⇒	.....
5	ANY	ANY	LN	LN	.....	⇒	.....
6	ANY	ANY	LP	LP	.....	⇒	.....
7	ZE	ZE	ZE	ZE	.....	⇒	.....

Task 6: Weighting of rules

The fuzzy editor allows to assign individual *Weights* to each rule. Thereby some rules can be set to be more important than others. We will use this fact in the following.

In a simple Φ-controller an overshoot can be seen if one tries to reach the operating point from below. If the helicopter model would ascend slower then this overshoot could be avoided.

- Which rule slows down the helicopter if the beam ascends at high speed?
- Increase the weight of this rule, relative to the other rules, by 10 percent. What are the effects of this increase on the controller behavior?
- What effect has a doubling of the weight on this rule?
- What rule dampens the change of the motor voltage? Double the weight on this rule.

After changing the rules in the FuzzyEditor you have to export it to the workspace and stop and start the simulink model to make the changes active.

Task 7: Position of the membership functions over the inputs

Once more we take a look at the Ψ-controller in this exercise. The model is supposed to be steered to a position at a clockwise offset of 68°. However no reference or offset values shall be changed. The new position shall be reached by only shifting the membership functions of Epsi! When changing the shape or location of the membership functions you need to export the new definition by saving the fuzzy controller structure to the workspace (*Ctrl+t*) and then rebuilding the simulink model (*ctrl+b* from the simulink window).

- What is the controller voltage y_Ψ, if the model is in the position $\Psi = 68°$? (Use the corresponding table in Sec. 2.2)
- Where should the maximum of the 'ZE'- membership function of Epsi be placed?
- The 'ZE'- membership function shall be moved accordingly. In addition the functions 'LN' and 'LP' have to be changed in such a way that they once again touch at the center of the 'ZE'-function. What is the expected voltage difference $r_\Psi - y_\Psi$ that you will get with this controller?

Next move the membership function back to its original position, so that it is again centered on 0V. Shape the functions for 'LN' and 'LP' accordingly.

Task 8: Shape of membership functions over the inputs

The steepness of a face of the membership function influences the behavior of the fuzzy controller. Steep faces result in step-like changes. Such changes are often unwanted. Frequently the ideal steepness can only be determined experimentally. For the inputs, gaps between their functions should be avoided.

The next exercises shall be solved for a poor regulator for the Ψ-position. Therefore, in the Ψ-controller, move the root points of the 'ZE' function of Epsi to $\pm 2\ V$ and the root point of 'LN' to $-2\ V$ and 'LP' to $2\ V$ respectively. Also here, don't forget to save the new definitions to the workspace and rebuild the simulink model.

- Test the behavior of this controller by applying stepwise changes to the reference value. What are the weaknesses of this controller?
- Did the controller improve by moving the root points of the 'LN'- and 'LP'-function of Epsi from $\pm 2\ V$ to $0\ V$?
- Does the controller improve further, when moving the root points of 'ZE' to the boundary ($\pm 10\ V$)? Why?

Undo also these changes. Move the root points of 'ZE' back to $\pm 4\ V$. The root points of 'LN' and 'LP' should be on $0\ V$.

Task 9: Membership functions over the output

In this exercise the membership functions of the output are supposed to be changed. Provided the given defuzzification method (*mean-of-maxima*) the shapes of the functions do not have a big influence. Thus only the positions of the functions are changed. Nevertheless such an approach can be quite complex and unpredictable as it influences the results of multiple rules. You will notice this complexity when moving the functions for the output of the Φ-controller.

- What bothers you as the Φ-controller moves the helicopter from the lowest position to the operating point?
- How can you eliminate this behavior by moving one of the output membership functions?
- Which problem can be introduced by changing the membership function? Please explain!

Task 10: Development of a better Ψ-controller

Provided you have carefully solved the tasks 2 to 9, you should now be able to design a fuzzy controller for the Ψ position on your own.

You should start using the Ψ-controller that you worked with in the previous exercises. However that controller has the disadvantage that it does not react equally to positive and negative control errors. This becomes especially obvious when large stepwise changes of the reference trajectory are made. This behavior originates from the addition of an offset voltage to the output that compensates for the distortion moment of the large rotor.

Think carefully about the possibilities to resolve this problem. Experiment with different approaches until you have found an effective solution.

Task 11: Develop, implement and optimize a new Φ-controller.

Now you may start designing the Φ-controller. Typing *fuzzy* will open the FIS editor. There you should save your Φ controller to *fuzzyPhi*. For the beginning it is recommended to use only few membership functions. Also restrict yourself to as few fuzzy rules as needed. Your initial fuzzy controller should be as simple as possible, otherwise you will quickly be daunted by the complexity.

You are asked to optimize the controller so that the helicopter model reaches the horizontal position from below quickly and with little overshoot. In this position the flight should be smooth and without any control error.

Far more complex is the controller optimization in the case where the helicopter starts from the upper resting position. Between the upper resting and the horizontal position lies the bifurcation point of the helicopter. In that point the position of the cross beam is unstable. If the model is moved from the upper resting position downward, then the rotation direction of the main rotor should be changed in this point ($\Phi_{bif} = 34°$), at the latest! This has to happen very quickly so the helicopter will not "crash" (hit the lower stopping position). Notice that the moment of inertia of the large rotor is rather big, much bigger than for the small rotor.

With the previously optimized Φ-controller the model is likely to hit the lower stopping position before levelling up to the operating point. Thus you are requested to avoid this disastrous behavior by adding additional rules.

Try some experiments, beginning from the upper left or upper right resting position. Be aware that the movement of the small rotor influences your Φ-control significantly. Don't spend too much time on this task. Just try a few things and see if you can improve the behavior.

Attention! Under all circumstances avoid hitting the lower stopping point!

When you are done please turn off the model helicopter hardware (behind the helicopter).

Chapter 4

Lessons Learned

This chapter shortly summarizes the most important facts from this experiment and gives a short list of questions. Please answer these and discuss them with an assistant.

Lesson Learned 1: Fuzzy concept

Why is the idea of fuzzy control appealing?

Lesson Learned 2: Limitations

What are the limitations and drawbacks of fuzzy control?

Lesson Learned 3: Defuzzyfication

You know now two defuzzyfication methods. The mean-of-maxima method and the center-of-gravity method. Explain shortly the difference of these two methods and think of case where this could be relevant.

Lesson Learned 4: Completion of Experiment

Please, fill out the online feedback form on the registration page under `MyExperiments`. Each student/participant has to fill out its own feedback form. This will help us to improve the experiment. Thank you for your help.

Appendix A

Details

A.1 Description of the Fuzzy-Editor

To enable fuzzy control, membership functions have to be defined to fuzzify and defuzzify every input and output signal. Furthermore rules have to be defined that govern how the degrees of satisfaction of the input signals are mapped to the output values, a step that is called aggregation.

Both steps are easily done with the FIS (*Fuzzy Inference System*) Editor in Matlab. The editor is started by typing *fuzzy Filename.fis* to the command prompt in Matlab[1]. In the upper part of the editor (Fig. A.1) the fuzzy controller with its multiple inputs and outputs is shown. By double clicking on the input and output blocks, an additional membership function editor opens. The fuzzy rule editor can be opened by double clicking on the center box. Whenever changes are made, the parameters must be saved to the workspace - using File→Export→To Workspace (or CTRL + T). If you only change the weighting of the rules, it is enough to stop and restart the simulink model to apply the changes. However, if you make changes to membership function shapes and locations, you will also need to rebuild the simulink model (CTRL + B from simulink window).

On the left the two inputs e_Ψ and $\dot{e}_\Psi$ are shown together with their corresponding membership function(fuzzification). In the middle is the block that includes the fuzzy rules (inference) which is connected to the output of the fuzzy controller shown together with its membership function(defuzzification). Moreover, the FIS-Editor has some selection lists (bottom left). Note, that the default valuesof these settings are correct.

A.1.1 The Membership Function Editor

As the name membership function editor indicates, it is used for definition, modification and removal of membership functions as seen in Fig. A.2.

On the left side all inputs and outputs are shown (*FIS Variables*). By clicking on it, an input or output is selected. The selected input or output is shown on the right side *Membership function plot* as a diagram with the membership functions that is relating to it. By clicking on a membership function this function turns red. It may then be changed as follows:

- With the mouse: The whole membership function can be moved by clicking with the left button on it and keeping the button pressed, while moving the function. The shape of the membership function can also be changed. Therefore one of the small squares that show up on the function, has to be moved.

- Manually: The name of the membership function can be changed in the lower right area of the window (*Name*). To increase the readability of your fuzzy controller please stick to the following standard terms when naming membership functions:

[1] e.g. *fuzzy fuzzyPhi* and *fuzzy fuzzyPsi*.

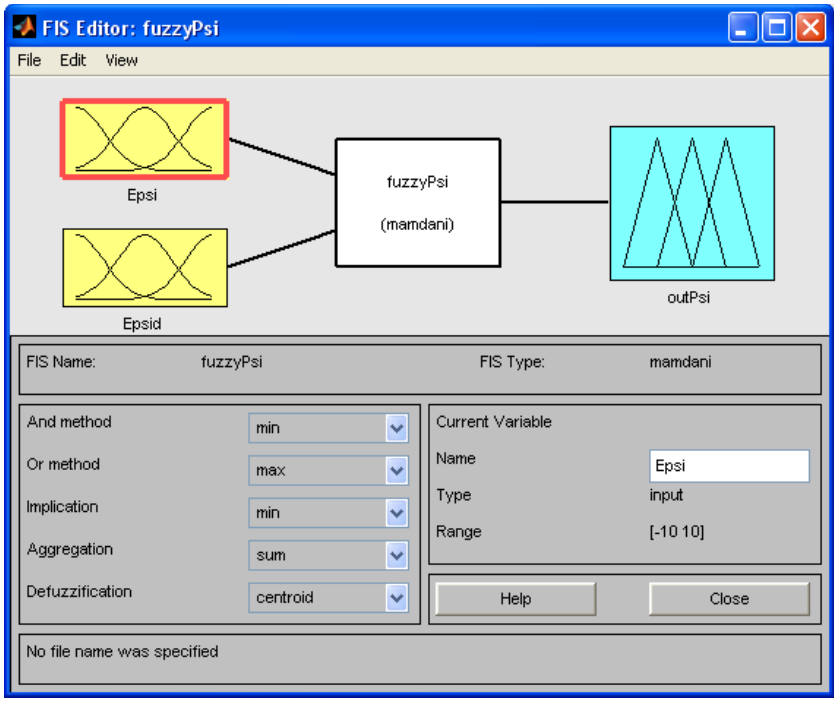


Figure A.1: FIS-Editor for an example of a fuzzy controller of the Ψ-Axis.

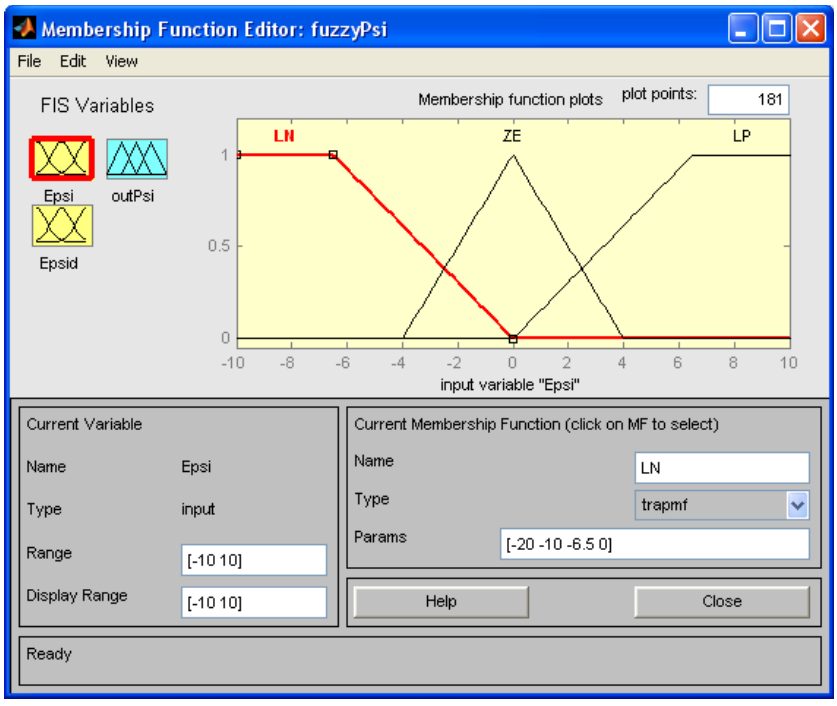


Figure A.2: The membership function editor for the e_Ψ input variable.

LN	Large Negative
MN	Medium Negative
SN	Small Negative
ZE	Zero
SP	Small Positive
MP	Medium Positive
LP	Large Positive

Below thename field, the type of a membership function can be determined. Commonly used types are *gaussmf*, *gauss2mf*, *pimf*, and *trimf* (*mf* stands for membership function). The shape of a membership function can not only be controlled with the mouse, but also in the field Parameters (*Params*). Inside this field the x-axis values of the small squares are entered. The positions of the squares define the shape of the membership function.

Membership functions are added and removed via the menu option *Edit*.

The range of input values for the membership functions is defined in the lower left corner (*Range*). Also the displayed region of the diagram is defined there (*Display Range*).

A.1.2 The Fuzzy Rules Editor

The Rules editor can be accessed by clicking on the menu *Edit→Rules*. An example is shown in Fig. A.3 for the Ψ axis case. It contains the numbered fuzzy rules in the upper text field. These rules define the inference step and stick to the scheme *If...and/or...then....*

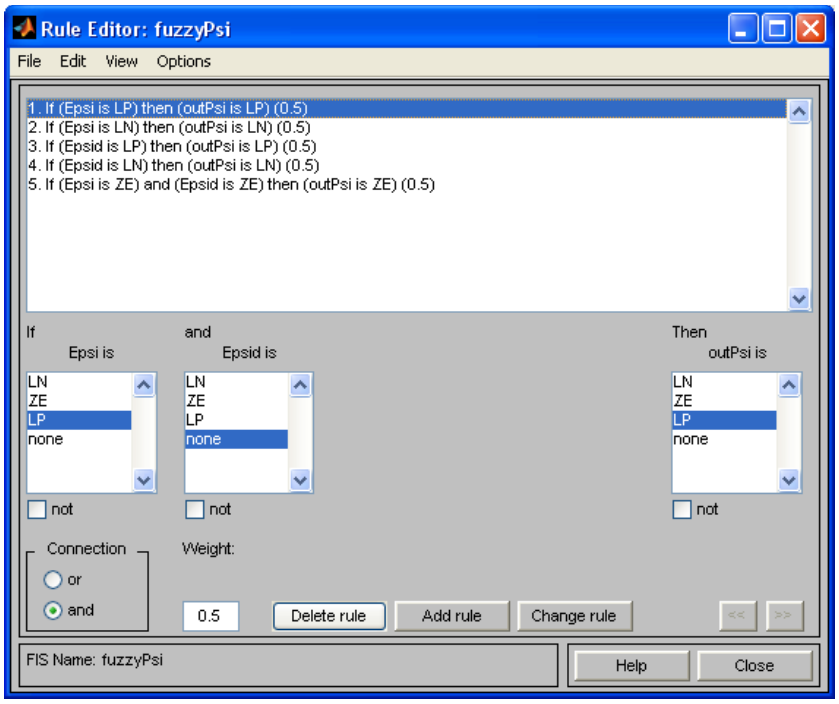


Figure A.3: The Fuzzy Rule Editor: Example for a fuzzy controller for the Ψ-axis.

- Rules can be edited by selecting them in the upper text field. The input and output terms that are connected by a rule are marked in the windows below. A term can be negated by pressing the corresponding *not*-button. If you don't want an input to be used in a rule, select the *none* option for this input. The type of interconnection between the inputs (*and*/*or*) is set with *Connection* in the lower left of the window. Notice: after these modifications they have to be activated. This is done by clicking the button *Change rule*.

- Every single rule has a weight that is set with respect to the weights of the other rules. This weight corresponds to the influence of this rule on the output after defuzzification. The weights of the rules are noted in parentheses, behind each rule in the upper window. The weights are normalised to a maximum of 1, i.e. they can take values between 0 and 1. After selecting a rule, the weight of a rule can be changed by changing *Weight* in the lower part of the window. For the change to become effective, the button *Change rule* has to be pushed afterwards.
- New rules may be added by pressing *Add rule*. Thereby the rule that is currently marked is added to the end of the list.
- *Delete rule* deletes the rule that is currently marked.

A.1.3 Remarks Concerning the Filesystem

Fuzzy controllers are saved as *datei.fis* on the hard disk (*File→Save to disk*). Before you simulate the closed-loop system, the fuzzy controller has to be saved to the workspace (*File→Export→Save to workspace* or *CTRL+T*), otherwise the fuzzy controller in Simulink cannot access the (updated) membership functions and fuzzy rules.

A.2 Numerical Example - Fuzzy Controller for a Heating and Air Conditioning (HAC) System

A.2.1 Fuzzy Control Problem Specification

The working principle and all calculations involved are described here in more detail for a system with two inputs and one output. The system in consideration is a heating and air conditioning (HAC) system for a room in a building. We have two inputs at hand, which are the room temperature T_i and the outside or ambient temperature T_o. Sensors provide us with the crisp values of these two temperatures and using a predefined set of rules a crisp output out_{HAC} for our HAC system has to be found. It is assumed that an input equal to zero means do nothing or shut off the HAC. A negative input means cooling and a positive input means heating. For simplicity, we define the following range for our control output $u \in [-1, 1]$. The fuzzy controller completes three steps at each iteration:

1. **Fuzzification:** Find the levels of the membership functions of the inputs for the given crisp values of the sensors
2. **Inference:** Apply the rules to the fuzzy input values, called aggregation, and if multiple rules for the same output membership function exist add them together according to their weight, called composition.
3. **Defuzzification:** Calculate the crisp output value from the aggregated membership functions of the inputs.

A.2.2 Fuzzy Controller Synthesis

The controller synthesis includes the choice of membership functions for all inputs and outputs plus the definition of the fuzzy rule set with their weights. Since this should be a small working example we keep the complexity low even though this does not lead to a satisfactory result. The emphasis is on the calculation only. We choose the following fuzzy sets with their according membership functions:

1. **Input membership functions:** We define a fuzzy set for the room temperature containing three values, cold, comfortable, and hot. For the ambient temperature we define a fuzzy set with two values, cold and hot. A graphical representation of the corresponding membership functions of the two fuzzy sets can be seen in Fig. A.4.

2. **Output membership functions:** We define a fuzzy set for the control output, which is fed to the HAC, containing three values, cooling, off, and heating. A graphical representation of the corresponding membership functions of the three fuzzy sets can be seen in Fig. A.5.

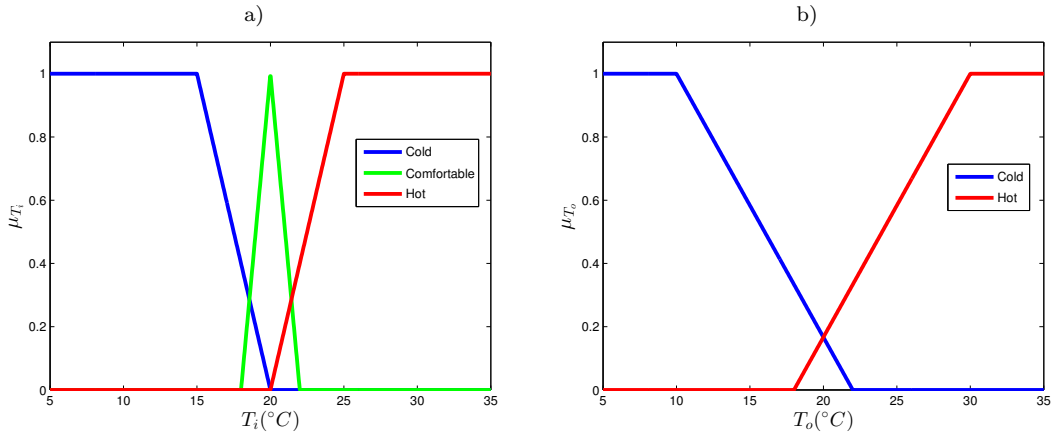


Figure A.4: Membership functions for the system inputs where Fig. a) is for the room temperature and Fig. b) for the ambient temperature

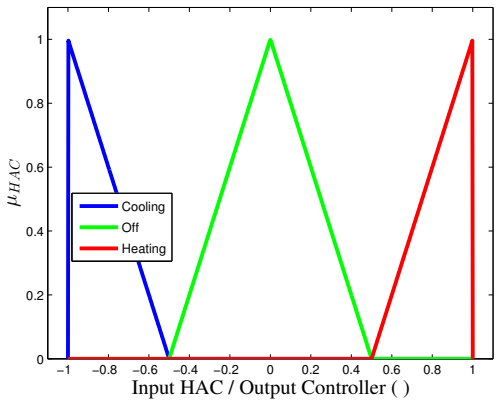


Figure A.5: Membership functions for the system's output.

We defined the set of rules for our inference step in Tab. A.1. All rules have weight 1 except for rule 4 and 5 which have a weight of 0.25. For the defuzzification we will use the *center-of-gravity* method thus taking a compromise among different solutions. Now we have defined all controller specification and we can calculate one iteration of the controller.

Table A.1: Set of Rules for our Fuzzy HAC Controller

R1:	**IF**	T_i is cold	**AND**	T_o is cold	**THEN**	the output is heating
R2:	**IF**	T_i is hot	**AND**	T_o is hot	**THEN**	the output is cooling
R3:	**IF**	T_i is comfortable	**AND**	T_o is none	**THEN**	the output is off
R4:	**IF**	T_i is cold	**AND**	T_o is hot	**THEN**	the output is off
R5:	**IF**	T_i is hot	**AND**	T_o is cold	**THEN**	the output is off

A.2.3 One Fuzzy Controller Iteration

We get the following temperature sensor readings

$$T_i = 17°C$$
$$T_o = 7°C$$

Fuzzification We now compute the degree of membership of the input values in their according fuzzy sets. This is done graphically, see Fig. A.6. Note that we get a degree of membership for all membership function within a fuzzy set for a single input. Using the graphical representation

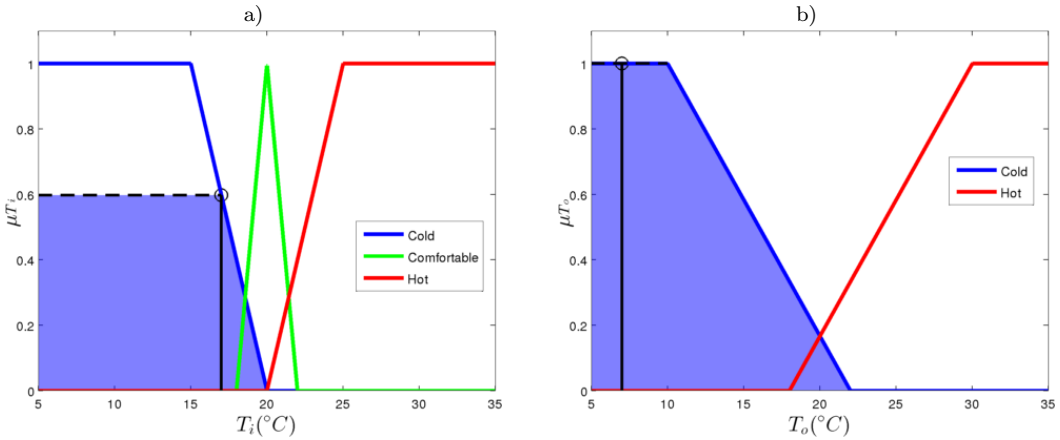


Figure A.6: Membership functions for the system inputs with the degree of membership for the given temperatures

in Fig. A.6 we get the following membership degrees for the room temperature:

$$\mu_{T_i}^{cold}(17) = 0.6$$
$$\mu_{T_i}^{comf}(17) = 0$$
$$\mu_{T_i}^{hot}(17) = 0$$

and the ambient temperature:

$$\mu_{T_o}^{cold}(7) = 1$$
$$\mu_{T_o}^{hot}(7) = 0$$

If the value of T_i *would be* 19 *instead of* 17*, we would get* $\mu_{T_i}^{cold}(19) = 0.2$ *and* $\mu_{T_i}^{comf}(19) = 0.5$

Inference Now we have to apply the rule set to our fuzzy input values, also called *Firing of rules* or aggregation. The '**AND**' statement in the rules translates to a *min* operation and the '*none*' statement means that we do not care about this value and thus it is set to 1 in a '**AND**' statement:

$$\begin{aligned}
R1 : \mu_{out}^{heating} &= \min\left(\mu_{T_i}^{cold}(17), \mu_{T_o}^{cold}(7)\right) &&= \min(0.6, 1) = 0.6 \\
R2 : \mu_{out}^{cooling} &= \min\left(\mu_{T_i}^{hot}(17), \mu_{T_o}^{hot}(7)\right) &&= \min(0, 0) = 0 \\
R3 : \mu_{out}^{off} &= \min\left(\mu_{T_i}^{comfi}(17), \mu_{T_o} none\right) &&= \min(0, 1) = 0 \\
R4 : \mu_{out}^{off} &= \min\left(\mu_{T_i}^{cold}(17), \mu_{T_o}^{hot}(7)\right) &&= \min(0.6, 0) = 0.6 \\
R5 : \mu_{out}^{off} &= \min\left(\mu_{T_i}^{hot}(17), \mu_{T_o}^{cold}(7)\right) &&= \min(0, 1) = 0
\end{aligned}$$

Since we have multiple values for the same output membership function and given weights w, we need to do a composition step:

$$\begin{aligned}
\mu_{out}^{cooling} &= \mu_{out}^{cooling} w^{cooling} && = 0 \\
\mu_{out}^{off} &= \mu_{out,R3}^{off} w_{R3}^{off} + \mu_{out,R4}^{off} w_{R4}^{off} + \mu_{out,R5}^{off} w_{R5}^{off} && = 0 \cdot 1 + 0 \cdot 0.25 + 0 \cdot 0.25 = 0 \\
\mu_{out}^{heating} &= \mu_{out}^{heating} w^{heating} && = 0.6
\end{aligned}$$

Defuzzification The last part is now to get a crisp output from the fuzzy output values. Since our membership function are not overlapping we can calculate the center of gravity for each membership function according to its degree and then take a weighted sum to get the crisp value. The center of gravity for the *heating* membership function can be seen in Fig. A.7. This method gives us the following output for the HAC system:

$$\begin{aligned}
out_{HAC} &= \frac{CG^{cooling} \cdot A^{cooling} + CG^{off} \cdot A^{off} + CG^{heating} \cdot A^{heating}}{A^{cooling} + A^{off} + A^{heating}} \\
&= \frac{-0.75 \cdot 0 + 0 \cdot 0 + 0.81 \cdot 0.21}{0 + 0 + 0.21} \\
&= 0.81
\end{aligned}$$

where CG stands for the x position of the center of gravity and A is the shaded area of each membership function.

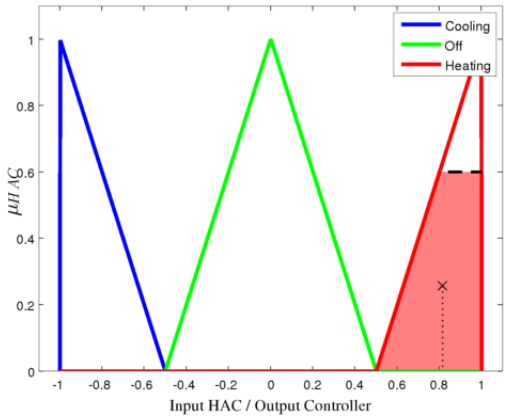


Figure A.7: Membership functions with corresponding degree and center of gravity of the shaded area, marked with a x.