CODEN: LUTFD2/(TFRT-5450)/1–54/(1991)

Application of Fuzzy Logic to Automatic Control: A Case Study of the Inverted Pendulum Problem

Helge Sturesson

Department of Automatic Control
Lund Institute of Technology
December 1991

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Master Thesis

Date of issue
December 1991

Document Number
CODEN: LUTFD2/(TFRT-5450)/1–54/(1991)

Author(s)
Helge Sturesson

Supervisor
H.Sugaya and G. Maier, ABB, K J Åström, LTH

Sponsoring organisation

Title and subtitle
Application of Fuzzy Logic to Automatic Control: A Case Study of the Inverted Pendulum Problem.

Abstract

This report refers to a master thesis project carried out at ABB Corporate Research, at the department of Computer Science. The purpose of the project was to investigate the possibility of applying fuzzy logic to the control of an inverted pendulum problem.

Two different techniques to design a fuzzy controller hasve been investigate and tested both on simulation and on a real equipment. The first approach is based on an inference method called *fuzzy relational composition.* Using this inference method may involve that the rulebase gets very large for processes having more than 2 states. In this report this problem was handled by decomposing the control task into different subtasks. In the second approach, called *weighted linear curve-fitting*, the idea is to model an expert's control action, in this case a digital controller, as well as possible.

The experimental results show that the fuzzy controller could be designed to perform competitively to a digital controller. However, this was achieved after spending a considerable amount of working hours in tuning the fuzzy controller.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
54

Recipient's notes

Security classification

1. Introduction

The goal of this project was to investigate the possibility to apply fuzzy logic to the control of an inverted pendulum problem. This process is both unstable and non-linear and these features make it an inviting candidate when investigating a new control strategy.

A brief overview of fuzzy logic history and its theory with an emphasis on applications to automatic control are given in Chapter 2. A software development tool called TIL shell was used to generate the fuzzy controller. This tool together with additional software, such as simulation environment and IO-drivers, is shortly described in Chapter 3. Two different techniques to design a fuzzy controller have been investigated in this report. First, a fuzzy controller is designed based on an inference method called *fuzzy relational composition* [7]. This is the traditional inference method for realizing fuzzy controllers and is described in Chapter 4. One significant drawback of this method, however, is that, in general, many partitions are required for each state variable. This means that the number of rules grows exponentially as the number of the states for the process increases. This was a main problem with the design of a fuzzy controller for the inverted pendulum. It was solved by decomposing the task into three subtasks. Another method of developing a fuzzy controller called *weighted linear curve-fitting* is described in Chapter 5. The idea behind this approach is to develop the control law by fitting the properties of a good controller or an experts behavior. In this report the fuzzy controller was fitted simply to an existing digital controller. Conclusions on fuzzy logic controllers being implemented by these two design methods are discussed in Chapter 6.

Due to the limited time of the project some issues remain undone. The original goal of the project was to achieve a detailed comparative study between a fuzzy and a digital controller in terms of robustness and designability. It became evident, however, that the design of a fuzzy controller was not as simple as expected. In addition, the software became quite extensive in both lines and time.

Finally, I want to thank my advisors, Dr. Georg Maier, Dr Hiro Sugaya and Prof. Karl-Johan Åström for their great support throughout this project. In particular I am deeply indebted to Hiro for his never ending encouragement and investment in this work. I also want to thank the staff at the department of Computer Science at ABB, for providing a very stimulating working environment and many useful advises.

2. Fuzzy Logic

This chapter gives an introduction to fuzzy logic. In the first section comes a short history of fuzzy logic and its application to process control. In section 2.2 the fuzzy logic theory is shortly described and in the last section the different elements of a fuzzy controller are explained.

2.1 Brief history

The starting point for Fuzzy logic was in 1965, when Lotfi A. Zadeh published his theory on fuzzy subsets [14]. In contrast to crisp subsets (binary logic), where objects can only be assigned the value of either 0 or 1, the fuzzy subset allows them to have all values from 0 to 1.
The research in finding applications to this theory has so far mainly been undertaken within the field of automatic control .

One of the pioneers in this field was Mamdani. In 1975 he presented, together with Assilian, the first fuzzy logic controller [8]. The process was a small boiler steam engine. The boiler pressure was controlled by the heat input to the boiler and the engine speed was controlled by adjusting a throttle at the input of the engine cylinder. Another early, but significant, work is the application of a fuzzy controller in a warm water plant, by Kickert and Lemke in 1976 [5].

In spite of many promising research results in the 70's, fuzzy control found no practical industrial applications until the Danish company F.L. Smidth developed their fuzzy controller for a cement kiln in 1980 [3]. Due to the great complexity of this process and the bad quality of available measurements, existing controllers on the cement kiln have shown a very poor performance. The performance of the fuzzy controller proved to be superior to all attempts done using traditional technique. Currently, Japan is the leading nation in research and industrial applications of fuzzy control. Applications range from camera auto-focusing to control systems of subways in Tokyo.

Advocates of fuzzy control claim that they are easier to implement than classical controllers. Less theory is required than for the design of a conventional controller. An objection to fuzzy control is that it is hard to analyze the behavior of the controller. Another objection is that so far no good comparative study has been done between fuzzy and conventional controllers [17].

2.2 Fuzzy Logic theory

2.2.1 Fuzzy subsets

To introduce fuzzy subsets let us first look at crisp subsets as used in binary logic. A crisp subset A is defined by a membership function $\mu_A(x)$ that is either 0 or 1.

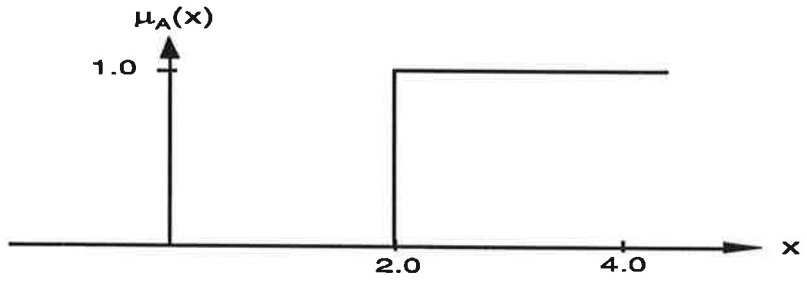


Figure 2.1. Membership function for a crisp subset A.

A fuzzy subset B, on the other hand, is defined by a membership function $\mu_B(x)$ that can have all values from 0 to 1.

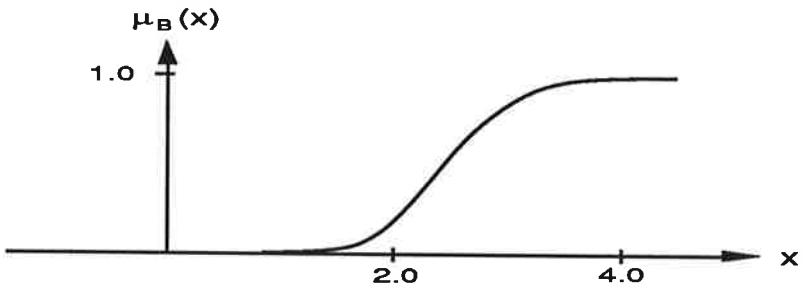


Figure 2.2. Membership function for a fuzzy subset B.

The usefulness of introducing this new type of subset is easiest explained by an example:

Suppose a regular computer program using binary logic has been assigned to distinguish a person's age between young and old. A decision must be made about where to place a borderline between young and old. For example as in Figure 2.3 at the age of 40. This implies that a person of age 39 is regarded as young, whereas a person of age 41 is old. Is this how a human being would reason? Surely not. His answer would

probably be that neither of them is completely young nor old. They are something in between, both young and old but to different degrees.

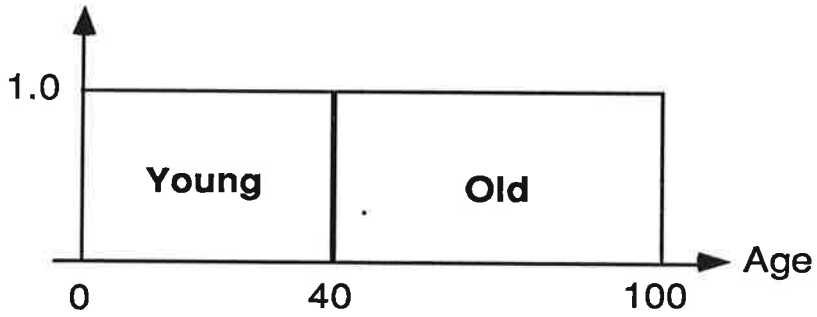


Figure 2.3. Solution of the problem above using two crips subsets Young and Old.

To solve this seemingly simple problem satisfactorily with binary logic is difficult, because we can not agree on a clear-cut answer to this question.

By using fuzzy logic, instead, the above example could be solved in a different way. The two subsets Young and Old can, for instance, be represented with the following membership functions as shown in Figure 2.4.

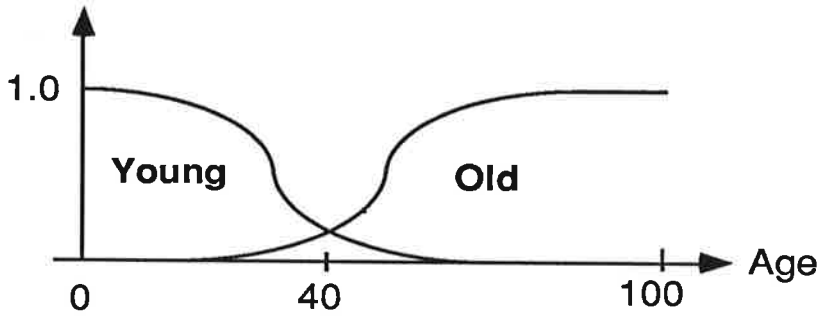


Figure 2.4. Solution of the problem above using two fuzzy subsets Young and Old.

We favour this solution over binary logic, because it corresponds more to the human way of reasoning.

Worth to note here is that the fuzzy subsets should not be interpreted as probality functions, for the sum of the membership functions do not have to be equal to 1.

2.2.2 Operations on fuzzy subsets

The three operations on fuzzy subsets will be descibed below, being

defined on the fuzzy subsets A and B with membership functions $\mu_A(x)$ and $\mu_B(x)$.

1. Intersection: (AND) Denoted A×B or AB

$$\mu_{AB}(x)=\min(\mu_A(x),\mu_B(x))$$

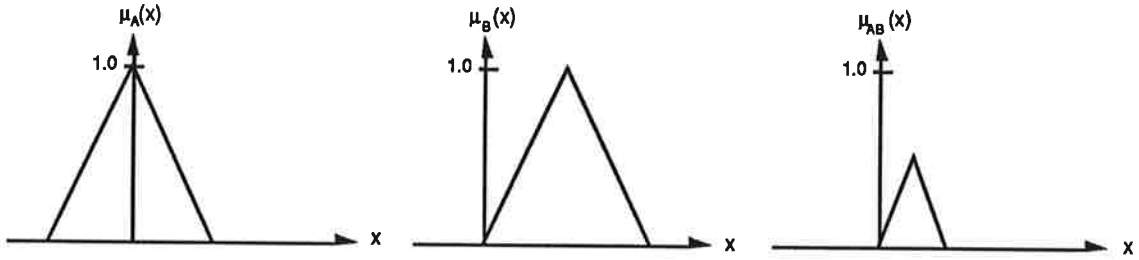


Figure 2.5. AND operation on two fuzzy subsets A and B.

2. Union: (OR) Denoted A∘B or A+B

$$\mu_{A+B}(x)=\max(\mu_A(x),\mu_B(x))$$

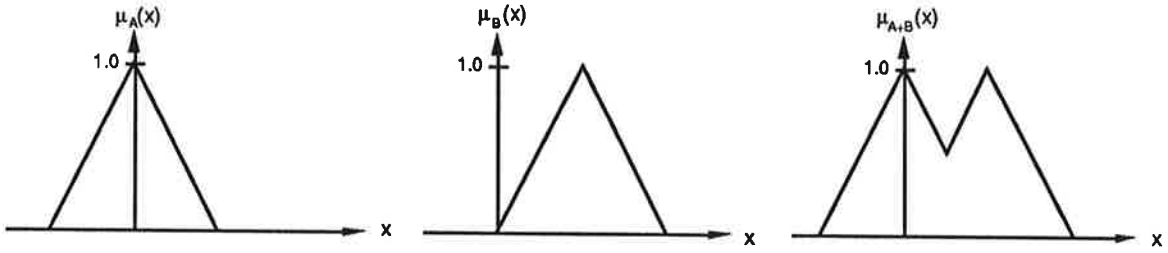


Figure 2.6. OR operation on two fuzzy subsets A and B.

3. Complement: (NOT) Denoted $\hat{A}$

$$\mu_{\hat{A}}(x)=1-\mu_A(x)$$

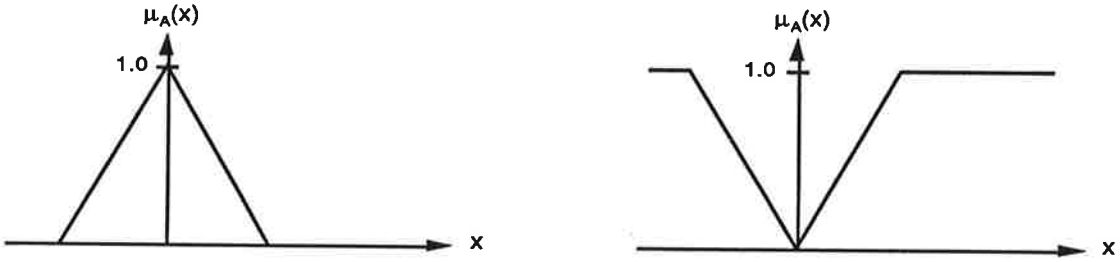


Figure 2.7. NOT operation on a fuzzy subset A.

This is a very brief introduction to fuzzy logic. The subsequent sections focus on fuzzy logic being applied to fuzzy control.

2.3 Fuzzy logic control

In this section the most common way to derive a fuzzy controller is discussed. As an example, fuzzy logic is being applied to a simplified inverted pendulum problem, considering balancing of the pendulum and ignoring where the cart is positioned. This process has 2 states: the angle (φ), to the pendulum and its angular velocity (φ'). A schematic fuzzy controller for this process could look as in Figure 2.8.

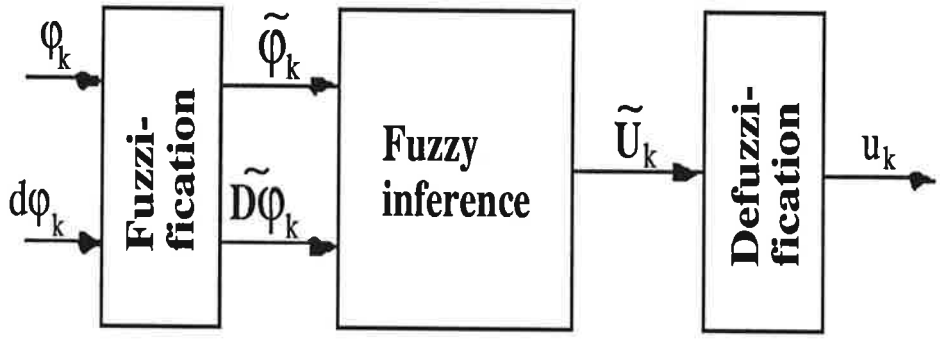


Figure 2.8. Schematic of a fuzzy controller.

The different components of the fuzzy controller: fuzzification, fuzzy inference and defuzzification are explained in the following sections.

2.3.1 Fuzzification

Fuzzification means that at every sample the numeric values of the states of the process are mapped to classes of fuzzy subsets. In this case φ and $d\varphi$ are mapped to the classes φ and $D\varphi$. The way this classes of fuzzy subsets are defined is by partitioning the state spaces into a certain amount of divisions.

Example: If φ can have values between +1.0 and -1.0 radians it may be appropriate to partition the universe for this state into 5 divisions, as shown in Figure 2.9.

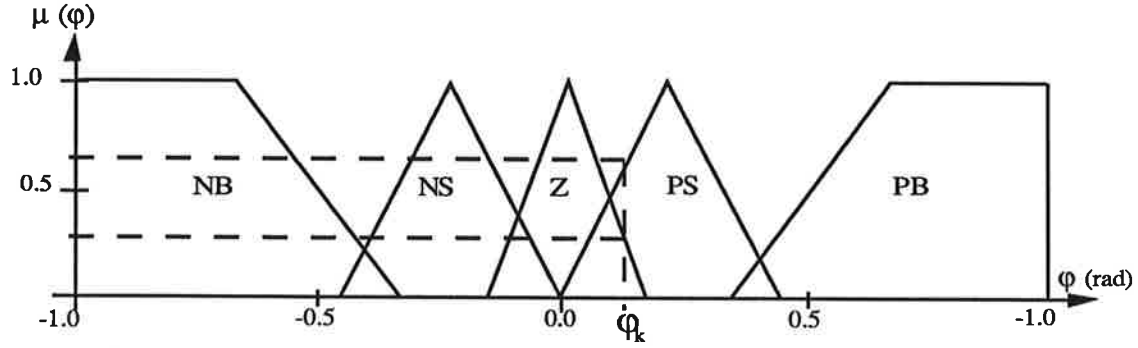


Figure 2.9. The 5 fuzzy subsets that represent the state φ: NB=Negative Big, NS=Negative Small, Z=Zero, PS=Positive Small and PB=Positive Big

This means that the class of fuzzy subsets for the state φ is:

$$\varphi_k \ni \{NB,NS,Z,PS,PB\}$$

To fuzzify the sample φ_k=0.15 in Figure 2.9 means that the corresponding membership functions $\mu(\varphi_k)$ will have the following values:

$$\mu_{NB}(0.15) = 0.00$$
$$\mu_{NS}(0.15) = 0.00$$
$$\mu_{Z}(0.15) = 0.30$$
$$\mu_{PS}(0.15) = 0.65$$
$$\mu_{PB}(0.15) = 0.00$$

These degrees to which the different subsets have been fulfilled will be used afterwards to infer the controller output.

2.3.2 Fuzzy inference

A common fuzzy rule format looks as below:

IF φ is PB **THEN** u is NB (R_1)
IF φ is PS **AND** φ' is NS **THEN** u is NS (R_2)

etc.

The output u is inferred from the rule premise by some fuzzy relational composition method. The most common of these methods is called max-min composition or compositional rule of inference [7]. The theory

behind max-min composition will not be discussed, but only explained by applying it on the two rules, R_1 and R_2, above. Each of these two rules will, according to this inference method, contribute to the fuzzy output U with a fuzzy subset. The more a rule is fulfilled, the higher value will the membership function for this fuzzy subset reach.
Let us label these fuzzy subsets for R_1 and R_2 to NB' and NS'.

Assume a sample where $\varphi=\varphi_k$ and $\varphi'=\varphi'_k$

The membership function for NB' (R_1) will at this sample be calculated as below:

$$\mu_{NB'}(u)=\min(\mu_{PB}(\varphi_k),\ \mu_{NB}(u))$$

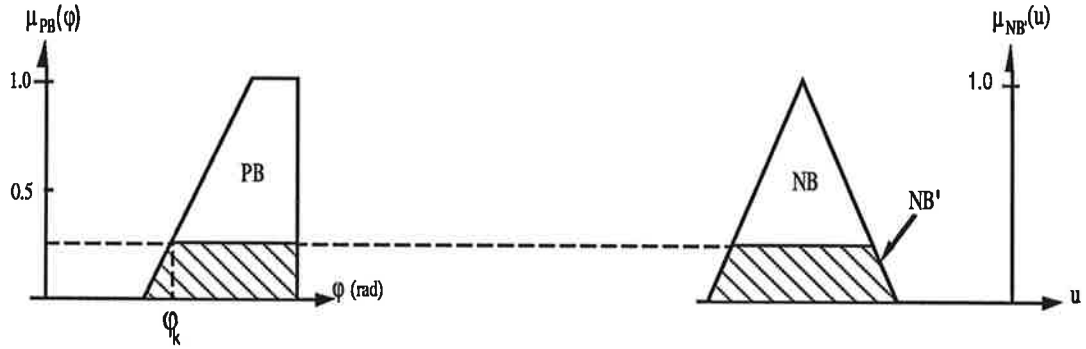


Figure 2.10. Max-min composition applied on R_1.

The second rule has two premises, but the calculations of $\mu_{NS'}(u)$ are performed in the same manner:

$$\mu_{NS'}(u) = \min(\mu_{PS}(\varphi_0),\ \mu_{NS}(\varphi'_0),\ \mu_{NS}(u))$$

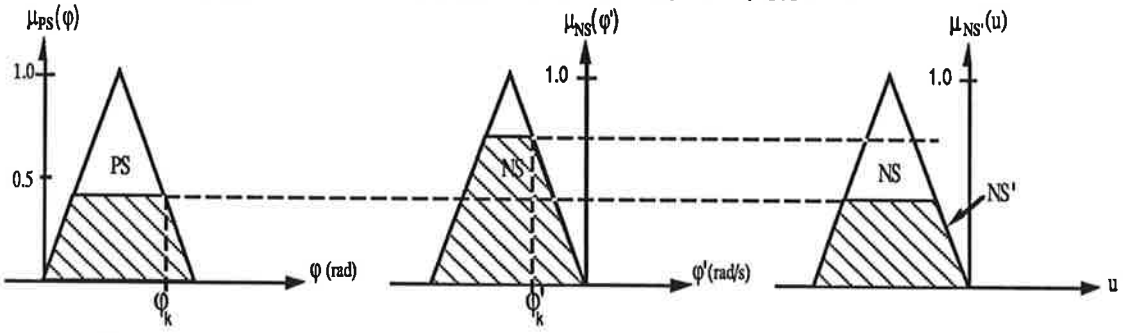


Figure 2.11. Max-min composition applied on R_2.

Another very similar inference method is called Max-Dot composition [12]. The only difference to max-min composition is that the output fuzzy set is changed to have a triangular shape, see Figure 2.12. Some

computation time is saved with this method because the output member is easier to compute when it has a triangular shape.

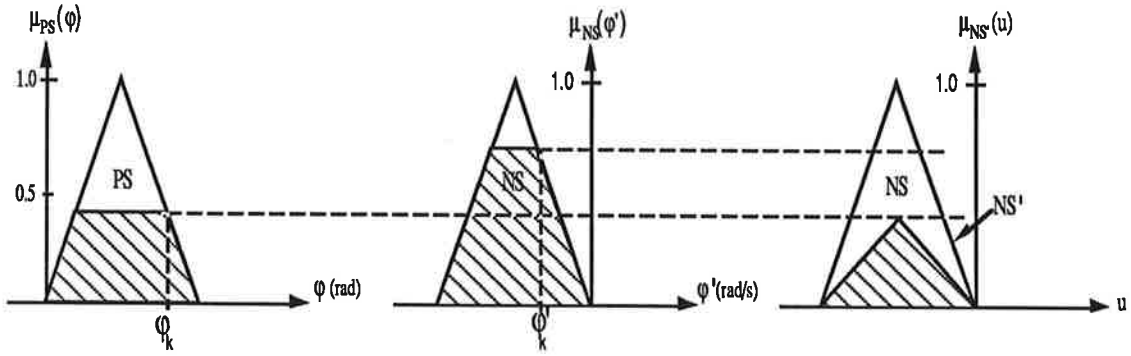


Figure 2.12. Max-Dot composition applied on R_2 .

2.3.3 Defuzzification

The fuzzy output U_k will be a sum of different fuzzy sets. This means that in the example treated in 2.3.1 and 2.3.2 U_k will be constructed as shown in Figure 2.13.

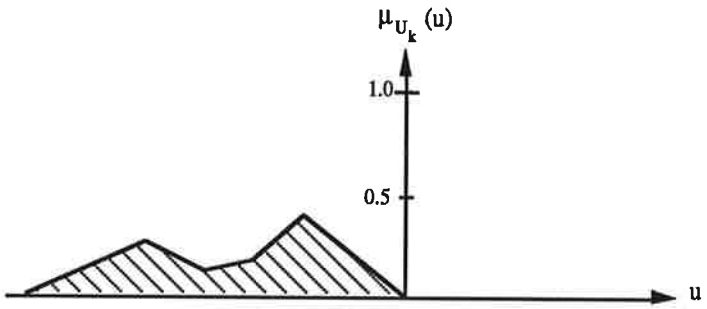


Figure 2.13. Fuzzy output U_k when max-dot inference method is used.

To achieve an output from the controller the function $\mu_{U_k}(u)$ in Figure 2.13 must somehow be converted into a crisp number. This is called defuzzification and the two most common methods are called the height and the centroid (center of gravity) method.

The height method is the most straightforward technique. As a representation for the curve the highest peak value is selected. But if $\mu_{U_k}(u)$ has more than one peak this method will of course be unfair. Moreover, a single rule may determine the final output. In the centroid method, instead, more activated rules will influence the result. The final output,

u_k, is calculated as the centre of gravity with respect to zero for the output function $\mu_{U_k}(u)$:

$$u_k = \frac{\int u \times \mu_{U_k}(u)\, du}{\int \mu_{U_k}(u)\, du}$$

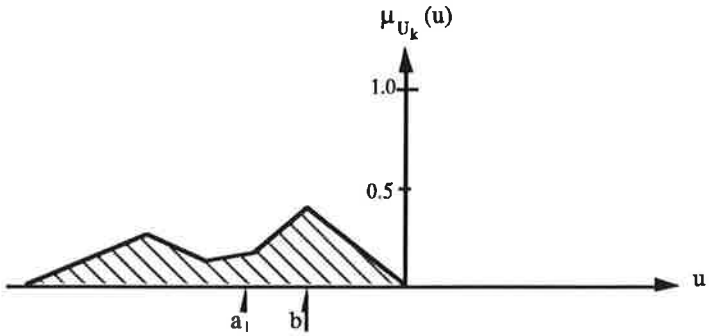


Figure 2.14. Final output from the fuzzy controller using (a) the centroid and (b) the height method.

3. Experimental set-up

In the first section the inverted pendulum problem will be discussed. The following section is a brief description of the working environment. It explains TIL Shell and gives a short description of the software that I wrote.

3.1 The inverted pendulum problem

3.1.1 Introduction

The problem is to balance an upright stick. The bottom of the stick is attached to a pivot on a cart that moves along a track as shown in Figure 3.1. Movement of both cart and stick is constrained to a vertical plane. The control action is to exert a force on the cart that gives it an acceleration to the left or to the right. This problem is clearly unstable by its nature. Furthermore, it is also a non-linear process. These features make this process an inviting candidate when investigating a new control strategy.

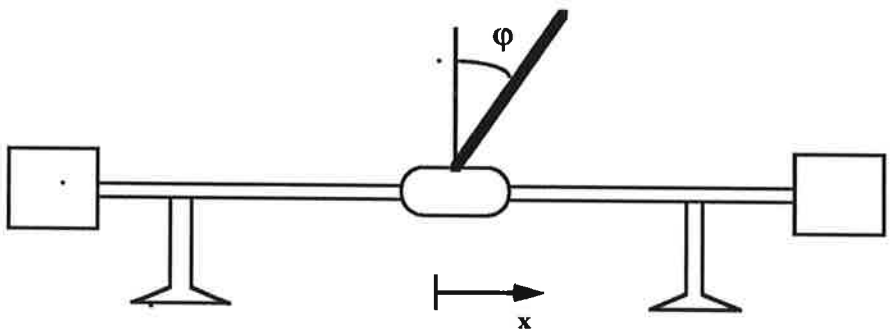


Figure 3.1 The inverted pendulum process

The prototype that has been used has an electrical servo motor to pull the cart. The angle and position are measured and their derivatives are computed (with high pass filtering). Two undesired conditions that arise in measurements are:

- The measurement of φ is heavily influenced by noise.

- There is a hysteresis around zero for the measurement of x. (See Figure 3.2).

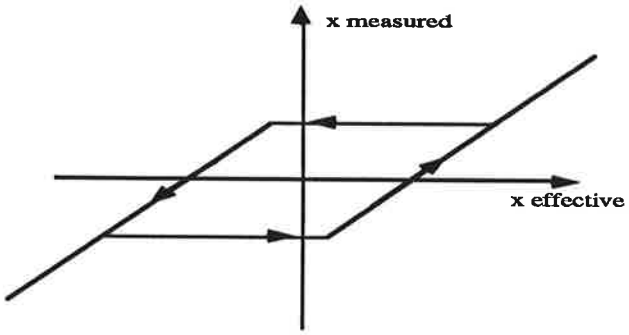


Figure 3.2. Hysteresis of a position measurement due to the sensor mechanics

3.1.2 Process model

A model that has been used is to regard the acceleration of the cart as proportional to the input to the process, see Figure 3.2 below.

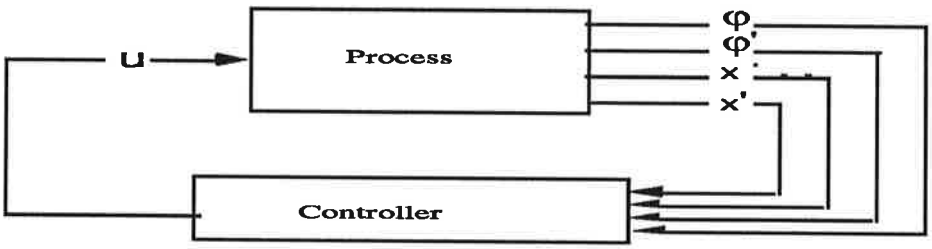


Figure 3.3 Process model of the inverted pendulum

The linearized differential equations for this process model can be written as:

$$\frac{d^2x}{dt^2} = u \qquad (1)$$

$$\frac{d^2\varphi}{dt^2} = p \times \varphi + q^2 \times \frac{d^2x}{dt^2} \times \cos(\varphi) \qquad (2)$$

The constants p and q are physical properties of the system

This model is, however, a simplification of the real process in two respects:

1: The friction is not modelled.

The acceleration of the cart depends not only on the input to the

process, but also of the cart's friction against the rail and air resistance. It is possible to get rid of this problem by feeding back the cart's actual speed in an inner loop and letting the control signal be the desired speed instead of acceleration. Experiments using this approach on the equipment available at the Research centre have shown, however, that this is not an appropriate model for the implementation used here.

2: The differential equations are not uncoupled.

The movements of the stick are in reality affecting the cart. However, if the cart is much heavier than the stick this effect will be very small. In this case the ratio between the cart's and the stick's mass is approximately 10. It has therefore been decided to discard this effect.

3.1.3 Digital controller

The design technique used is state feedback control. On the real equipment at the Research Centre a well operating digital controller did already exist and was therefore used as it was. The controller had been tuned on-line and the control design was not documented.

Worth to note here is that the feedback from the x and x' is positive! I.e the way the cart is moved in the right direction is in fact by making use of the strong balancing feedback.

3.2 Working environment

The working environment is realized on a Macintosh2 and main components are shown in Figure 3.4. In the first section TIL Shell will be explained. It is a software development tool used for creating fuzzy controllers. In the next section the written software modules will be shortly described.

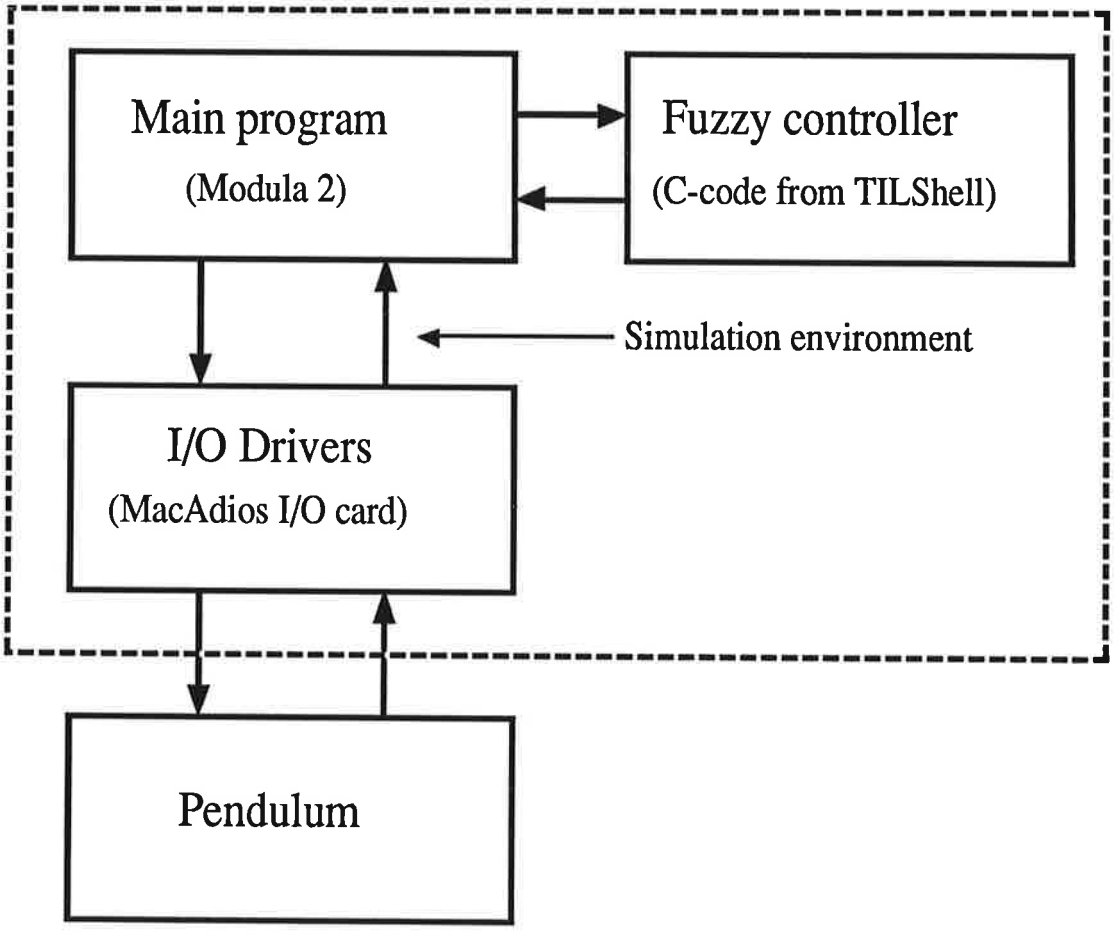


Figure 3.4. The different modules and their connections in the working environment.

3.2.2 TIL Shell

TIL Shell is a software development tool which has been developed by Togai InfraLogic (Irving, CA, USA) [12].

The TIL Shell is a graphical programming environment exclusively designed for generating fuzzy logic controllers. Projects are first written in a graphical language labeled Fuzzy Programming Language

and this code is then compiled to standard C-code. The structure of the graphical language and the objects that are possible to manipulate within it are shown in Figure 3.4.

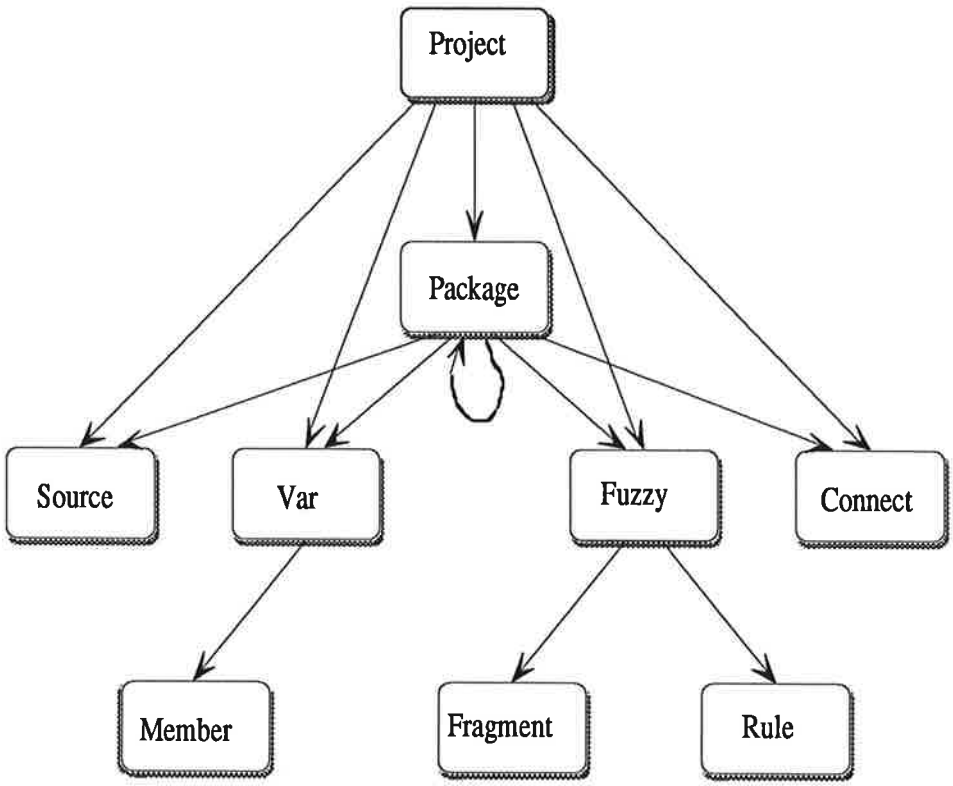


Figure 3.5. Structure of the graphical programming language used in TIL Shell.

The objects are altered using different editors: Project editor, rule editor, source editor, and so on.

Fuzzy objects

Fuzzy objects define the fuzzy rulebases. It is made up of rule and fragment objects that are displayed in the fuzzy editor.

Rule objects

The rules are defined in the rule editor. This editor will be started whenever a rule is opened or added in the fuzzy editor. It is possible to add and modify a rule's premises and conclusions. The different

conjunctions in a premise can be AND or OR. It is also possible to use the complement of the input variables (see Figure 2.7).

Source and fragment objects

Source and fragment objects are both used to embed C-code in the program, but in different ways. The source object provide the facility to implement a complete procedure, for example a driver for the controller, whereas fragments enable the user to stick in single lines in between two rules. The fragments are used if, for example, the controller is to be changed from outside TIL Shell. For the design of the controller in chapter 5, it was necessary to have the degree to which each rule was fulfilled at every sample. This was done by using fragments (see further Appendix B).

Var objects

These objects correspond to inputs and outputs to the system. The state space as well as possible initial and default variable values are chosen here. It is also possible to decide whether a variable should be represented as fuzzy or crisp.
To represent a variable as fuzzy is, of course, useful when the input is distorded by noise, but it could also be of value to use this facility to model the error arising in a measurement. However, in this report all variables are regarded as crisp.

Member objects

The member objects are defined in the var editor. It is possible to represent membership functions with an arbitrary mathematical equation. However, the most common representation and the one used in this report is trapezoid functions. A tempting use of a more complicated representation results in a lot more computation time.
The design of a fuzzy controller is very much of an experimental nature and especially so in terms of choosing the membership functions. Therefore, it would be desirable to be able to change the shape of a membership function outside TIL Shell while running the control application. This is very simple to do if trapezoid functions are used.

Project, package and connect objects

The purposes of these objects are quite straightforward and not worth

any lengthy discussion. Project objects define the fuzzy system, package objects are used to divide the project into different parts for the sake of lucidity and connect objects join different objects together

Compiling the FPL-Code into C-code

There are several options on how to compile the finished project. Defuzzification can be made using either the centroid or the height method. As for the inference method it can be chosen as either max-min or max-dot. The main reason for choosing max-dot is that it is a faster method than max-min. Furthermore, there are several different options concerning how to represent the membership functions. They can be represented as either fix maps or generated when the C-procedure is called. Again it is a question of time which one to choose.

In this project defuzzification has been done by taking centroid, inference method is according to max-dot, and representation of the membership functions are fixed.

3.2.3 Software

The main program is constructed so that it is possible to switch between a fuzzy and a digital controller. Furthermore, it is also possible to choose between a simulated environment, using a linearized process model, and the real equipment. A simple plot function provides the facility of showing the results on the screen and storing them on a file.

All software modules are written in Modula2. Because TIL Shell generates C-code an interface between these two languages has been required. In the Modula2 manual there is a short chapter concerning this. It is explained that Modula2 and C push variables in a different order on the stack. But, it is insufficient only to take this into account because the datatypes are also represented differently. Example: In C, 32 bits will be reserved for a value parameter of type signed byte. However, if the variable is represented as a variable parameter (represented as a pointer in C) of type signed byte only 8 bits will be reserved!! There are several examples of a similar nature and it is therefore very hard to give any overall direction of how this interface should be designed. It is highly recommended to look into the generated assembler-code and make sure how many bits are reserved in different situations.
For the communication between the pendulum equipment and a I/O driver module, a MacAdios Jr card was installed on the Macintosh. The

delivered software driver for this card turned out to be extremely slow. Therefore, a new extra driver was written in Modula2. A read or write operation with this new driver requires less than 0.1 ms, an improvement with a factor 20.

4. Fuzzy controller design using traditional fuzzy reasoning

4.1 Introduction

With traditional fuzzy reasoning is understood that the output from the controller is inferred according to some fuzzy relational composition method, for example max-dot composition, as discussed in section 2.3.

A main problem when designing a fuzzy controller of this type is to find out conclusions for each rule. If the process has previously been operating well, thanks to a skilled operator, it is wise to use his knowledge when deriving rules. Otherwise, as in the case with the inverted pendulum problem, a rule base has to be designed based on intuition and possible knowledge about the process.

Another disadvantage with this method is that, in general, many partitions are needed for each state variable. This leads to a rapidly growing number of rules as the number of state variables increases. For the inverted pendulum problem with 4 state variables (angle φ, angular velocity φ', cart position x and cart speed x') this drawback was extremely difficult to handle. Through experiments it became evident that φ had to be partitioned in at least 5 divisions. The 3 other variables have to be partitioned into at least 3 divisions (Negative, Zero and Positive), but this leads to a large rule base consisting of minimum $5*3^3$ =225 rules!
To derive so many rules by hand is practically impossible; thus in this report the rules have been reduced by decomposing the problem into different subtasks. Each task is a fuzzy controller specialized in solving one part of the problem. The output from the different tasks will not be defuzzified separetely, but they will contribute to the output u with the rules being activated at each sample. The conclusions from all activated rules will then be defuzzified together and generate the controller output. This means that the controller output (u) is not generated as an arithmetric sum of the different tasks, but as a weighted average of all activated rules. The consequence of this is discussed in chapter 4.3.

For the inverted pendulum process the fuzzy controller is divided into 3 tasks:

$$u = \{\text{Balancing}\} + \{\text{Positioning}\} + \{\text{Halting}\}$$

The balancing task will be discussed in section 4.2. The positioning and

the halting tasks are described in section 4.3. In the last section, 4.4, the performance of the resulting fuzzy controller is discussed and compared to the performance of a digital controller. The two controllers are run both on a simulation basis and on a real equipment.

4.2 Implementation of the balancing task

How to choose the fuzzy subsets for φ and φ' is mainly a matter of experiments. The state spaces for φ and φ' were chosen to be from +0.8 to -0.8 radians and from +4.0 to -4.0 radians/second. With these ranges it became evident that both variables had to be partitioned in at least 5 divisions.
The reason for this is that the resolution will not be good enough with only 3 divisions. The controller will be either too fast or too slow depending on whether the fuzzy subsets are approaching or moving away from zero. To use 7 divisions would of course be even better in terms of resolution, but results in an increase of rules from 25 (5 divisions) to 49. It was therefore decided to use only 5 divisions. The universe of discourse for the output was partitioned into 7 divisions.

φ	φ' NB	NS	Z	PS	PB
NB	1 NB	2 NB	3 NB	4 NM	5 NS
NS	6 NB	7 NM	8 NS	9 Z	10 PS
Z	11 NM	12 NS	13 Z	14 PS	15 PM
PS	16 NS	17 Z	18 PS	19 PM	20 PB
PB	21 PS	22 PM	23 PB	24 PB	25 PB

Figure 4.1. The final decison table for the balancing task. Each number at the upper left corner indicates a rule index made

The rule base and the membership functions for the states and the output were at first intuitively designed. The rule set was tested against a

pendulum simulation program based on a linearized model of the process and compared to how the digital controller behaved. The digital controller therefore served as a reference of a well operating controller.

The tuning of the controller was done by changing both the rules and the membership functions. A lot of efforts were put in finding well-operating membership functions, but the rule base proved to be almost optimal from the beginning. The final rule base looks as shown in Figure 4.1 and only two changes have been made during the simulations. Originally, rule numbers 10 and 16 had a conclusion of u equals zero instead of the final PS and NS. The strategy was then to leave the pendulum alone if it was moving in the right direction, but this caused big overshooting in many cases. Therefore it was decided to change these rules and thereby braking the pendulum more before it reached the centre point.

When implementing the rule base two simplifications were done to reduce the number of rules. Rule 1, 2 and 3 and rule 23, 24 and 25 were reformulated as:

IF (φ is NB) **AND** (φ' is **NOT** POS) **THEN** u = NB

IF (φ is PB) **AND** (φ' is **NOT** NEG) **THEN** u = PB

The representations of the fuzzy sets NEG and POS are shown in Figure 4.2b. Through this simplifcation the final rule base is reduced to a total of 21 rules. The advantage with this reduction, however, is minor.
Note that it would not be correct to rewrite rule 1,2 and 3 neither as:

IF (φ is NB) **AND** ((φ' is NB) **OR** (φ' is NS) **OR** (φ' is Z))

THEN u = NB

nor as:

IF (φ is NB) **AND** (φ' is **NOT** PS) **AND** (φ' is **NOT** PB)

THEN u = NB

The final membership functions are shown in Figure 4.2. The dashed membership functions for NS and PS are dated from an early experiment . They are included because the transition from those to the

tilted fuzzy subsets for NS and PS involved a significant improvement of the fuzzy controller.

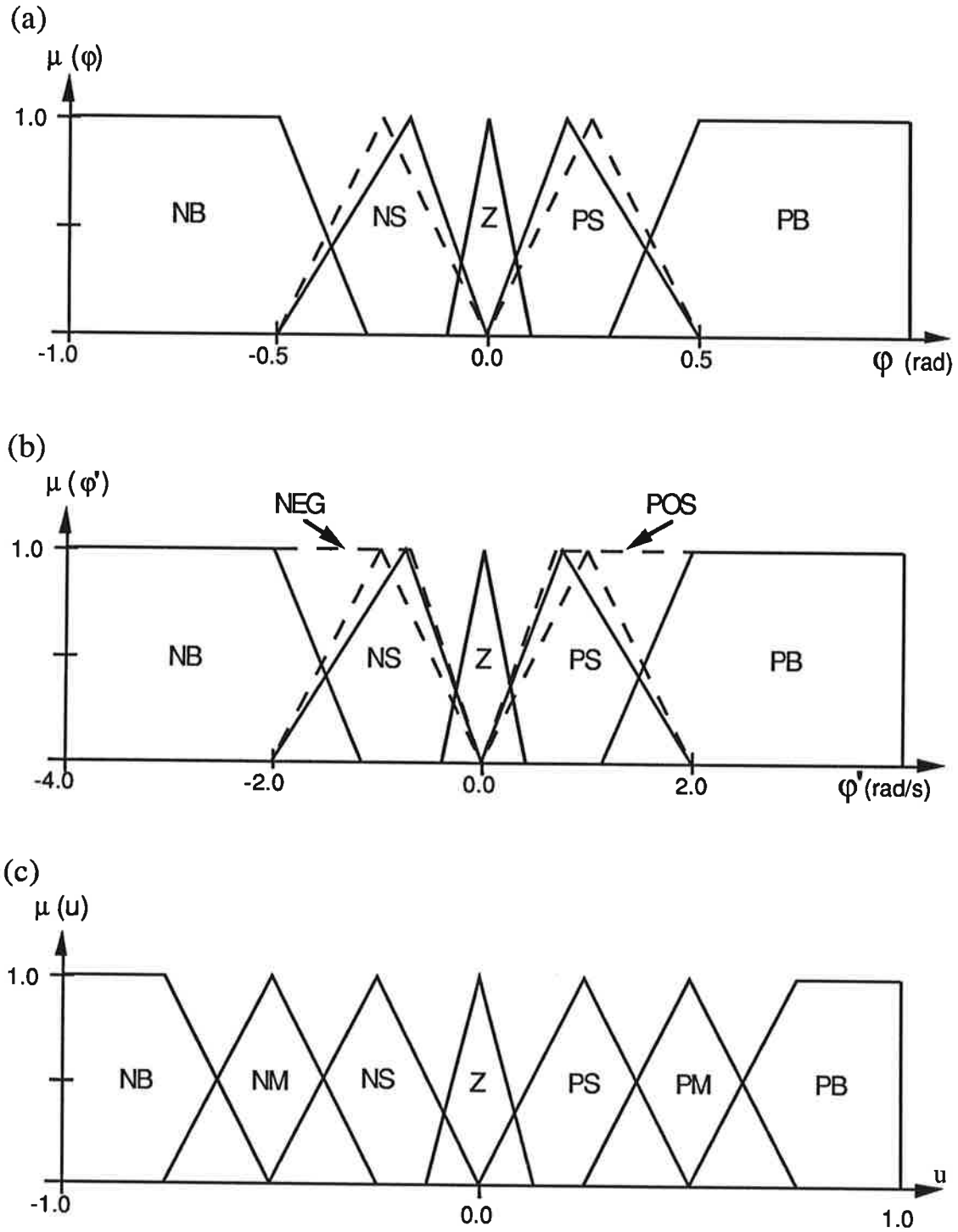


Figure 4.2. Membership functions for (a) angle, (b) angular velocity and (c) output.

In the beginning the fuzzy controller caused large oscillations around the set point, but by moving the tip of the membership functions for NS

and PS towards zero this problem was reduced drastically, see Figure 4.3.

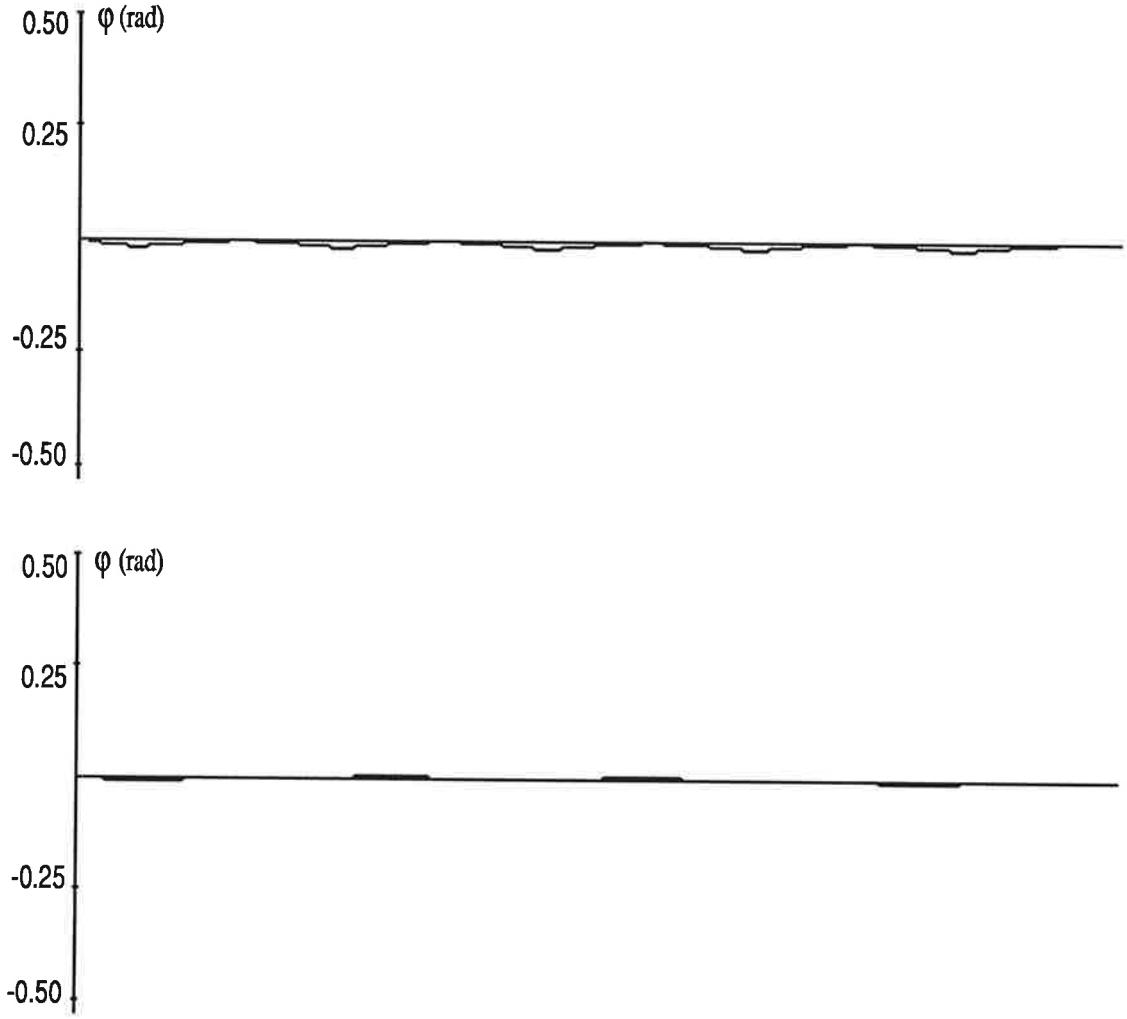
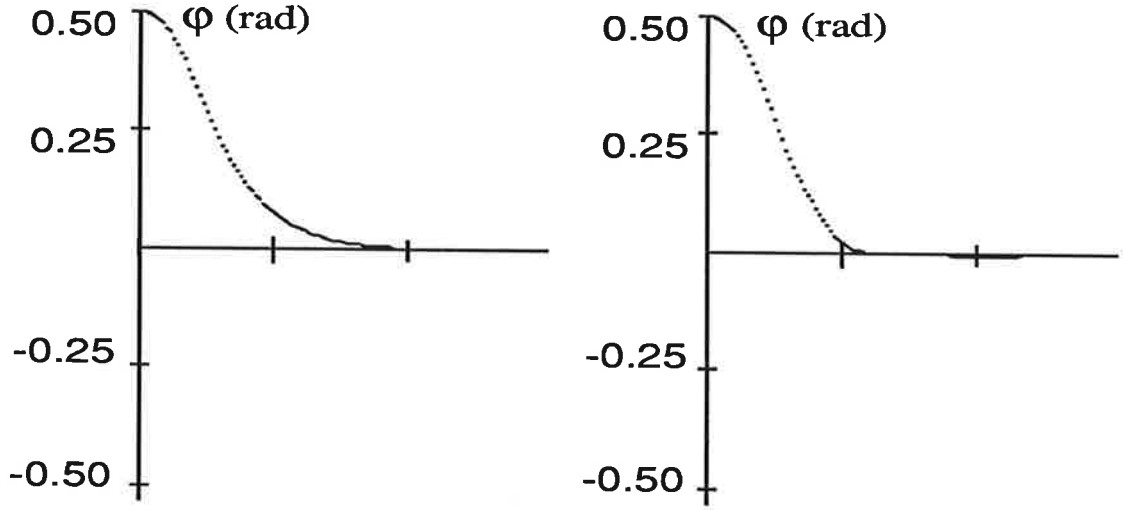


Figure 4.3.Oscillations around j=0.0 before (upper) and after (lower) the change of the membership functions for NS and PS.

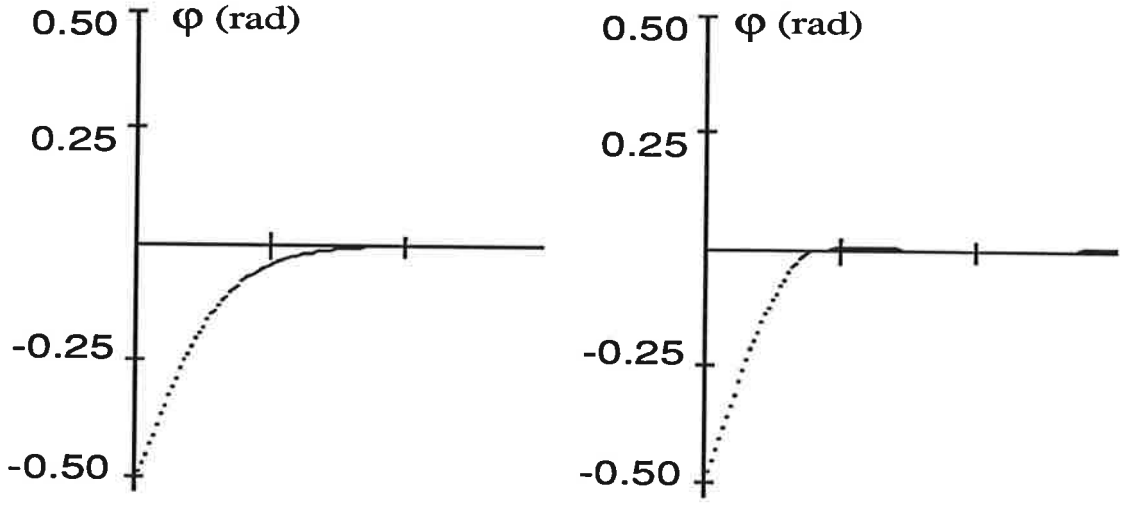
The effect is that the amplification for the controller is turned up in the region around zero and turned down a little for bigger angles. The last point implies that the controller should respond slower for bigger angles, but in practice this effect was not noticeable. In order to suppress the oscillations completely it would probably have been best to introduce a new set of membership function between Z and NS and Z and PS, but that would also have involved a lot more work in terms of a more complex rule base. Note that the fuzzy subset Z for the angle and angular velocity is very small. This shape also increases the amplification around zero for the controller.
Three simulations, with different settings of initial values, were run using a linearized model and being compared to a digital controller in Figure 4.4 to Figure 4.6. Besides the problem of oscillations the fuzzy

controller compares quite well with the digital controller. In some cases it performs better than the digital controller and in some cases it performs slightly worse. With more tuning probably a very good result would have been achieved, but fine tuning is extremely time consuming to do manually and was therefore left undone.



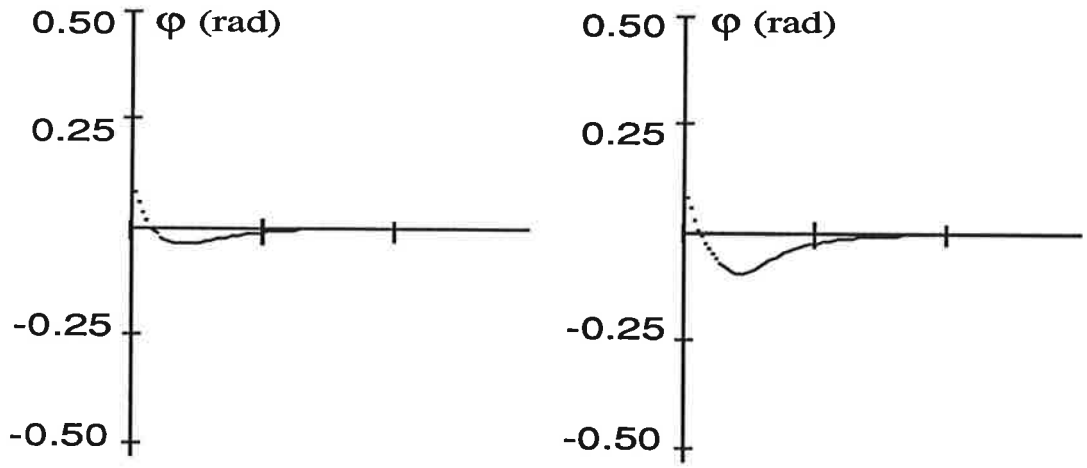
(a) Digital controller (b) Fuzzy controller

Figure 4.4. Simulation of (a) digital and (b) fuzzy controller for φ_0=0.5 rad and φ_0'=0.0 rad/s.



(a) Digital controller (b) Fuzzy controller

Figure 4.5. Simulation of (a) digital and (b) fuzzy controller for φ_0=-0.5 rad and φ_0'=0.3 rad/s.



(a) Digital controller (b) Fuzzy controller

Figure 4.6. Simulation of (a) digital and (b) fuzzy controller for φ_0=0.1 rad and φ_0'=-0.5 rad/s.

4.3 Implementation of halting and positioning task

The purpose of these two tasks is to bring the cart to the centre and forcing it to remain there. The feedback from x and x' is positive and the positioning is therefore accomplished by trying to move the cart in the opposite direction than the desired one.

The halting task

The halting task will only be activated when the stick is balanced out and the cart is close to the centre. Namely, the cart is in the centre region, but moving away or coming towards the centre point. To accomplish this the membership functions for position and speed were declared as shown in Figure 4.6:

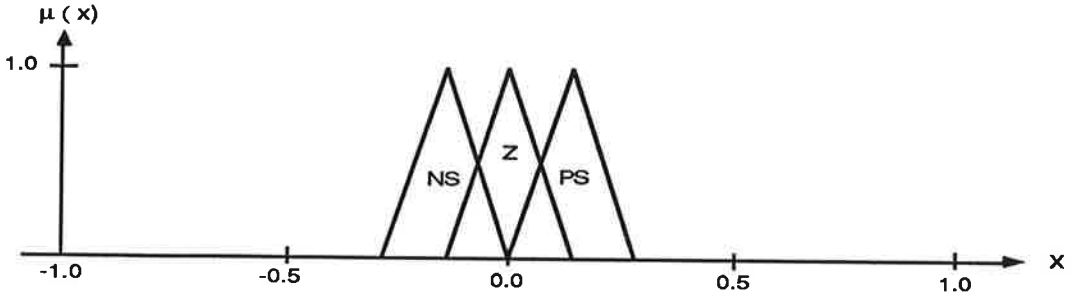


Figure 4.6a. Membership functions for the position used in the halting task.

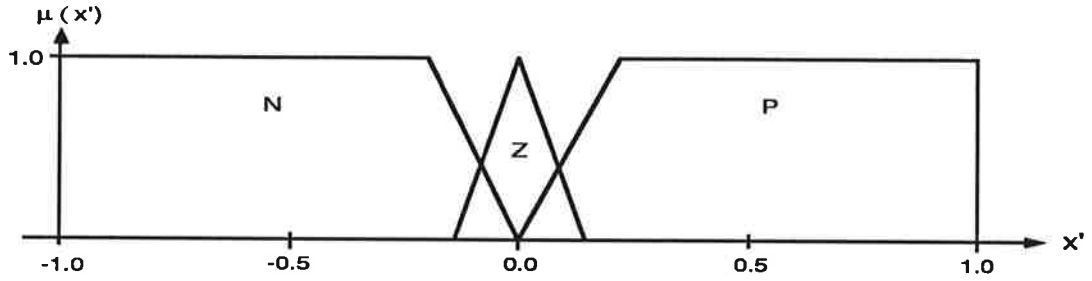


Figure 4.6b. Membership functions for the speed used in the halting task.

Two additional membership functions NZ (Negative Zero) and PZ (Positive Zero) are added to the output u, as shown in dashed lines in Figure 4.7:

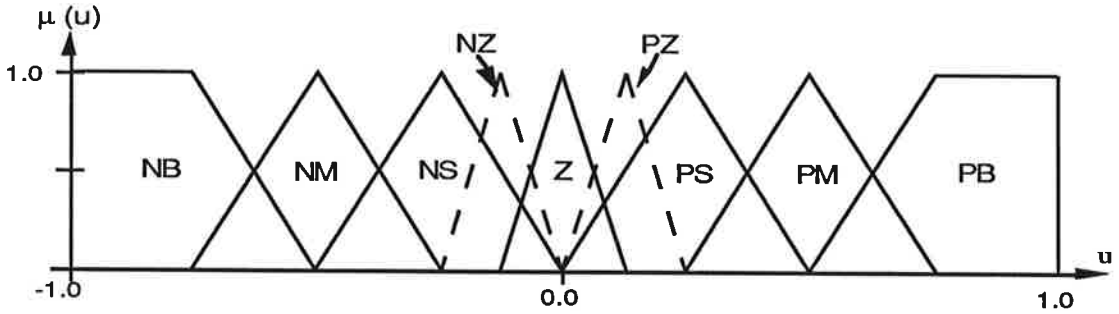


Figure 4.7. Additional membership functions, NZ and PZ, for the output u.

Four rules are declared for the halting task:

IF (φ is Zero) **AND** (φ' is Zero) **AND** (x is Positive Small)
AND (x' is Negative) **THEN** u = Negative Zero (R_{22})

IF (φ is Zero) **AND** (φ' is Zero) **AND** (x is Negative Small)
AND (x' is Negative) **THEN** u = Negative Small (R_{23})

IF (φ is Zero) **AND** (φ' is Zero) **AND** (x is Negative Small)
AND (x' is Positive) **THEN** u = Positive Zero (R_{24})

IF (φ is Zero) **AND** (φ' is Zero) **AND** (x is Positive Small)
AND (x' is Positive) **THEN** u = Positive Small (R_{25})

The first two rules act on the pendulum when it is coming from the right to the left and the other two when it is moving in the opposite direction. The reason why NS and PS could not be used in all cases is

that it is sometimes a too strong action to use. If the pendulum is coming towards the centre, but with a very low speed, the use of NS instead of NZ might impede the pendulum too much and force it to change its direction and not reaching it's goal. This reasoning is totally of an experimental nature and it may be feasible to use other configurations of membership functions

The positioning task

The positioning task will be activated independently of the pendulum´s position. This fact makes it more complicated to implement than the halting task, because the total output from the controller is not generated as an arithmetric sum, but as the centre of gravity of the contributing rules. This means that the positioning task will interfere with the halting task if no precautions are taken. An example explains this easily. Suppose that the pendulum is located as in Figure 4.8.

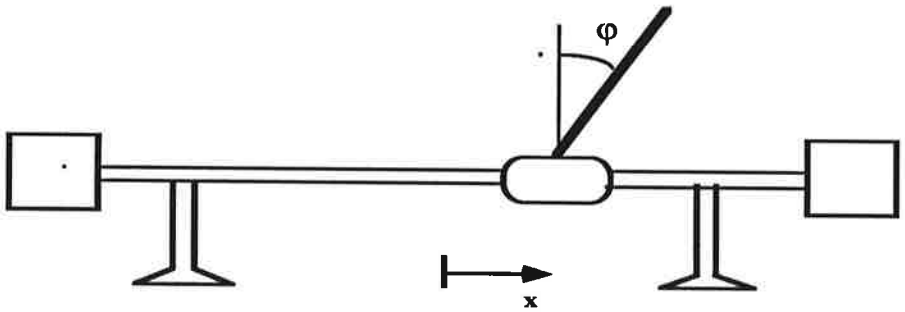


Figure 4.8. The inverted pendulum.

A big positive force is required to balance the pendulum and an additional small positive force is desired to positioning the pendulum (Note the positive feedback of x). Assume this is implemented as below:

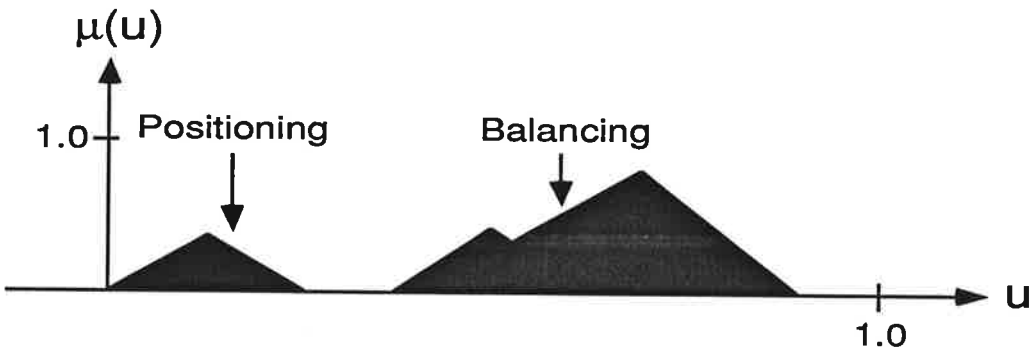


Figure 4.9. An incorrect attempt to superimpose the positioning force.

Because of the centroid defuzzification the resulting control action will be smaller than if only the balancing action had been used!

The desired result would instead look as shown below:

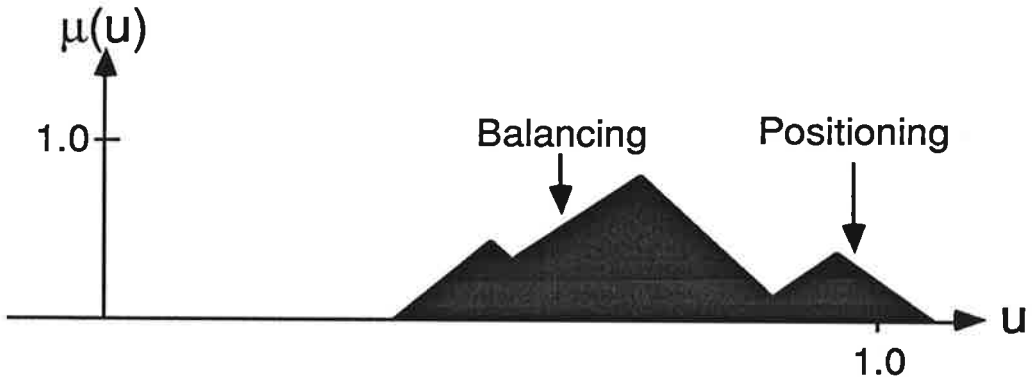


Figure 4.9. The desired goal of the superimposition of the balancing and the positioning forces.

The weight of the positioning must always be very small and subordinated to the balancing act. Clearly, it is impossible to use regular membership functions for x and design a rule like:

IF X is P **THEN** u = PB

But, if this rule is never fulfilled to more than a degree of, say, 5-10 percent, it can perform the desired contribution. This is indeed what has been used to implement the positioning act. A new type of membership functions has been introduced for x as shown in Figure 4.10.

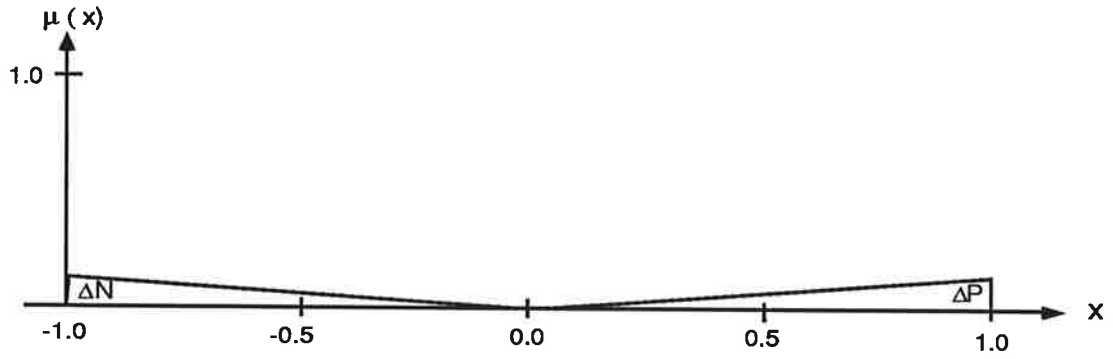


Figure 4.10. Membership functions for x used in the positioning task.

The rules for feedback from x have been formulated as:

IF (x is ΔP) **THEN** u = PB (R_{26})

IF (x is ΔN) **THEN** u = NB (R_{27})

Feedback from the position alone is insufficient to control the cart; the speed must also be considered. Experiments proved that the feedback from the speed could be a great deal 'larger' than the feedback from the position and the membership functions were therefore chosen as in Figure 4.11.

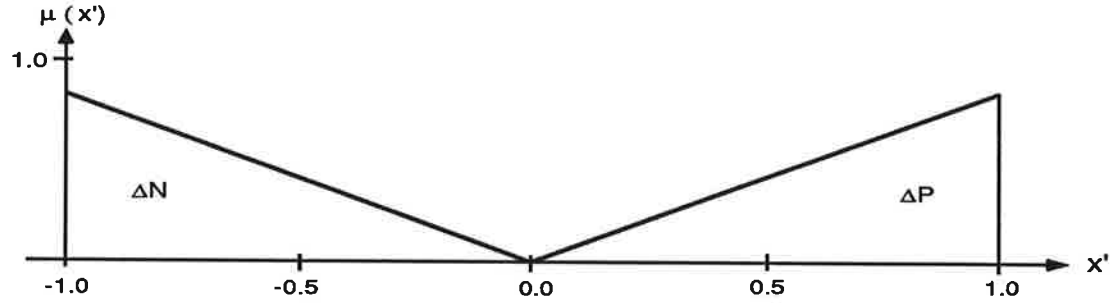


Figure 4.11. Membership functions for x' used in the positioning task.

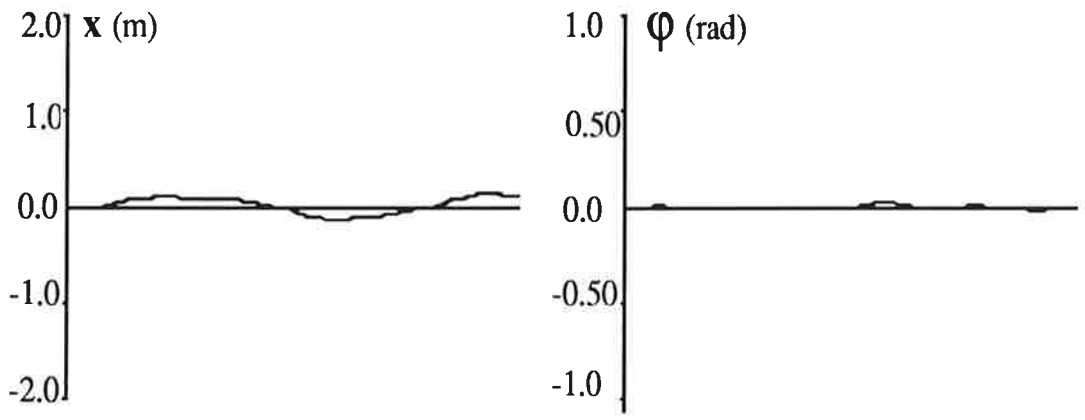
The corresponding rules for this are defined as follows:

IF (x' is ΔP) **THEN** u = PB (R_{28})

IF (x' is ΔN) **THEN** u = NB (R_{29})

4.4 Experimental results

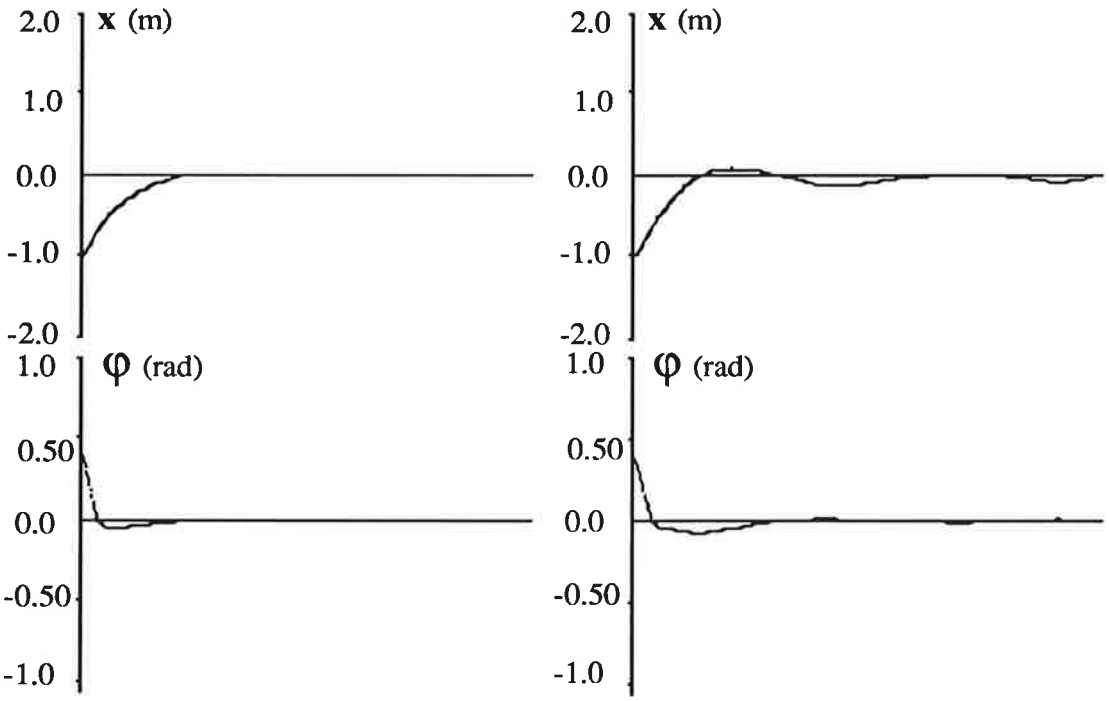
All in all 29 rules were needed to implement this fuzzy controller and the throughput time is approximately 0.8 ms. The controller was run both on a linearized model and on the real equipment. One big difference between the fuzzy and the digital controller on simulation basis is that the fuzzy controller had severe problems with oscillations around the set point, as shown in Figure 4.12. The oscillations around φ=0.0 have been mentioned in chapter 4.2 and are acceptable, but the oscillations around x=0.0 are a great deal larger and all attempts to suppress them failed. A solution may be, as for the oscillations around φ=0.0, to increase the controller´s amplification when x nears to zero. However, no successful result has been achieved so far.



(a) Centre position (b) Centre angle

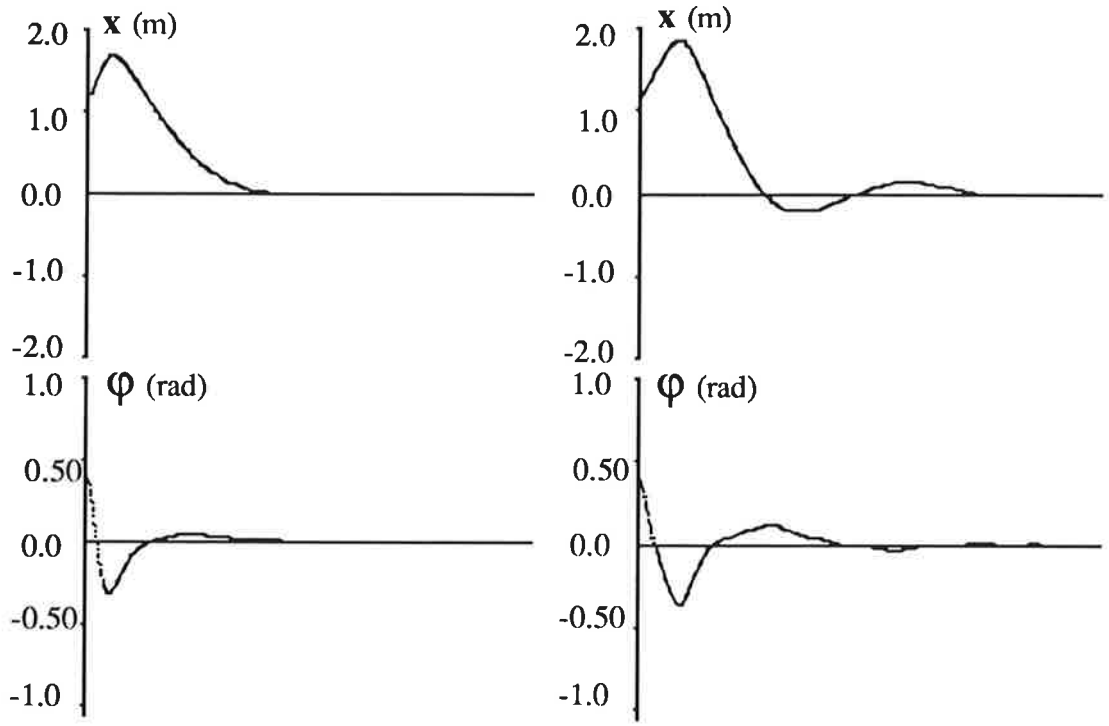
Figure 4.12. Oscillations of the fuzzy controller around (a) x=0.0 m and (b) φ=0.0 rad using a pendulum simulator.

Using a pendulum simulator, the performance of the fuzzy controller is being compared to the digital controller under different initial values for φ and x, as shown in Figure 4.13 to Figure 4.16:



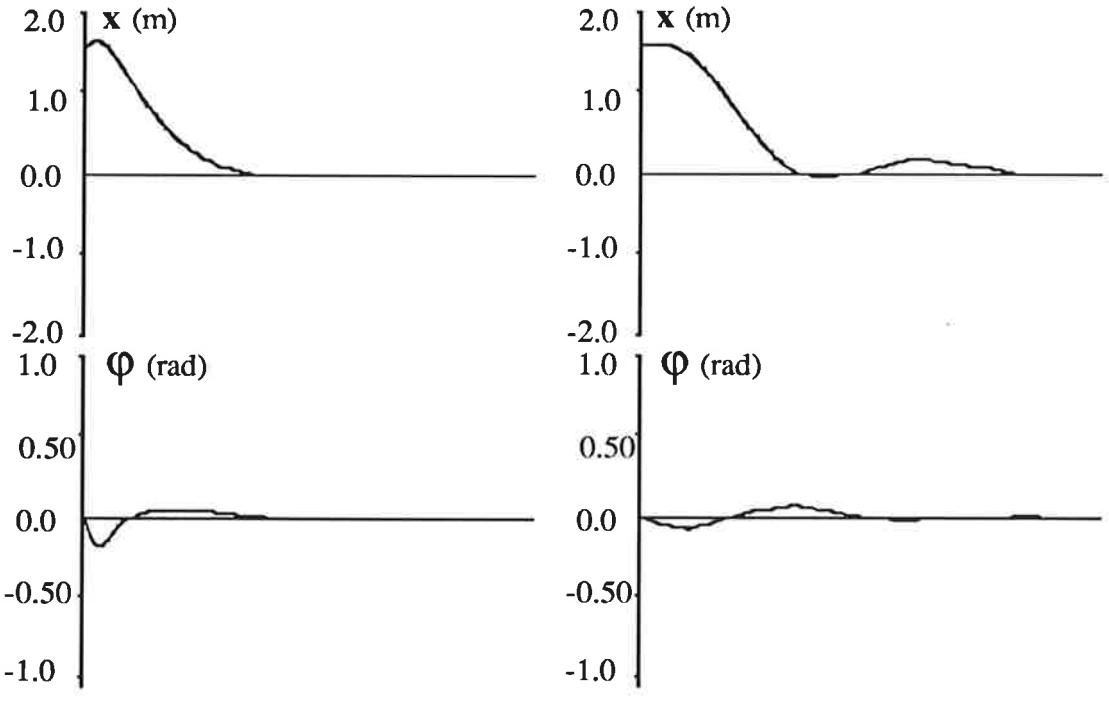
(a) Digital controller (b) Fuzzy controller

Figure 4.13. Performance comparison of (a) digital and (b) fuzzy controller with initial values for φ_0=0.4 rad and x_0=-1.0 m.



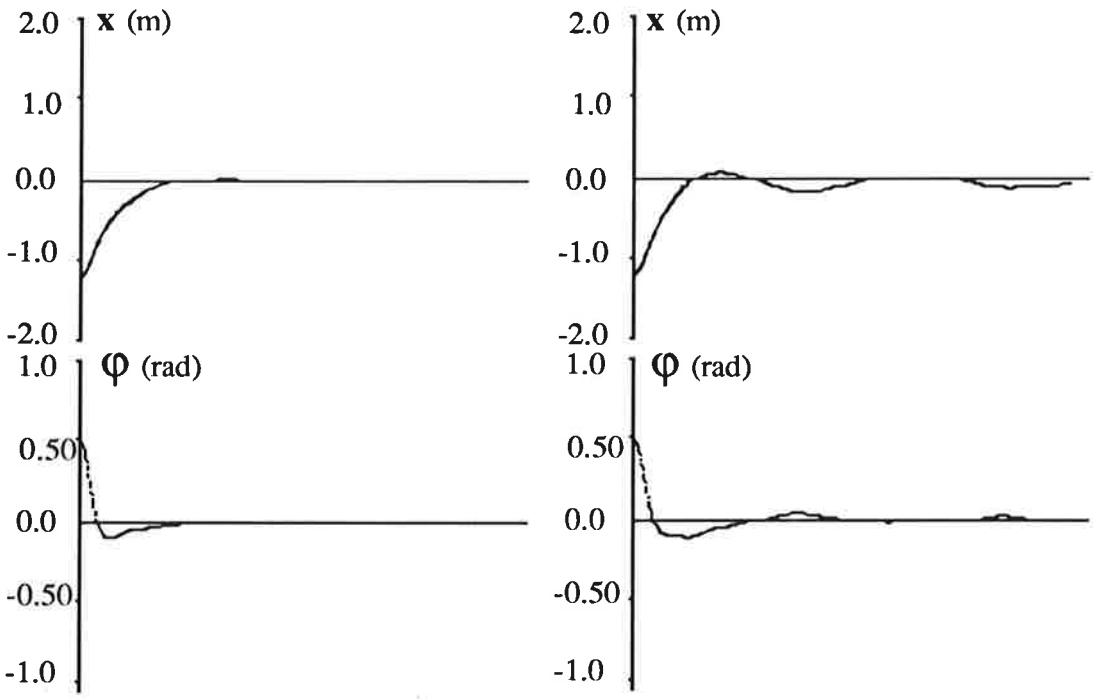
(a) Digital controller (b) Fuzzy controller

Figure 4.14. Performance comparison of (a) digital and (b) fuzzy controller with initial values for φ_0=0.4 rad and x_0=1.2 m.



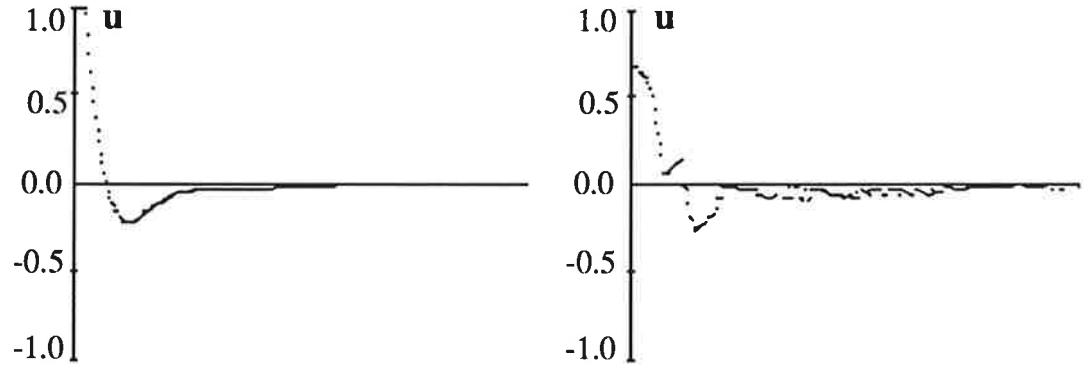
(a) Digital controller (b) Fuzzy controller

Figure 4.16. Performance comparison of (a) digital and (b) fuzzy controller with initial values for φ_0=0.0 rad and x_0=1.6 m.



(a) Digital controller (b) Fuzzy controller

Figure 4.16. Performance comparison of (a) digital and (b) fuzzy controller with initial values for φ_0=0.5 rad and x_0=-1.2 m.



(a) (b)

Figure 4.17. The output comparison for (a) digital and (b) fuzzy controller during the simulation run of Figure 4.14.

Disregarding the problem with oscillations the fuzzy controller works, but it is evident in these simulations that the digital controller is behaving much better. The controller output in Figure 4.17 shows clearly that the fuzzy controller is not optimally tuned. But the problem is that the controller is extremely hard to calibrate. A small change of the membership functions can have a considerable influence on the

controller output. There is an automated way to do this tuning, i.e., by using genetic algorithms [4]. The method seems to be promising, but due to the limited time it was not investigated.

The fuzzy controller was also run on a real pendulum equipment. It has been observed that the fuzzy controller performed, surprisingly, competitive to the digital controller. The controllers seem to be equally fast and in practice they both have a problem of drifting around the centre point. Why the fuzzy controller is working better on the real equipment was not verified, but one theory is that the unmodelled friction favours the fuzzy controller. Another idea is that the non-linear properties of the process favours the fuzzy controller.

5. Fuzzy controller design using 'weighted linear curve-fitting'

5.1 Introduction

The purpose of *weighted linear curve-fitting* is to model an expert's control actions as good as possible, and our motivation to investigate this appoach was triggered by the article Fuzzy control of a model car [10].

The first industrial application using fuzzy logic was to control a cement kiln. This process had previously been very hard to automate for several reasons and only an experienced operator could control it properly. The fuzzy controller was designed by incorporating control rules being documented in the operator manual. This is equivalent to question an operator what he would have done if the process parameters had changed in such and such ways. This information was directly used to create the rule base. The operator apparently knew what he was doing. Now assume that the operator is controlling a process very well, but that he can not clearly explain on what reasons he is taking a certain action. A method to use his decision making nevertheless would be to model his actions in a certain way.

The course of action for this modelling implies inferring the output for each rule as a function of the different states involved in the rule's premise. The new rule format looks is defined as follows:

R_i: **IF** x_1 is A_1 **AND** x_2 is A_2 **AND** ... **AND** x_k is A_k
THEN $y_i = f(x_1, x_2, \ldots, x_k)$

The conclusions y_i (i = 1.. n, n=number of rules) are constructed as linear functions of the different premise variables:

$$y_i = p_{i0} + p_{i1}*x_1 + p_{i2}*x_{i2} + \ldots + p_{ik}*x_k \qquad (1)$$

The final output, inferred from n rules, is calculated as a weighted average as shown in equation (2):

$$y = \frac{w_1*y_1 + w_2*y_2 + \ldots + w_n*y_n}{\Sigma_{i=1}^{n} w_i} \qquad (2)$$

The different weights (w_i , i = 1.. n) are the degrees to which each rule has been fulfilled. Note that the output function y is non-linear. The reason for this is that the different weights (w_i , i = 1.. n) are not constants, but functions of the different state variables (x_j , j = 1.. k). The coefficients (p_{ij}, i=1..n, j=1..k) are identified by first collecting input-output data on the process and then use, for example, least square estimation to obtain the coefficients that minimize the output error (see further Appendix A).

TIL Shell does not provide a facility to implement *weighted linear curve-fitting*. A modification had to be done to gain access to the different weights (w_i , i = 1.. n) at each sample. How this was solved is described in Appendix B.

In order to design a fuzzy controller with this technique it is thus required to have an expert's control action. The only expert available on the inverted pendulum problem was the digital controller and it served, therefore, as a supervisor for this process.

Solving the inverted pendulum problem by using *weighted linear curve-fitting* requires some consideration, for a large number of rules will cause some computational problems:

1. Too many sample data:
For 225 rules and 5 coefficients at least 1125 data points are required, but to achieve a confident result probably a double amount of data points is desired. The choice of these sample data cannot be chosen randomly, but they should at least cover the entire control surface. Namely, the sample data should be selected in such a way that all rules have been activated more or less a same number of times. It also seems natural to choose the data in such a way that critical process changes are well covered.

2. Computer intensive calculations:
To solve the system of equations requires a large amount of memory space and, additionally, much computation time as well.
With the program used in this report, the computation time is proportional to $O(n^2)$, where n is a number of rules.

The goal is to achieve a rule base with fewer rules than 225 and this was carried out in two different ways. In section 5.2 the problem is solved by dividing the rule base into two rule bases, one for balancing and one

for positioning. In section 5.3 experiments are done to solve the problem by having less divisions for each state variable. The results with these two approaches are discussed in the last section 5.3.

5.2 Solving the problem with two separated rule bases

The main idea with this method is to decompose the problem into different subtasks, i.e., very similar to method described in section 4.2. The rules are divided into balancing and positioning tasks, but the difference is that the conclusions are inferred according to the equations (1) and (2) on page 36.

It was decided to partition each variable in five divisions. The members for the angle and angular velocity were chosen as in Figure 4.2a and 4.2b and the membership functions for position and speed were chosen as in Figure 5.1.

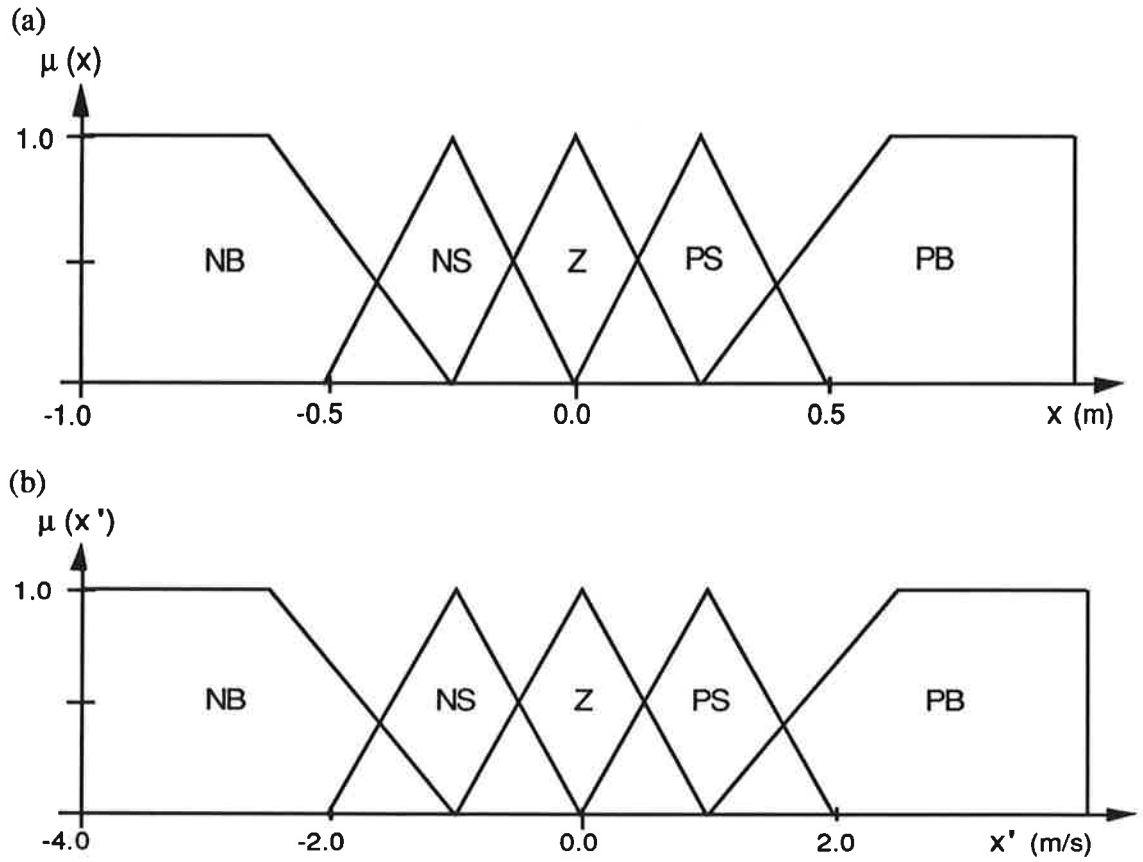


Figure 5.1. Membership functions for (a) position and (b) speed.

Five divisions for each state lead to a rule base of 5*5 + 5*5 = 50 rules:

IF (φ is NB) **AND** (φ' is NB) **THEN** $y_1 = f(\varphi, \varphi', x, x')$ (R_1)

.....

....

..

IF (φ is PB) **AND** (φ' is PB) **THEN** $y_{25} = f(\varphi, \varphi', x, x')$ (R_{25})

IF (x is NB) **AND** (x' is NB) **THEN** $y_{26} = f(\varphi, \varphi', x, x')$ (R_{26})

.....

....

..

IF (x is NB) **AND** (x' is NB) **THEN** $y_{50} = f(\varphi, \varphi', x, x')$ (R_{50})

Note that every rule has only two premises, but the conclusion is inferred from all four state variables! According to the theory for weighted linear curve-fitting this method is not correct, because a rule´s conclusion should be inferred from only the state variables involved in the premises. But this way to divide the problem has not been found in any publication, and at the time for the experiments it sounded natural that the effect of a balancing rule should depend on where the cart is positioned, and that a positioning rule should also depend on how the pendulum is balanced. Whether this is true or not has not been verified.

It was decided to collect the sample data on the real equipment because the model did not correspond so well to the real process. The system of equations was solved by a program written in Mathematica, see further appendix C. The program ran on a SUN Sparc2 workstation and required about 40 hours CPU-time to solve the problem.

The performance of the resulting controller was very good. By visual observations it was very hard to recognize any difference between the fuzzy and the digital controller.

5.3 Experiments with few partitions of the state variables

The first attempt was to consider the simplified problem of only balancing the pendulum. This was simulated using the same linearized model as earlier.
To model a digital controller should be fairly simple, due to it's linearity. It was therefore decided to experiment with the following membership functions as shown in Figure 5.3. Note, that two adjacent

fuzzy sets always have to overlap each other and therefore the two fuzzy sets, Positive and Negative, in Figure 5.3b are designed as they are.

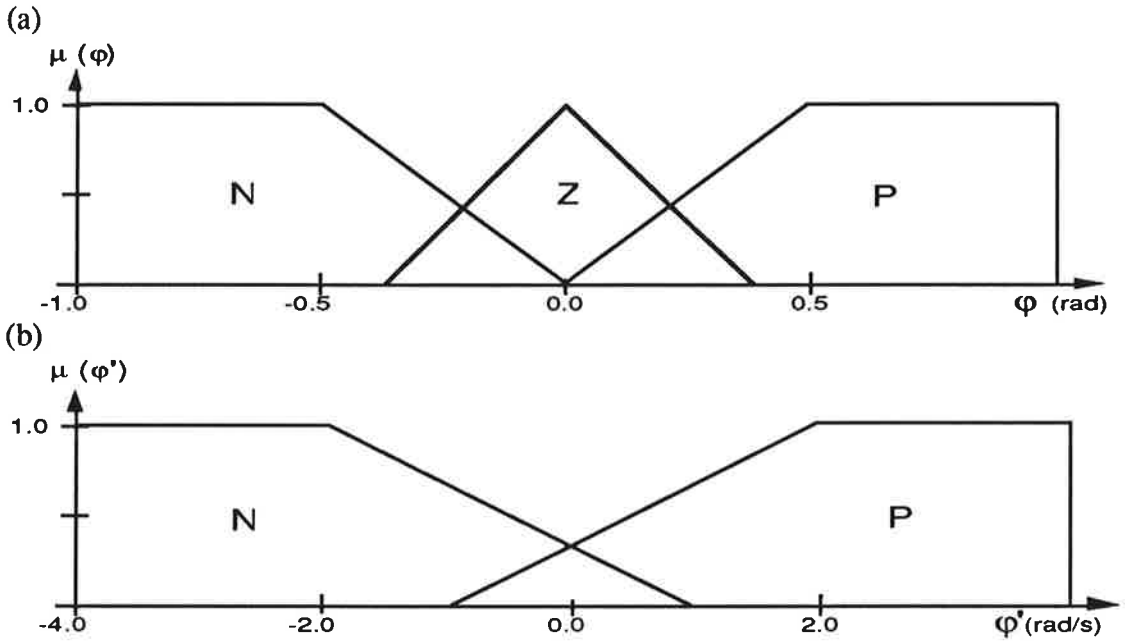


Figure 5.3. Membership functions for (a) angle and (b) angular velocity, for the simplified problem to only balance the pendulum.

This partition implied a rule base of 6 rules:

(R_1)	**IF**	(φ is N)	**AND**	(φ' is N)	**THEN**	y_1
(R_2)	**IF**	(φ is N)	**AND**	(φ' is P)	**THEN**	y_2
(R_3)	**IF**	(φ is Z)	**AND**	(φ' is N)	**THEN**	y_3
(R_4)	**IF**	(φ is Z)	**AND**	(φ' is P)	**THEN**	y_4
(R_5)	**IF**	(φ is P)	**AND**	(φ' is N)	**THEN**	y_5
(R_6)	**IF**	(φ is P)	**AND**	(φ' is P)	**THEN**	y_6

The corresponding coefficients to this rule base look as in Table 6.1:

	p_0	p_1	p_2
(R_1)	-0.00134781	42.4016	5.28934
(R_2)	-0.00307395	42.4039	5.29078
(R_3)	-0.00300253	42.3918	5.29023
(R_4)	-0.00126861	42.3975	5.28978
(R_5)	0.00182227	42.3931	5.29048
(R_6)	0.00301682	42.4068	5.28933

Table 6.1. Identified coefficents for the fuzzy controller with 6 rules

Note that all p_{0j}, p_{1j} and p_{2j}, respectively, are essentially the same. Let us say they are all totally identical, i.e:

$$p_{0j}= p_0 ,\ p_{1j}= p_1 ,\ p_{2j}= p_2 ,\ j=1, 2, ..., 6$$

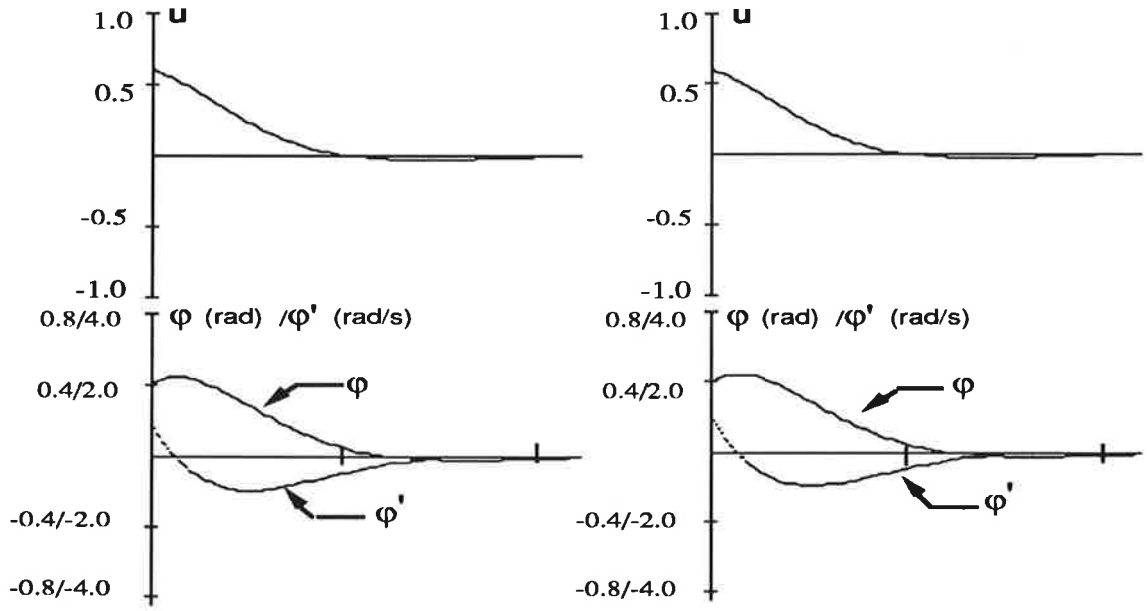
That implies that the output y can be calculated as below:

$$y = \frac{w_1*y_1 + w_2*y_2 +....+ w_6*y_6}{\Sigma_{i=1}^{6} w_i} = [\ y_1=y_2=...y_6=p_0+p_1*\varphi+p_2*\varphi'\ ] =$$

$$= \frac{w_1*(p_0+p_1*\varphi+p_2*\varphi') + w_2*(p_0+p_1*\varphi+p_2*\varphi') +....+ w_6*(p_0+p_1*\varphi+p_2*\varphi')}{\Sigma_{i=1}^{6} w_i} =$$

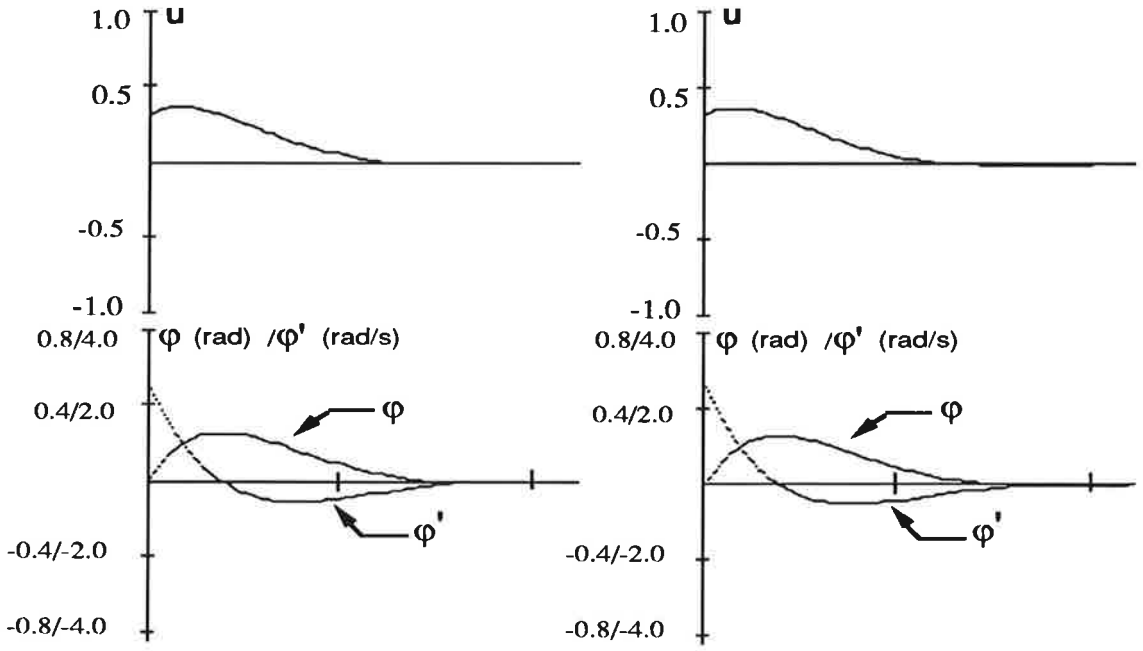
$$= \frac{p_0*\Sigma_{i=1}^{6} w_i+p_1*\varphi*\Sigma_{i=1}^{6} w_i +p_2*\varphi'*\Sigma_{i=1}^{6} w_i}{\Sigma_{i=1}^{6} w_i} = p_0 + p_1*\varphi + p_2*\varphi'$$

This result shows that the *weighted linear curve-fitting* technique has almost perfectly modelled the digital controller. The performance of the controller is illustrated by two simulations in Figure 5.4 and Figure 5.5.



(a) Digital controller (b) Fuzzy controller with '6 rules'

Figure 5. 4. Performance comparison of (a) digital and (b) fuzzy controller with initial values for φ_0=0.4 rad and φ_0'=0.8 rad/s.



(a) Digital controller (b) Fuzzy controller with '6 rules'

Figure 5. 5. Performance comparison of (a) digital and (b) fuzzy controller with initial values for φ_0=0.0 rad and φ_0'=2.2 rad/s.

The fuzzy '6 rules controller performs, of course, very well compared to the digital controller. Small discrepancies could be observed between the two controllers in some extreme situations, but this is probably due to rounding errors.

With this successful results to balance the pendulum it seemed plausible that the real problem, i.e balancing and positioning, could be solved in the same simple manner. An experiment was conducted where all 4 states were partitioned in 2 divisions, as was done for φ' in Figure 5.3b. The resulting rule base consisted therefore only of 16 rules.

Contrary to our expectation, this approach failed completely and the reason why this happened cannot be verified. Maybe some of the states should have been partitioned in 3 divisions instead of only 2. Another more plausible theory is that the shape of the membership functions was badly chosen. The top parts of the membership functions for positive and negative are flat and therefore information is lost in these ranges. To attempt to use fuzzy subsets that do not have this flat top for at least x and x' would be worthwhile. The problem to consider only balancing the pendulum was indeed solved using 'flat' membership functions, but to overview what will happen in different situations is also much simpler for this problem. In Figure 5.3 it can be seen that the plateaus start for φ equals +0.5 or -0.5 radians and φ' equals +2.0 or -2.0

radians/second. These ranges were chosen because it was regarded that if the angle or the angular velocity exceeded these values the controller output would in most cases be very large or saturated. But to anticipate the controller output when also the position is considered is in most cases impossible.

How big influence the shapes of the membership functions have when *weighted linear curve-fitting* is used is an intricate question. Information is lost when the membership function has a flat top, but the dynamics of a rule does not only depend on the degree of fulfillment but also on the numeric values of each state varible. This is a great difference when compared to the traditional inference methods, such as max-dot composition. This means also that not so many rules should be required with weighted linear curve-fitting.

5.4 Experimental Results

In this chapter 2 different approaches to design a fuzzy logic controller for the inverted pendulum problem, with weighted linear curve-fitting tachnique, have been investigated. It was first realized that some trick was needed to reduce the number of rules drastically.

The first way to achieve this was to divide the rule base in two rule bases, one for balancing and one for positioning. This leads to a rule base with 50 rules, 25 rules for balancing and 25 rules for positioning. This design technique does not exactly correspond to the theory of weighted linear curve-fitting because each rule has only two premises but the conclusion is inferred from all 4 states. The resulting controller was, however, run on the real equipment and performed very well. No real discrepancy could be observed between the fuzzy and the digital controller. The throughput time for the fuzzy controller was approximately 2 ms.

The second technique to decrease the number of rules was simply to partition the different states in fewer divisions. Successful experiments only considering balancing the pendulum were conducted with a rule base of only 6 rules. But the experiments for solving the real problem, with both balancing and positioning, failed for unknown reasons. Unfortunately little time was left to investigate why this happened. However, this approach is probably the best candidate for solving the inverted pendulum problem.

6. Conclusions

The purpose of this project was to investigate the possibility of applying fuzzy logic to the control of an inverted pendulum process. Two different techniques were investigated to design the fuzzy controller.

The first approach is based on max-dot composition as inference method. A main drawback with this inference method is that the states have to be partitioned in many divisions, which leads to a large rule base. This drawback was handled by decomposing the problem into simpler subtasks. The resulting controller performed similar to the digital controller, both on simulation and and on the real equipment, but this was achieved after a considerable amount of working hours. The tuning problems are due to the fact that the controller is very sensitive to changes of membership functions, and it is hard to express intuitively what changes are improving or deteriorating it's performance. There is an automated way to do this work, using for example a genetic algorithm [4]. This sounds very promising, but it has not been investigated in this project.

The second approach, i.e., *weighted linear curve-fitting*, attempts to model expert's control actions. In this case a previously designed digital controller was used as an expert. The output for each rule is inferred as a linear function of the different state variables. The final output is then generated as a weighted average of all rules, where the different weights are the degrees to which the rules have been fulfilled. The different coefficients, in the linear functions mentioned above, are obtained by first collecting input-output data of the digital controller and then by solving a system of equations using weighted linear regression. The computation time of these calculations depends on the number of rules and thus the rule base shall be kept as small as possible. This was achieved in two ways. The first attempt was to divide the problem into two different subtasks. This method does not strictly correspond to the theory of *weighted linear curve-fitting*, but the controller performed very well. The second experiment was to use very few partitions for each state variable. A fuzzy controller consisting of only 6 rules was successfully applied to the simplified problem of only balancing the pendulum without positioning. For the real problem of both balancing and positioning a fuzzy controller with 16 rules was designed, but this approach failed completely. Probably because the shapes of the membership functions were chosen badly.

To use max-dot composition as inference method seems most suitable for simple processes with less than say 50 rules. To achieve a good result in a short period of time the controller design should be done using some tuning algorithm for the membership functions.

If expert's actions are known, the weighted linear curve-fitting technique can be applied within a predictable time. This technique implies that the partitions of each state do not have to be as many as when using the max-dot composition. Furthermore, the troublesome decision-making for the rules conclusions can be readily eliminated.

7. References

1. Andersson U. Implementation of a fuzzy controller, CODEN: LUTFD2/(TFRT-5415)/1-56/(1989).

2. Grolimund L. and Häutle H. Investigating the robustness of fuzzy rules, Semester Project, ETH Zürich, 1990. (in German)

3. Holmblad L.P. and Ostergaard J.J. Control of a cement kiln by Fuzzy Logic, Fuzzy Information and Decision Processes, M.M Gupta Sanchez eds. North Holland, Amsterdam, 1982.

4. Karr C. Genetic algorithms for fuzzy controllers, AI Expert, February 1991, 26-33.

5. Kickert W.J.M and Lemke H.R. Application of a Fuzzy Controller in a Warm Water Plant, Automatica, vol 12, 1976, 301-308.

6. King P.J. and Mamdani E.H. The application of Fuzzy control systems to Industrial processes, *Automatica*, 13, 1977, 235-242.

7. Klir J.G. and Folger T.A. Fuzzy Sets, Uncertainty, and Information, Prentice Hall,Englewood Cliffs, New Jersey, 1988.

8. Mamdani M. and Assilian S. A fuzzy logic controller for a dynamic plant, International Journal of Man-Machine Studies, vol 7, 1975, 1-13.

9. Pedrycs W. Fuzzy Control and Fuzzy Systems, John Wiley & Sons Inc.,1989.

10. Sugeno M. and Nishida M. Fuzzy control of a model car, Fuzzy Sets and Systems, 16, 1985, 103-113.

11. Takagi T. and Sugeno M. Derivation of fuzzy control rules from human operator's control actions, IFAC Fuzzy Information, Marseille, France, 1983.

12. Togai InfraLogic. TILShell User's manual, Togai InfraLogic, California, 1990.

13. Tong R.M. A Control Engineering Review of Fuzzy Systems, *Automatica*, 13, 1977, 559-569.

14. Zadeh L.A. Fuzzy Sets, *Information and Control*, 1965, 338-353.

15. Zadeh L.A. Making computers think like people, *IEEE Spectrum*, August 1984, 26-32.

16. Zadeh L.A. Fuzzy Logic, *IEEE Computer Magazine*, 21, 4 (1988), 83-88.

17. Åström K.J. Intelligent Control, Lund instute of technology, Box 118, S-221 00 Lund, Sweden

8. Appendices

Appendix A

Theory of linear regression

According to theory of weighted linear curve-fitting the output, y, from the fuzzy controller, of the inverted pendulum, will be inferred as below:

$$y = \frac{1}{\sum_{r=1}^{l} w^r}[w^1 \times (p_0^1 \times g_0 + p_1^1 \times g_1 + ... + p_4^1 \times g_4) + w^2 \times (p_0^2 \times g_0 + p_1^2 \times g_1 + ... + p_4^2 \times g_4) + ... + w^l \times (p_0^l \times g_0 + p_1^l \times g_1 + ... + p_4^l \times g_4)]$$

$g_0 = 1, \quad g_1 = x, \quad g_2 = x', \quad g_3 = \varphi, \quad g_4 = \varphi'$

l is the number of rules for the controller and w^r is the degree to which rule number 'r' has been fulfilled.

Introduce:

$$\delta_i = y_i - u_i$$

where y_i is the output from the fuzzy controller and u_i is the expert's control action.

Linear regression implies to minimize the error-function below:

$$\sum_{i=0}^{n} (y_i - u_i)^2 = \sum_{i=0}^{n} (\delta_i)^2$$

n is the number of collected sample data

The minimization of this error-function is achieved when all derivatives equals zero:

$$\frac{\partial}{\partial p_j^r}\left(\sum_{i=0}^{n}(\delta_i)^2\right)=0 \qquad j=0,1,\ldots,4 \qquad r=1,2,\ldots,1$$

$$\frac{\partial}{\partial p_j^r}\left(\sum_{i=0}^{n}(\delta_i)^2\right)=\sum_{i=0}^{n}\frac{\partial}{\partial p_j^r}(\delta_i)^2=\sum_{i=0}^{n}2\times\delta_i\times\frac{\partial\delta_i}{\partial p_j^r}=0 \qquad (A)$$

$$\frac{\partial\delta_i}{\partial p_j^r}=0+0+\ldots+\frac{\partial}{\partial p_j^r}\left(\frac{w^r\times p_j^r\times g_j}{\sum_{r=1}^{l}w^r}\right)+0+\ldots+0=\frac{w^r\times g_j}{\sum_{r=1}^{l}w^r} \qquad (B)$$

$$(A),\ (B)\Rightarrow \quad \sum_{i=0}^{n}[y_i-u_i]\times\frac{w^r\times g_j}{\sum_{r=1}^{l}w^r}=0 \qquad \Leftrightarrow$$

$$\sum_{i=0}^{n}\left(\left[\frac{1}{\sum_{r=1}^{l}w^r}\left[w^1\times(p_0^1\times g_0+p_1^1\times g_1+\ldots.+p_4^1\times g_4)+w^2\times(p_0^2\times g_0+p_1^2\times g_1+\ldots.+p_4^2\times g_4)+\right.\right.\right.$$

$$\left.\left.\left.\ldots.+w^l\times(p_0^l\times g_0+p_1^l\times g_1+\ldots.+p_4^l\times g_4)\right]-u_i\right]\times\frac{w^r\times g_j}{\sum_{r=1}^{l}w^r}\right)=0 \qquad \begin{matrix}\text{for } j=0,1,..,4\\ \text{and } r=1,2,..,1\end{matrix}$$

$$\sum_{i=0}^{n}\frac{w^r\times g_j}{\left(\sum_{r=1}^{l}w^r\right)^2}\times\left[w^1\times(p_0^1\times g_0+p_1^1\times g_1+\ldots+p_4^1\times g_4)+w^2\times(p_0^2\times g_0+p_1^2\times g_1+\right.$$

$$\left.\ldots+p_4^2\times g_4)+\ldots+w^l\times(p_0^l\times g_0+p_1^l\times g_1+\ldots+p_4^l\times g_4)\right]=\sum_{i=0}^{n}\frac{u_i\times w^r\times g_j}{\sum_{r=1}^{l}w^r}$$

This is now a system of linear equations where the desired coefficients can be solved.

Appendix B

Implementing weighted linear curve-fitting using TIL shell

To implement this technique using TIL Shell required some consideration, because TIL shell does not provide a direct access to the degree to which a rule has been fulfilled.
The way this has been solved is by using a trick. First, it was observed that in the generated C-code a variable labeled alpha was assigned the degree of belief for each rule.
Then fragments were declared in the fuzzy rule base:

```
/* Rule 0 */
.....

/* Fragment 0 */

Deg_Belief [0] = alpha;

.....
...

/* Rule n */
.....

/* Fragment n */

Deg_Belief [n] = alpha;
```

The variable Deg_Belief was declared as an output from the fuzzy rule base and this meant that in the C-code it got declared as an external variable and could therefore be accessed from outside TIL Shell.

Appendix C

Mathematica program code used for solving the system of linear equations in chapter 5

```
(*-------------------------------------*)
(* Weighted linear regression analysis *)
(* for the fuzzy controller of an      *)
(* inverted pendulum.                  *)
(*                                     *)
(*      Version 14-May-1991; CRBC3; Su *)
(*-------------------------------------*)

(* Step 1: Read in angle data       *)
(* -------------------------------- *)
(*   In: {phi,phidot,x,xDot,u,w1,... ,w50} *)

dataFile = "Traj_50Data.dat"; (* Read in data table *)
maxRule = 50;
maxFVar = 5;    (* Input and output fuzzy variables *)
maxPoli = 5;    (* Number of linear polinomials g1, g2, .. *)

OpenRead[dataFile];
ATab = ReadList[dataFile,Table[Number,{maxFVar+maxRule}]];
Close[dataFile];

(*----- Compute total weight *)
Wi = Table[ Sum[ ATab[[i,maxFVar+r]], {r,maxRule}],
            {i,Length[ATab]}
          ];
Wi2 = Table[ Wi[[i]]^2, {i,Length[ATab]}];

(* Step 2: Compute matrix b         *)
(* -------------------------------- *)
(*          [m] [p] = [b]           *)

(*-- 7(w*u/Wi){1,phi,phiDot,x,xDot} *)
bSum[r_] :=
   Sum[ (ATab[[i,r]] ATab[[i,5]] / Wi[[i]])
            {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
        {i, Length[ATab]}
      ];

b = Flatten[ Table[ bSum[maxFVar+r], {r,maxRule}]

           ];

(* Step 3: Compute matrix m         *)
(* -------------------------------- *)
(*          [m] [p] = [b]           *)

(*----- 7(wr*ws/Wi^2){ {1,phi,phiDot,x,xDot} *)
(*-----              phi{1,phi,phiDot,x,xDot} *)
(*-----           phiDot{1,phi,phiDot,x,xDot} *)
(*-----                x{1,phi,phiDot,x,xDot} *)
(*-----             xDot{1,phi,phiDot,x,xDot} *)
```

```
mSum[r_,s_] :=
     Sum[ (ATab[[i,r]] ATab[[i,s]] / Wi2[[i]])
 {             {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,1]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,2]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,3]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,4]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]}
 },
          {i, Length[ATab]}
        ];

m = Flatten[
      Table[ Transpose[
               Table[ mSum[maxFVar+r,maxFVar+s], {s,maxRule}]],
             {r,maxRule}
           ]
      ,1];

Do[ m[[i]]=Flatten[m[[i]]] , {i,Length[m]}];

outFile = "matrix.M";
OpenWrite[ outFile, FormatType -> OutputForm];
Write[ outFile, TableForm[ m]];
Close[ outFile];

(* Step 4: Solve a system of r*j    *)
(*         linear equations         *)
(* -------------------------------- *)
(*     {r,1,maxRule}, {j,1,maxPoli} *)

p = Flatten[
       Table[ a[r,j], {r,maxRule}, {j,maxPoli}], 1];

Solution = Solve[ m . p == b, p];

Ps = Partition[ Flatten[ p /. Solution, 1], maxPoli];
     (* Transform it *)

outFile = "matrix.P";
OpenWrite[ outFile, FormatType -> OutputForm];
  Write[ outFile, TableForm[ Ps]];
Close[ outFile];

(* Step 5: Generate a table with    *)
(*         weighted linear interpo- *)
(*         rations wu               *)
(* -------------------------------- *)
(*  {{ i, phi, phiDot, x, xDot, u, wu}, ... }*)

u[ i_] :=
 { i, ATab[[i,1]], ATab[[i,2]], ATab[[i,3]], ATab[[i,4]],
   ATab[[i,5]],
   Sum[ ( ATab[[i,maxRule+maxFVar]]
( Ps[[r,1]] + Ps[[r,2]] ATab[[i,1]] + Ps[[r,3]] ATab[[i,2]]
            + Ps[[r,4]] ATab[[i,3]] + Ps[[r,5]] ATab[[i,4]])
        ), { r, maxRule}
      ] / Wi[[i]]
 };

CompTab = Table[ u[i], {i,Length[ATab]}];
```

```
outFile = "Traj_50Data.CMP";
OpenWrite[ outFile, FormatType -> OutputForm];
  Write[ outFile, TableForm[ CompTab]];
Close[ outFile]
```

Appendix C

Mathematica program code used for solving the system of linear equations in chapter 5

```
(*-------------------------------------*)
(* Weighted linear regression analysis *)
(* for the fuzzy controller of an      *)
(* inverted pendulum.                  *)
(*                                     *)
(*      Version 14-May-1991; CRBC3; Su *)
(*-------------------------------------*)

(* Step 1: Read in angle data       *)
(* -------------------------------- *)
(*   In: {phi,phidot,x,xDot,u,w1,... ,w50} *)

dataFile = "Traj_50Data.dat"; (* Read in data table *)
maxRule = 50;
maxFVar = 5;    (* Input and output fuzzy variables *)
maxPoli = 5;    (* Number of linear polinomials g1, g2, .. *)

OpenRead[dataFile];
ATab = ReadList[dataFile,Table[Number,{maxFVar+maxRule}]];
Close[dataFile];

(*----- Compute total weight *)
Wi = Table[ Sum[ ATab[[i,maxFVar+r]], {r,maxRule}],
            {i,Length[ATab]}
          ];
Wi2 = Table[ Wi[[i]]^2, {i,Length[ATab]}];

(* Step 2: Compute matrix b         *)
(* -------------------------------- *)
(*         [m] [p] = [b]            *)

(*-- 7(w*u/Wi){1,phi,phiDot,x,xDot} *)
bSum[r_] :=
   Sum[ (ATab[[i,r]] ATab[[i,5]] / Wi[[i]])
            {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
        {i, Length[ATab]}
      ];

b = Flatten[ Table[ bSum[maxFVar+r], {r,maxRule}]

           ];

(* Step 3: Compute matrix m         *)
(* -------------------------------- *)
(*         [m] [p] = [b]            *)

(*----- 7(wr*ws/Wi^2){ {1,phi,phiDot,x,xDot} *)
(*-----              phi{1,phi,phiDot,x,xDot} *)
(*-----           phiDot{1,phi,phiDot,x,xDot} *)
(*-----                x{1,phi,phiDot,x,xDot} *)
(*-----             xDot{1,phi,phiDot,x,xDot} *)
```

```
mSum[r_,s_] :=
     Sum[ (ATab[[i,r]] ATab[[i,s]] / Wi2[[i]])
 {             {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,1]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,2]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,3]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]},
   ATab[[i,4]] {1,ATab[[i,1]],ATab[[i,2]],ATab[[i,3]],ATab[[i,4]]}
 },
          {i, Length[ATab]}
        ];

m = Flatten[
      Table[ Transpose[
               Table[ mSum[maxFVar+r,maxFVar+s], {s,maxRule}]],
             {r,maxRule}
           ]
      ,1];

Do[ m[[i]]=Flatten[m[[i]]] , {i,Length[m]}];

outFile = "matrix.M";
OpenWrite[ outFile, FormatType -> OutputForm];
Write[ outFile, TableForm[ m]];
Close[ outFile];

(* Step 4: Solve a system of r*j    *)
(*         linear equations         *)
(* -------------------------------- *)
(*     {r,1,maxRule}, {j,1,maxPoli} *)

p = Flatten[
       Table[ a[r,j], {r,maxRule}, {j,maxPoli}], 1];

Solution = Solve[ m . p == b, p];

Ps = Partition[ Flatten[ p /. Solution, 1], maxPoli];
     (* Transform it *)

outFile = "matrix.P";
OpenWrite[ outFile, FormatType -> OutputForm];
  Write[ outFile, TableForm[ Ps]];
Close[ outFile];

(* Step 5: Generate a table with    *)
(*         weighted linear interpo- *)
(*         rations wu               *)
(* -------------------------------- *)
(*  {{ i, phi, phiDot, x, xDot, u, wu}, ... }*)

u[ i_] :=
 { i, ATab[[i,1]], ATab[[i,2]], ATab[[i,3]], ATab[[i,4]],
   ATab[[i,5]],
   Sum[ ( ATab[[i,maxRule+maxFVar]]
( Ps[[r,1]] + Ps[[r,2]] ATab[[i,1]] + Ps[[r,3]] ATab[[i,2]]
            + Ps[[r,4]] ATab[[i,3]] + Ps[[r,5]] ATab[[i,4]])
       ), { r, maxRule}
     ] / Wi[[i]]
 };

CompTab = Table[ u[i], {i,Length[ATab]}];
```

```
outFile = "Traj_50Data.CMP";
OpenWrite[ outFile, FormatType -> OutputForm];
  Write[ outFile, TableForm[ CompTab]];
Close[ outFile]
```