Fuzzy Logic for G Toolkit Reference Manual

March 1997 Edition
Part Number 321511A-01

Internet Support

`support@natinst.com`
E-mail: `info@natinst.com`
FTP Site: `ftp.natinst.com`
Web Address: `http://www.natinst.com`

Bulletin Board Support

BBS United States: (512) 794-5422
BBS United Kingdom: 01635 551422
BBS France: 01 48 65 15 59

Fax-on-Demand Support

(512) 418-1111

Telephone Support (U.S.)

Tel: (512) 795-8248
Fax: (512) 794-5678

International Offices

Australia 02 9874 4100, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 09 527 2321, France 01 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186,
Israel 03 5734815, Italy 06 5729961, Japan 03 5472 2970, Korea 02 596 7456,
Mexico 5 520 2635, Netherlands 31 348 43 34 66, Norway 32 84 84 00, Singapore 2265886,
Spain 91 640 0085, Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200,
U.K. 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway   Austin, TX 78730-5039   Tel: (512) 794-0100

Important Information

Contents

Figures

Tables

About This Manual

The *Fuzzy Logic for G Toolkit Reference Manual* describes the features, functions, and operation of the Fuzzy Logic Toolkit. You can use this toolkit to design and implement rule-based fuzzy logic systems for process control or expert decision making. To use this manual effectively, you should be familiar with basic control theory. Knowledge of rule-based systems and fuzzy logic is helpful, but not absolutely necessary.

Organization of This Manual

The *Fuzzy Logic for G Toolkit Reference Manual* is organized as follows:

- Chapter 1, *Introduction*, introduces the Fuzzy Logic for G Toolkit. It contains system configuration information, installation instructions, and basic information that explains how to start using this toolkit.
- Chapter 2, *Overview of Fuzzy Logic*, introduces fuzzy set theory and provides an overview of fuzzy logic control.
- Chapter 3, *Fuzzy Controllers*, describes various implementations and Input/Output (I/O) characteristics of fuzzy controllers.
- Chapter 4, *Design Methodology*, provides an overview of the design methodology of a fuzzy controller.
- Chapter 5, *Using the Fuzzy Logic Controller Design VI*, describes how to design a fuzzy controller using the Fuzzy Logic Controller Design VI.
- Chapter 6, *Implementing a Fuzzy Controller*, describes how to implement a fuzzy controller and includes a pattern recognition application example.
- Chapter 7, *Fuzzy Logic VI Descriptions*, contains descriptions of the fuzzy logic VIs.

- Appendix A, *References*, lists the reference material used to produce the VIs in this manual. These references contain more information on the theory and algorithms implemented in the fuzzy logic VIs.
- Appendix B, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.
- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.
- The *Index* contains an alphabetical list of key terms and topics in this manual, including the page where you can find each one.

Conventions Used in This Manual

The following conventions are used in this manual:

bold	Bold text denotes a parameter, menu name, palette name, menu item, return value, function panel item, icon name, or dialog box button or option.
italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept.
bold italic	Bold italic text denotes an activity objective, note, caution, or warning.
`monospace`	Text in this font denotes text or characters that you should literally enter from the keyboard. Sections of code, programming examples, and syntax examples also appear in this font. This font also is used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, filenames, and extensions, and for statements and comments taken from program code.
<>	Angle brackets enclose the name of a key on the keyboard—for example, <PageDown>.
-	A hyphen between two or more key names enclosed in angle brackets denotes that you should simultaneously press the named keys—for example, <Control-Alt-Delete> for Windows.
<Control>	Key names are capitalized.

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options»Substitute Fonts** directs you to pull down the **File** menu, select the **Page Setup** item, select **Options**, and finally select the **Substitute Fonts** option from the last dialog box.
paths	Paths in this manual are denoted using backslashes (\) to separate drive names, directories, and files, as in `C:\dir1name\dir2name\filename`.
☞	This icon to the left of bold italicized text denotes a note, which alerts you to important information.

Abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms are listed in the *Glossary*.

Related Documentation

The following documents contain information you might find helpful as you read this manual:

- *LabVIEW User Manual*
- *LabVIEW Tutorial*
- *BridgeVIEW User Manual*
- *G Programming Reference Manual*

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and configuration forms for you to complete. These forms are in Appendix B, *Customer Communication*, at the end of this manual.

Chapter 1

Introduction

This chapter introduces the Fuzzy Logic for G Toolkit. It contains system configuration information, installation instructions, and basic information that explains how to start using this toolkit. This chapter refers you to other chapters for more information.

Your Fuzzy Logic for G Toolkit contains the following materials:

- The Fuzzy Logic Toolkit disks
- *Fuzzy Logic for G Toolkit Reference Manual*

Required System Configuration

You must have LabVIEW or BridgeVIEW to use the Fuzzy Logic Toolkit. System requirements are the same as those for LabVIEW or BridgeVIEW. You might need one or more DAQ hardware devices to implement process control of a physical system.

Installation

The following sections contain instructions for installing the Fuzzy Logic for G Toolkit on the Windows 95, Windows NT, Windows 3.*x*, Macintosh and Power Macintosh platforms.

Windows 95 and Windows NT

Complete the following steps to install the toolkit.

1. Launch Windows 95.
2. Insert disk 1 of the Fuzzy Logic for G Toolkit into the 3.5-inch disk drive.
3. From the **Start** menu, choose **Run** and enter `A:\setup.exe`.
4. Follow the instructions on your screen.

Once you have completed the on-screen installation instructions, you are ready to run the Fuzzy Logic Controller Design VI.

Windows 3.*x*

Complete the following steps to install the toolkit.

1. Launch Windows.
2. Insert disk 1 of the Fuzzy Logic for G Toolkit into the 3.5-inch disk drive.
3. From the File Manager, run `SETUP.EXE`.
4. Follow the instructions on your screen.

Once you have completed the on-screen installation instructions, you are ready to run the Fuzzy Logic Controller Design VI.

Macintosh and Power Macintosh

Complete the following steps to install the toolkit.

1. Insert disk 1 of the Fuzzy Logic for G Toolkit into the 3.5-inch disk drive and double-click on the **Fuzzy Logic Installer** icon.
2. Follow the instructions on your screen.

Once you have completed the on-screen installation instructions, you are ready to run the Fuzzy Logic Controller Design VI.

Introduction to Fuzzy Logic

Fuzzy logic is a method of rule-based decision making used for expert systems and process control that emulates the *rule-of-thumb* thought process used by human beings.

Fuzzy logic can be used to control a process that a person can control manually with expertise gained from experience. The linguistic control rules that a human expert can describe in an intuitive and general manner can be directly translated to a rule base for a fuzzy logic controller. Chapter 2, *Overview of Fuzzy Logic*, contains a more detailed explanation of fuzzy logic.

How Does the Fuzzy Logic Toolkit Work?

With the Fuzzy Logic for G Toolkit, you can design a fuzzy logic controller (or expert system for decision making) and implement the controller in your G applications. Fuzzy membership functions and the controller rule base are defined with the Fuzzy Logic Controller Design VI. The Controller Design VI is a standalone VI with a graphical user interface for completely defining all controller/expert system components. All parameters of the defined controller are saved into a controller data file.

Two additional VIs are used to implement the fuzzy controller in your G application. The Load Fuzzy Controller VI is used to load all parameters of the fuzzy controller saved in a data file by the Controller Design VI. This data is then wired to the Fuzzy Controller VI, which implements the fuzzy logic inference engine. Process parameters (controller inputs) are wired to the inputs of the Fuzzy Controller VI, and controller outputs are output by the VI. By wiring data acquired by your data acquisition hardware to the fuzzy controller, you can implement real-time decision making or control of your physical system. Additionally, outputs of the fuzzy controller can be used by your data acquisition (DAQ) analog output hardware to implement real-time process control.

Where Should I Start?

If you are not familiar with fuzzy logic and rule-based control, read Chapter 2, *Overview of Fuzzy Logic*, Chapter 3, *Fuzzy Controllers*, and Chapter 4, *Design Methodology*, which provide an overview of typical methods of fuzzy controller design and implementation. Start with the following chapters if you are familiar with fuzzy logic but interested in learning more about fuzzy controllers: Chapter 5, *Using the Fuzzy Logic Controller Design VI*, describes how to use the toolkit to design a fuzzy controller and save the controller data to a file. Chapter 6, *Implementing a Fuzzy Controller*, describes how to use the remaining Toolkit VIs to implement the designed controller in your G applications. The toolkit VIs are explained in detail in Chapter 7, *Fuzzy Logic VI Descriptions*.

Chapter 2

Overview of Fuzzy Logic

This chapter introduces fuzzy set theory and provides an overview of fuzzy logic control.

What is Fuzzy Logic?

Fuzzy logic is a method of rule-based decision making used for expert systems and process control that emulates the rule-of-thumb thought process used by human beings. The basis of fuzzy logic is fuzzy set theory which was developed by Lotfi Zadeh in the 1960s. Fuzzy set theory differs from traditional Boolean (or two-valued) set theory in that partial membership in a set is allowed.

Traditional Boolean set theory is two-valued in the sense that a member either belongs to a set or does not—represented by 1 or 0, respectively. Fuzzy set theory allows for partial membership, or a degree of membership, which might be any value along the continuum of 0 to 1.

A linguistic term can be defined quantitatively by a type of fuzzy set known as a *membership function*. The membership function specifically defines degrees of membership based on a property such as temperature or pressure. With membership functions defined for controller or expert system inputs and outputs, you can formulate a rule base of IF-THEN type conditional rules. Such a rule base and the corresponding membership functions are employed to analyze controller inputs and determine controller outputs by the process of fuzzy logic inference.

By defining such a fuzzy controller, process control can be implemented quickly and easily. Many such systems are difficult or impossible to model mathematically, which is required for the design of most traditional control algorithms. In addition, many processes that might or might not be modeled mathematically are too complex or nonlinear to be controlled with traditional strategies. However, if a control strategy can be described qualitatively by an *expert*, fuzzy logic can be used to define a controller that emulates the heuristic rule-of-thumb strategies of the expert. Therefore, fuzzy logic can be used to control a process that a human can control manually with expertise gained from experience. The linguistic control rules that a human expert can describe in an intuitive and general manner can be directly translated to a rule base for a fuzzy logic controller.

Types of Uncertainty

Real world situations are often too uncertain or vague to describe precisely. Completely describing a complex situation requires more detailed data than a human being can recognize, process and understand.

When applying fuzzy logic concepts, there are three different types of uncertainty: stochastic, informal, and linguistic.

Stochastic uncertainty is the degree of uncertainty of the occurrence of a certain event. The event itself is well-defined, and the stochastic uncertainty is not related to when the event occurs. This type of uncertainty is used to describe only large-numbered phenomena.

Informal uncertainty results from a lack of information and knowledge about a situation.

Linguistic uncertainty results from the imprecision of language. *Much greater*, *too high* and *high fever* describe subjective categories with meanings that depend on the context in which they are used.

Modeling Linguistic Uncertainty with Fuzzy Sets

One of the basic concepts in fuzzy logic is the mathematical description of linguistic uncertainty using fuzzy sets. People often are forced to make decisions based on imprecise, subjective information. Even when the information does not contain precise quantitative elements, people can use fuzzy sets to manage complex situations successfully.

You do not need to have well-defined rules to make decisions. Most often, you can approximate with rules that cover only a few distinct cases and apply them to a given situation. This approximation is possible because of the flexibility of the rules.

For example, if the family doctor agrees to make a house call if a sick child has a high fever of 102° F, one definitely would summon the doctor when the thermometer reads 101.5° F.

This situation, however, cannot be modeled satisfactorily using conventional dual logic because the patient with a body temperature of 101.5° F does not fulfill the criterion for suffering from a high fever, and the doctor would not be called. A graphical representation of such a set is shown in Figure 2-1.

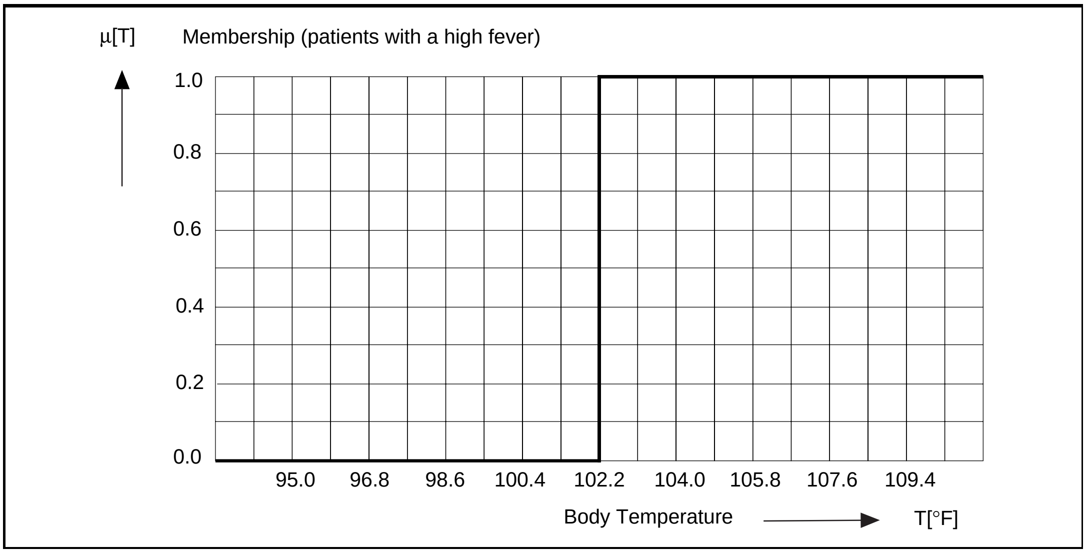


Figure 2-1. Modeling Uncertainty by Conventional Set Membership

Even if the body temperature was measured with an accuracy of up to five decimal places, the situation would be exactly the same. The higher precision does not change the fact that patients with a body temperature below 102° F do not fit into the category of patients with a high fever, while all patients with a body temperature of 102° F and higher fully belong to that category.

Modeling uncertain facts such as *high fever* sets aside the strict distinction between the two membership values one (TRUE) and zero (FALSE) and allows arbitrary intermediate membership degrees instead. With respect to conventional set theory, you can generalize the set notion by allowing elements to be "more-or-less" members of a certain set. This type of set is known as a fuzzy set. A graphical representation of such a set is shown in Figure 2-2.

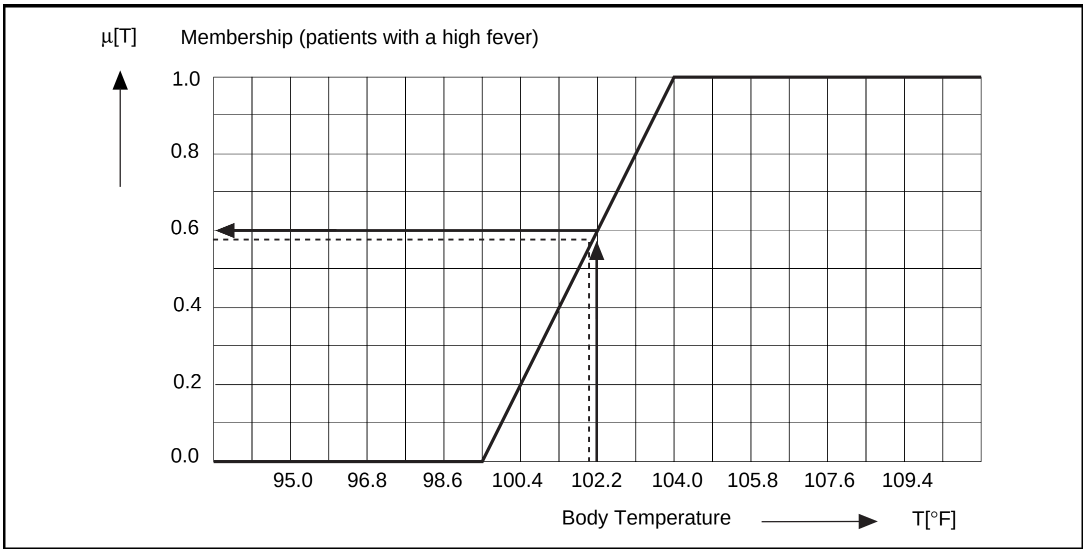


Figure 2-2. Modeling Uncertainty by Fuzzy Set Membership

In the illustration, each body temperature is associated with a certain degree of membership (μ(T)) to the *high fever* set. The function μ(T) is called degree of membership of the element (T ∈ BT) to the fuzzy set *high fever*. The body temperature is called *characteristic quantity* or *base variable* T of the universe BT. Notice that μ ranges from 0 to 1, the values representing absolutely no membership to the set and complete membership, respectively.

The degree of membership to the fuzzy set *high fever* also can be interpreted as degree of truth associated to the statement "the patient suffers from high fever." Thus, using fuzzy sets defined by membership functions within logical expressions leads to the notion *Fuzzy Logic*.

As shown in Figure 2-2 the degree of membership is represented by a continuous function μ(T) which often is called a fuzzy set. How to define membership functions for certain applications is discussed in the *Definition of Linguistic Variables* section of Chapter 4, *Design Methodology*.

Notice that a body temperature of 102° F is considered only slightly different from a body temperature of 101.5° F, and not considered a threshold.

Linguistic Variables and Terms

The primary building block of fuzzy logic systems is the linguistic variable. A linguistic variable is used to combine multiple subjective categories describing the same context. In the previous example, there is *high fever* and *raised* temperature as well as *normal* and *low* temperature in order to specify the uncertain and subjective category body temperature. These terms are called *linguistic terms* and represent the possible values of a *linguistic variable*. Each linguistic term is represented by a fuzzy set defined by a membership function.

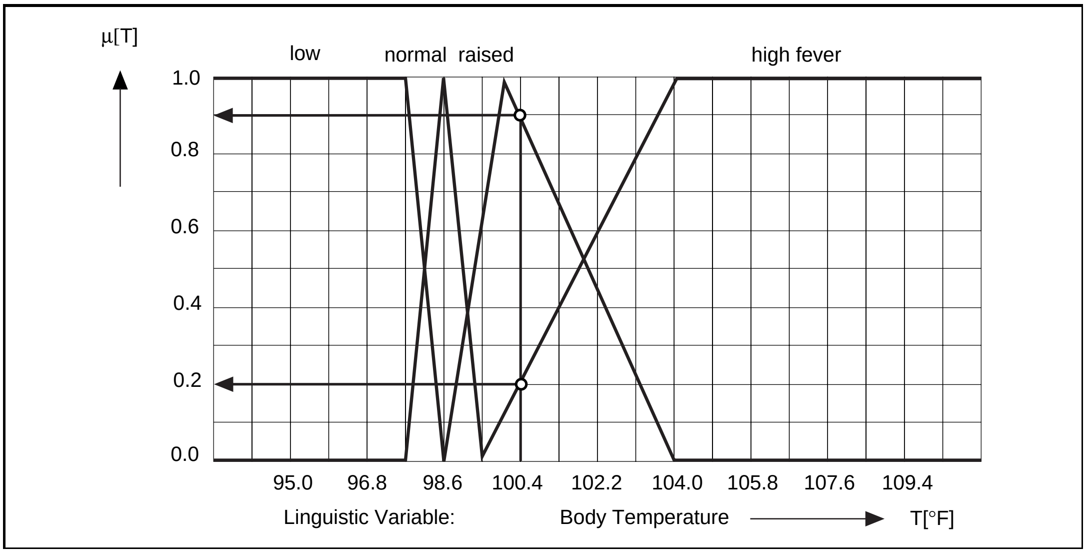


Figure 2-3. A Linguistic Variable Translates Real Values into Linguistic Values

The linguistic variable shown in Figure 2-3 allows for the translation of a crisp measured body temperature, given in degrees Fahrenheit, into its linguistic description. A body temperature of 100.5° F, for example, might be evaluated as a *raised* temperature, or a slightly *high fever*. The overlapping regions of neighboring linguistic terms are important when using linguistic variables to model engineering systems.

Rule-Based Systems

Another basic fuzzy logic concept involves rule-based decision-making processes. A detailed and precise mathematical description is not always necessary for optimized operation of an engineering process. In other words, human operators often are capable of managing complex situations of a plant without knowing anything about differential equations. Their engineering knowledge is perhaps available in a linguistic form such as “if the liquid temperature is correct, and the pH-value is too high, adjust the water feed to a higher level.”

Because of fully-developed nonlinearities, distributed parameters, and time constants that are difficult to determine, it is often impossible for a control engineer to develop a mathematical system model. With fuzzy logic, linguistic representation of engineering knowledge is used to implement a control strategy.

Suppose you must automate the maneuvering process leading a truck from an arbitrary starting point to a loading ramp. The truck should run at a constant low speed and stop immediately when it docks at the loading ramp. A human driver is capable of controlling the truck by constantly evaluating the current drive situation, mainly defined by the distance from the target position and the orientation of the truck, to derive the correct steering angle. This is shown in Figure 2-4.

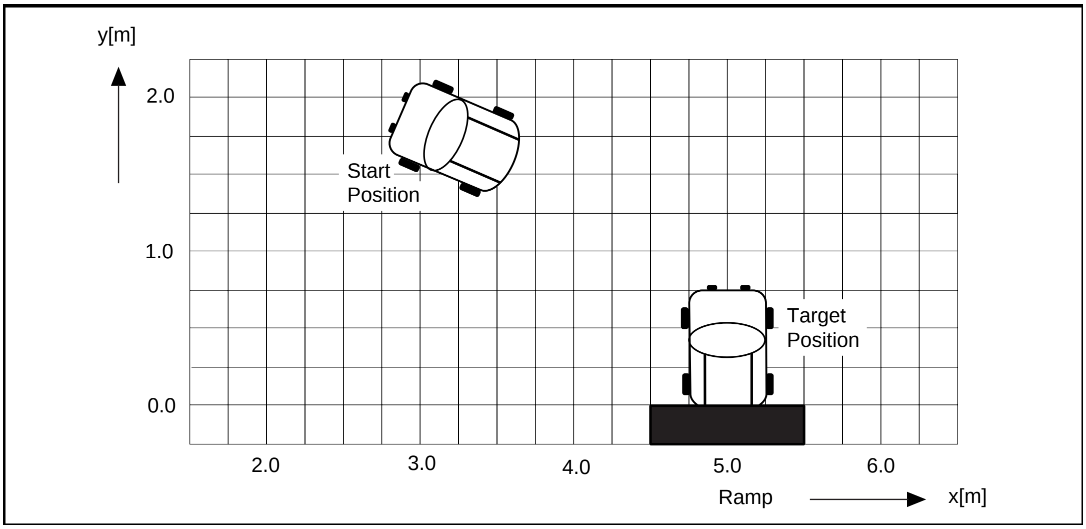


Figure 2-4. Automation of a Maneuvering Process Example

Implementing a Linguistic Control Strategy

To automate the truck control, an ultrasonic distance sensor monitors the truck position in *x*-direction, and an electronic compass monitors the truck orientation. Each drive situation is identified by at least two conditions: The first one describes the vehicle position *x* from the loading ramp, and the second condition describes the vehicle orientation β. The conditions are combined with the word AND, representing the fact that both conditions must be valid for the respective situation.

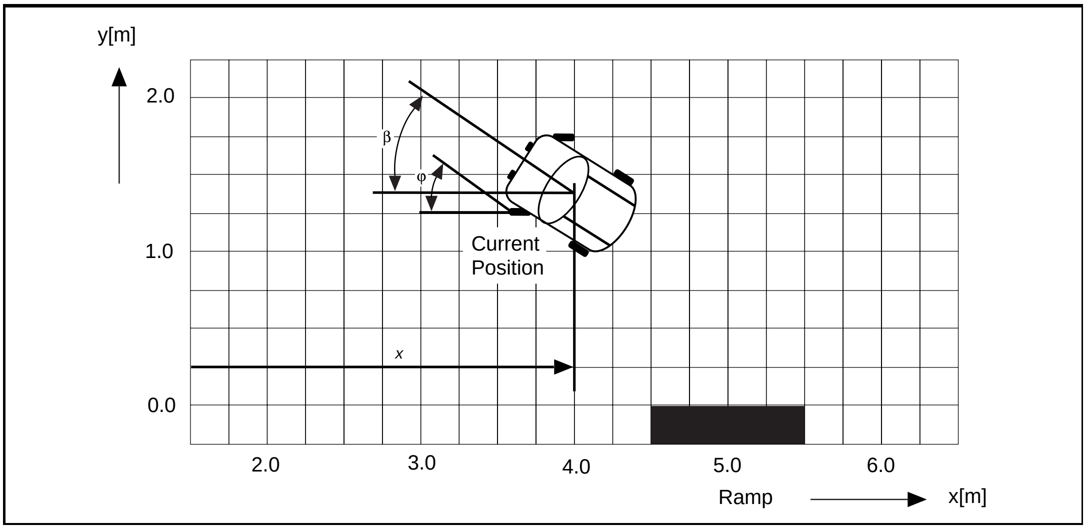


Figure 2-5. Condition: Vehicle Position *x* and Orientation β, Action: Steering Angle φ

The situation shown in Figure 2-5 describes a vehicle position left from the target center with a left-hand orientation β, and a large negative steering angle φ with the steering wheel turned all the way to the left.

A control strategy can be defined by using IF-THEN rules such as the following:

IF <situation> **THEN** <action>

The above rule format describes the necessary reaction, or conclusion, to a certain situation, or condition.

By asking an expert driver for advice about how to proceed when maneuvering the vehicle to the target position, you might learn some rules-of-thumb that can be described by the following IF-THEN rules:

IF vehicle position x is *left center* **AND** vehicle orientation β is *left up* **THEN** adjust steering angle φ to *positive small,*

or

IF vehicle position x is *center* **AND** vehicle orientation β is *left up* **THEN** adjust steering angle φ to *negative small,*

or

IF vehicle position x is *left center* **AND** vehicle orientation β is *up* **THEN** adjust steering angle φ to *positive medium,*

or

IF vehicle position x is *center* **AND** vehicle orientation β is *up* **THEN** adjust steering angle φ to *zero.*

☞ **Note:** ***The conditions of each rule are composed of uncertain linguistic terms like*** *left center*, *left up*, ***and so on. Even the conclusion of each rule contains vague and imprecise facts such as*** *negative small*. ***Because there are no precise definitions of the words used in the rules above, there is no way to implement them directly using IF-THEN statements from a conventional programming language.***

You can implement a linguistic control strategy using fuzzy logic, which is capable of modeling uncertain linguistic facts like *left center* or *high fever,* with fuzzy sets.

First, you must define a linguistic variable for each characteristic quantity of the maneuvering process. For example, **vehicle position *x*** and **vehicle orientation** β are process or input variables, and **steering angle** φ is an output variable.

A linguistic variable is made up of a number of linguistic terms describing the different linguistic interpretations of the characteristic quantity being modeled. Each linguistic term is defined again by an appropriate membership function (fuzzy set).

Figures 2-6, 2-7, and 2-8 show membership functions for the inputs and output of the truck controller.

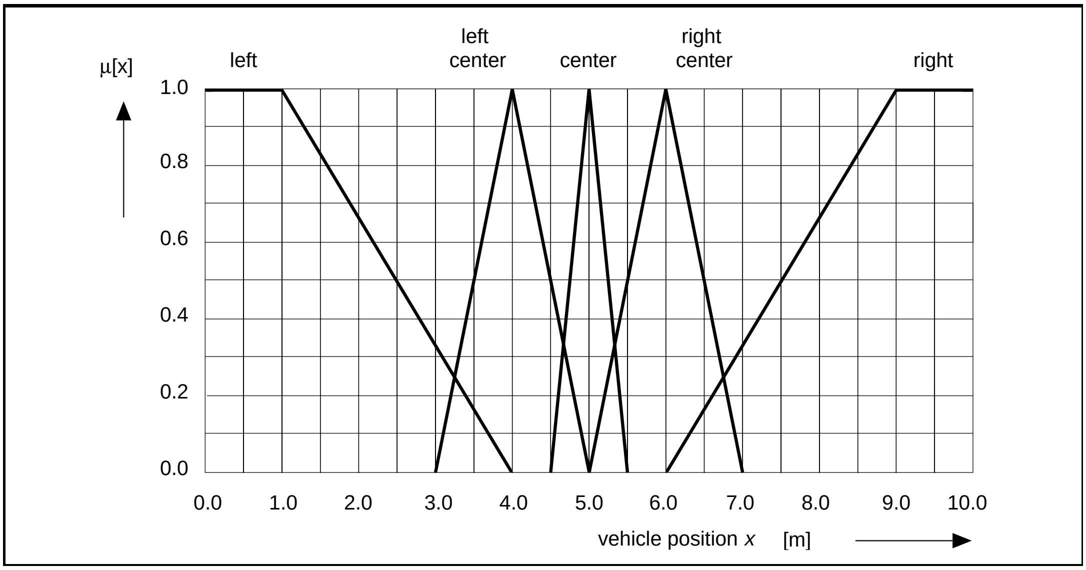


Figure 2-6. Linguistic Variable Vehicle Position *x* and Its Linguistic Terms

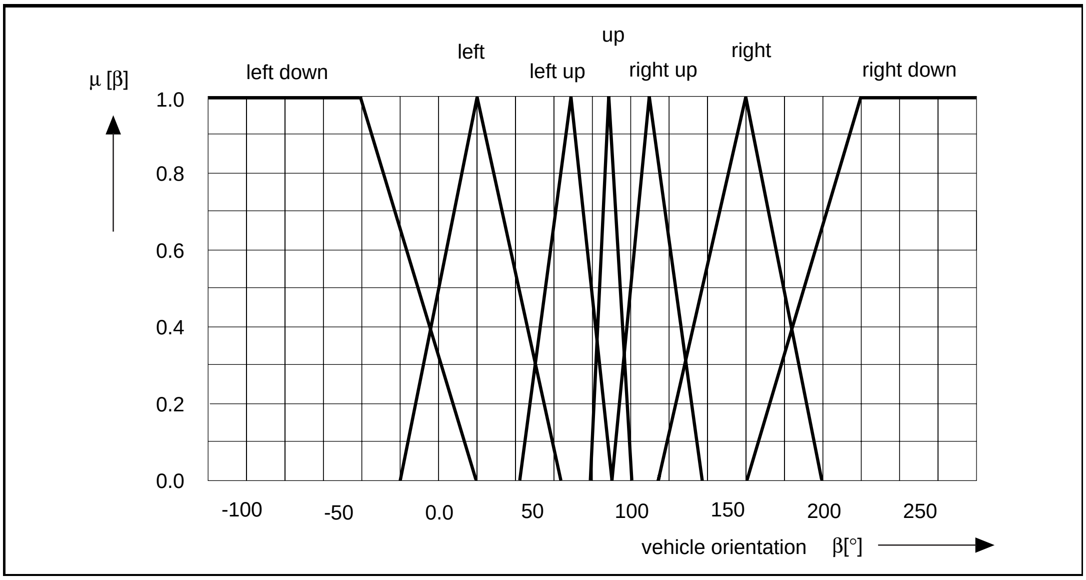


Figure 2-7. Linguistic Variable Vehicle Orientation β and Its Linguistic Terms

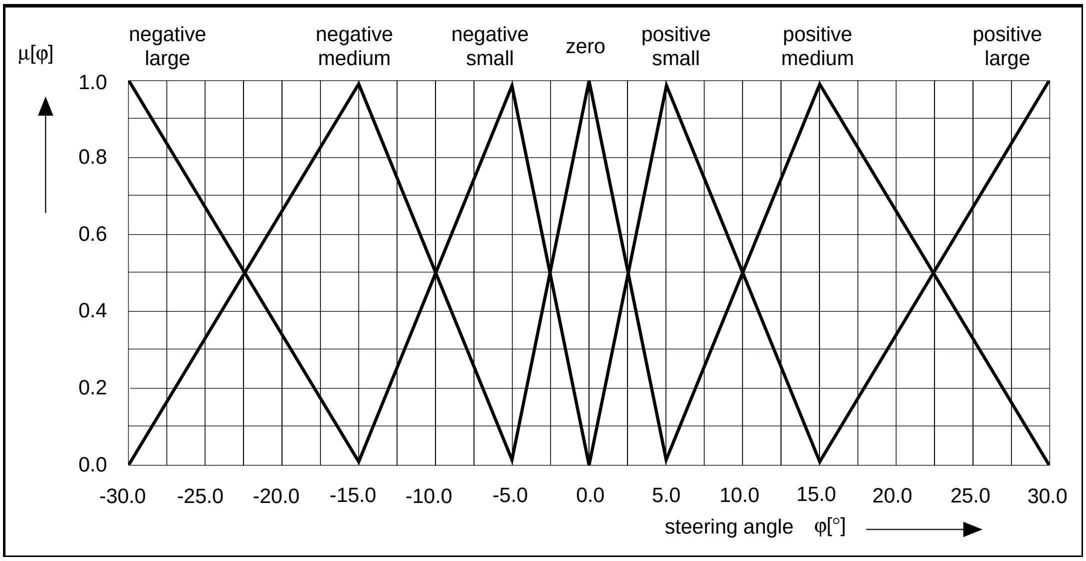


Figure 2-8. Linguistic Variable Steering Angle φ and Its Linguistic Terms

Looking at the following rule of the linguistic control strategy

> **IF** *vehicle position x* is *center* **AND** *vehicle orientation* β is *up*
> **THEN** adjust *steering angle* φ to *zero*,

the condition is composed of the linguistic term *center* from the linguistic variable, vehicle position *x*, and the linguistic term, *up*, from the linguistic variable, vehicle orientation β, are combined with the AND operator.

Because there are five terms for the linguistic variable, vehicle position *x*, and seven terms for the linguistic variable, vehicle orientation β, there are at most N = 35 different rules available to form a consistent rule base. Because there are only two input variables in this case, the complete rule base can be documented in matrix form, as shown in Figure 2-9.

AND		**vehicle position *x* [m]**				
		left	left center	center	right center	right
vehicle orientation β[°]	left down	negative small	negative medium	negative medium	negative large	negative large
	left	positive small	negative small	negative medium	negative large	negative large
	left up	positive medium	positive small	negative small	negative medium	negative large
	up	positive medium	positive medium	zero	negative medium	negative medium
	right up	positive large	positive medium	positive small	negative small	negative medium
	right	positive large	positive large	positive medium	positive small	negative small
	right down	positive large	positive large	positive medium	positive medium	negative small

Figure 2-9. Complete Linguistic Rule Base

Each combination of a column and a row describes a certain maneuvering situation—the condition of a certain rule. The conclusion is given by the term at the intersection of the column and row.

As an example, the following rule is highlighted in Figure 2-9:

IF *vehicle position x* is *left center* **AND** *vehicle orientation* β is *left* **THEN** adjust *steering angle* φ to *negative small.*

Structure of the Fuzzy Logic Vehicle Controller

The complete structure of a fuzzy logic controller is shown in Figure 2-10.

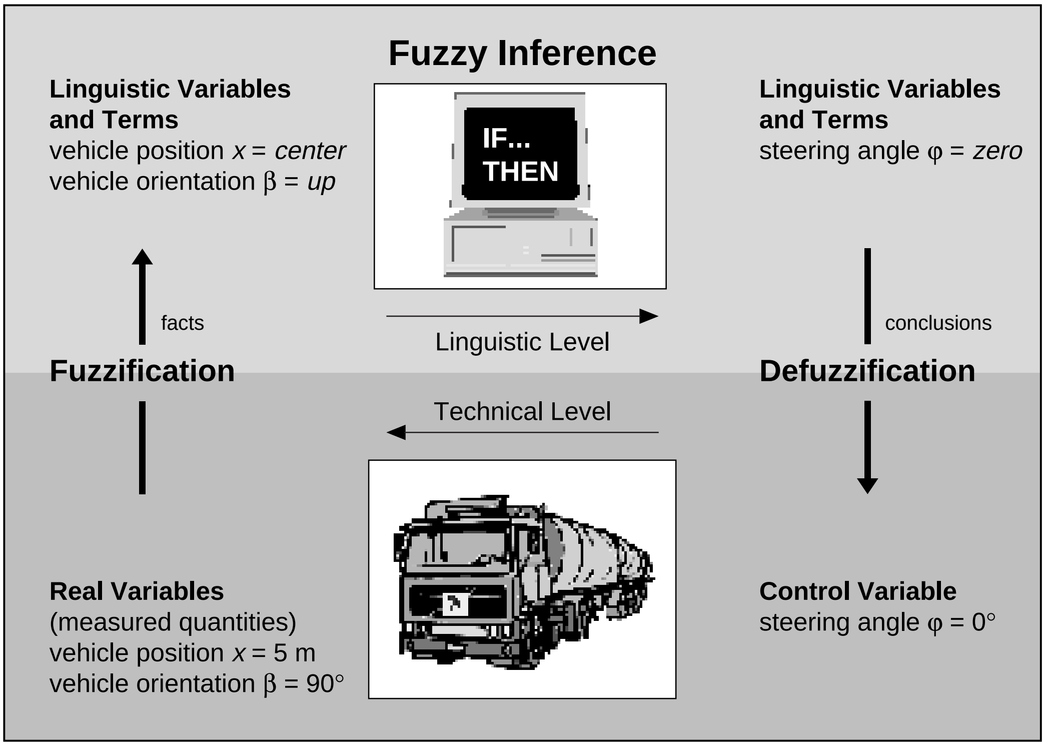


Figure 2-10. Complete Structure of a Fuzzy Controller

In the first step all sensor signals must be translated into linguistic variables. For example, a measured vehicle position x of 4.8 m must be translated to the linguistic value *almost center, just slightly left center*. This step is called *Fuzzification* because it uses fuzzy sets for translating real variables into linguistic variables.

Once all input variable values are translated into corresponding linguistic variable values, the *Fuzzy Inference* step is executed to derive a conclusion from the rule base that represents the control strategy. The result of this step is a linguistic value for the output variable. For example, the linguistic result for steering angle adjustment might be *steering angle φ a little less than zero*.

The *Defuzzification* step translates the linguistic result back into a real value representing the current value of the control variable.

Fuzzification Using Linguistic Variables

For a more detailed look at the fuzzification process, consider a maneuvering situation in which the vehicle position *x* is 5.1 m and the vehicle orientation β is 70°.

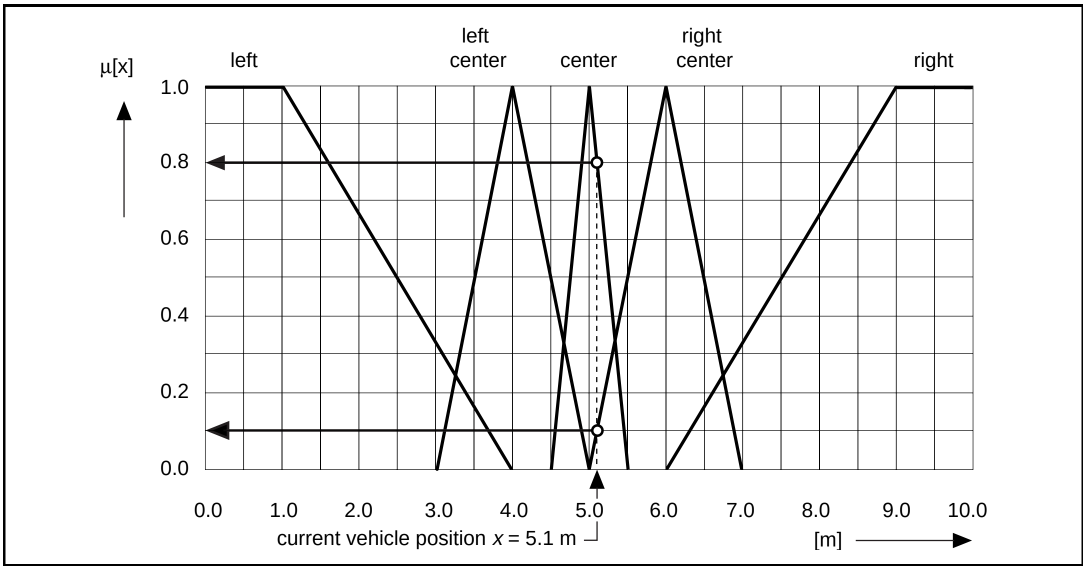


Figure 2-11. Fuzzification of the Vehicle Position $x = 5.1$ m

The current vehicle position $x = 5.1$ m belongs to the following linguistic terms (fuzzy sets):

left	with a degree of	0.0
left center	with a degree of	0.0
center	**with a degree of**	**0.8**
right center	**with a degree of**	**0.1**
right	with a degree of	0.0

The current vehicle position of 5.1 m is translated into the linguistic value {0.0, 0.0, **0.8**, **0.1**, 0.0}, which you can interpret as *still center, just slightly right center*.

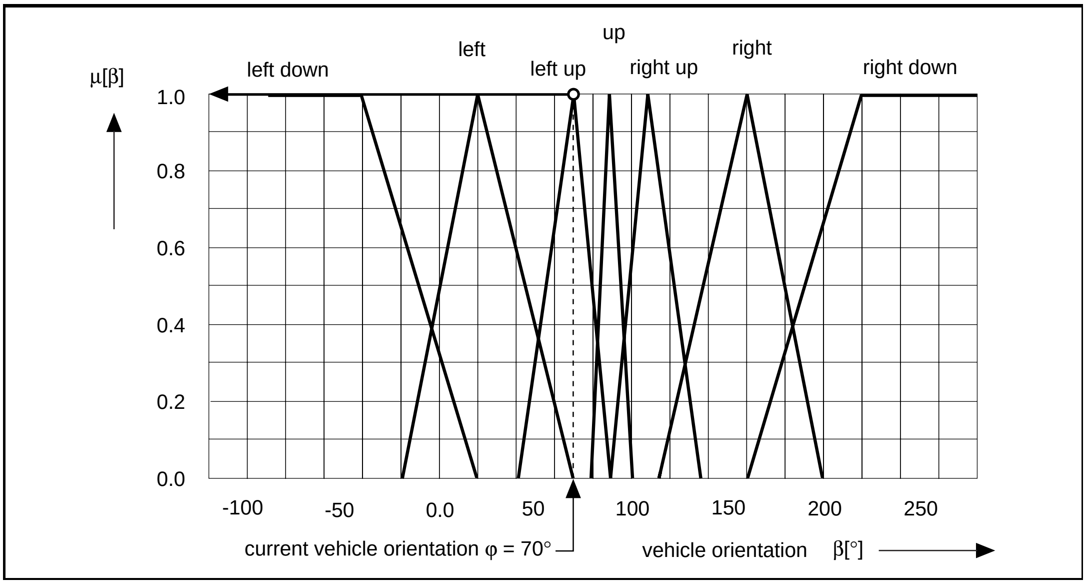


Figure 2-12. Fuzzification of the Vehicle Orientation φ = 70°

The current vehicle orientation φ = 70° belongs to the following linguistic terms (fuzzy sets):

left down	with a degree of	0.0
left	with a degree of	0.0
left up	**with a degree of**	**1.0**
up	with a degree of	0.0
right up	with a degree of	0.0
right	with a degree of	0.0
right down	with a degree of	0.0

The current vehicle orientation of 70° is translated into the linguistic value {0.0, 0.0, **1.0**, 0.0, 0.0, 0.0, 0.0}, which can be interpreted as *left up*.

How to define linguistic terms and membership functions is described in Chapter 4, *Design Methodology*.

Fuzzy Inference Using IF-THEN Rules

After all physical input values have been converted into linguistic values, identify all rules from the rule base that apply to the current maneuvering situation. These rules are identified in order to calculate the values of the linguistic output variable. The fuzzy inference step consists of two components:

- Aggregation—Evaluation of the IF part (condition) of each rule
- Composition—Evaluation of the THEN part (conclusion) of each rule

In the example, notice the IF part of each rule logically combines two linguistic terms from different linguistic variables with the word AND. Because our linguistic terms represent conditions that are partially true, the Boolean AND from conventional dual logic is not suited to model the word AND. So, you must define new operators that represent logical connectivities such as AND, OR, and NOT.

The three operators used in the majority of fuzzy logic applications are defined as listed in Figure 2-13.

AND: $\mu A \bullet B = \mathbf{min}(\mu A, \mu B)$

OR: $\mu A + B = \mathbf{max}(\mu A, \mu B)$

NOT: $\mu \neg A = 1 - \mu A$

Figure 2-13. Default Set of Fuzzy Logic Operators

Notice that these definitions agree with the logical operators used in Boolean logic. A truth table yields equivalent results using conventional operators.

The minimum operator represents the word AND. It is applied in the **aggregation** step to calculate a degree of truth for the IF condition of each rule in the rule base indicating how adequately each rule describes the current situation.

In the example situation, only the following two rules are valid descriptions of the current situation. These rules usually are called the *active rules*. All the other rules are called *inactive*.

(1) **IF** vehicle position x is ***center*** **AND** vehicle orientation β is ***left up***
(degree of truth = 0.8) minimum (degree of truth = 1.0) = **0.8**

THEN adjust steering angle φ to ***negative small***

(2) **IF** vehicle position x is ***right center*** **AND** vehicle orientation β is ***left up***
(degree of truth = 0.1) minimum (degree of truth = 1.0) = **0.1**

THEN adjust steering angle φ to ***negative small***

Each rule defines an action (conclusion) to be taken in the THEN condition. The degree to which the action is valid is given by the adequateness of the rule to the current situation. This adequateness is calculated by the aggregation step as the degree of truth of the IF condition.

In this case, the rule indicated by (1) results in the action “adjust steering angle φ to *negative small*” with a degree of 0.8. The rule indicated by (2) results in the action “adjust steering angle φ to *negative medium*” with a degree of 0.1.

The resulting conclusion (or action) must be composed of the differently weighted THEN conclusions of the active rules. This is done within the composition step.

The rules of this rule base are defined alternatively, i.e. they are logically linked by the word OR. Because the resulting conclusions of the rules are partially true, you cannot use the OR operator from conventional dual logic to calculate the resulting conclusion. In fuzzy logic, you must use the maximum operator (see Figure 2-13) instead.

For example, assume that two rules assert different degrees of truth for the linguistic term *positive medium*. One rule asserts *positive medium* with degree of truth 0.2, while another asserts *positive medium* with degree of truth 0.7. Because the two rules are related by the OR operator, the output of the fuzzy inference for the linguistic term is the maximum value of 0.7. Because the truck example has only one rule asserting a nonzero degree of truth for both *negative medium* and *negative small*, those values become the maximum values used.

The final result of the fuzzy inference for the linguistic variable steering angle φ is shown below:

negative large	to a degree of	0.0
negative medium	**to a degree of**	**0.1**
negative small	**to a degree of**	**0.8**
zero	to a degree of	0.0
positive small	to a degree of	0.0
positive medium	to a degree of	0.0
positive large	to a degree of	0.0

This type of fuzzy inference is called *Max-Min inference*. Because of certain optimization procedures of fuzzy systems, sometimes it is necessary to associate individual weights to each rule.

Defuzzification Using Linguistic Variables

The fuzzy inference process results in a linguistic value for the output variable. In this case, you can interpret the linguistic value {0.0, 0.1, 0.8, 0.0, 0.0, 0.0,0.0} as *still negative small* or *just slightly negative medium*. To use this linguistic value to adjust the steering wheel, it must be translated into a real (physical) value. This step is called *defuzzification* (see Figure 2-10).

The relationship between the linguistic values and the corresponding real values always is given by the membership function definitions describing the terms of the linguistic output variable (see Figure 2-8). In the example, you obtained a fuzzy inference result that is both fuzzy and ambiguous because there are two different actions with nonzero truth degrees to be taken at the same time. You must combine two conflicting actions that are defined as fuzzy sets to form a *crisp* real value. A solution to this problem is to find the best compromise between the two different goals. This compromise represents the best final conclusion received from the fuzzy inference process.

One of the two most commonly used methods for calculating the best compromise is the *Center-of-Area* method (CoA), also called *Center-of-Gravity* (CoG).

Following this defuzzification method, all membership functions representing the conclusion terms are truncated at the degree of validity of the rule to which the conclusion term belongs. The areas under the resulting function of all truncated terms are superimposed. Find the geometric center of the resulting area to determine the crisp compromise value, as shown in Figure 2-14.

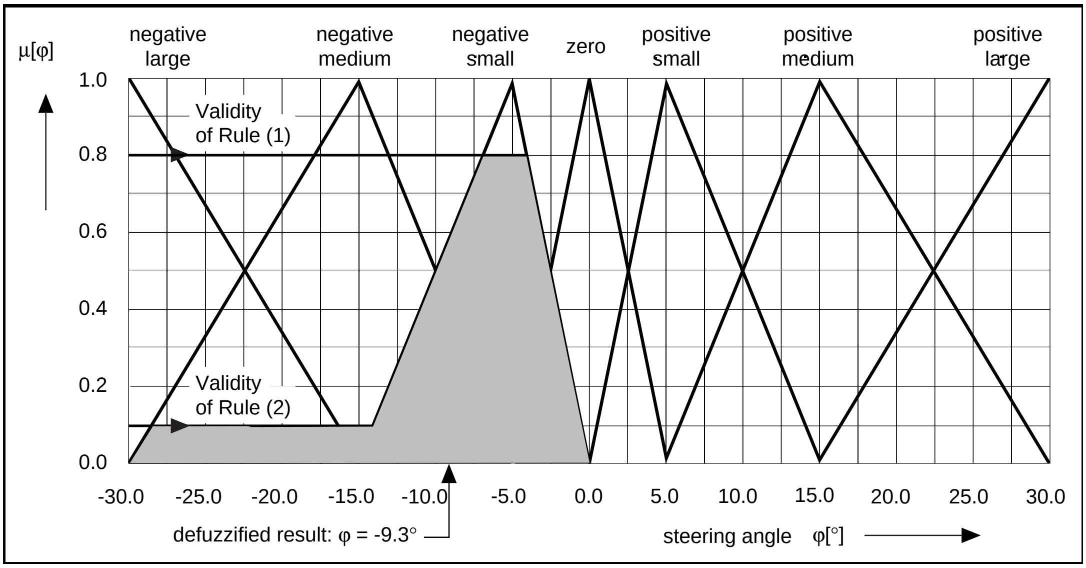


Figure 2-14. Defuzzification According to Center-of-Area (CoA)

This defuzzification method requires much computation because of the numerical integration necessary to calculate the center-of-area.

The second defuzzification method is called *Center-of-Maximum* (CoM). In the first step of this method, you determine the *typical* value of each term in the linguistic output variable. In the second step, you calculate the best compromise with a weighted average of typical values of the terms.

The most common approach to determine the typical value of each term is to find the maximum of the respective membership function. In the case of trapezoidal membership functions the median of the maximizing interval is chosen.

Each typical value is weighted by the degree to which the action term (conclusion) is true. Then, the crisp output value is calculated by a weighted average as shown in Figure 2-15.

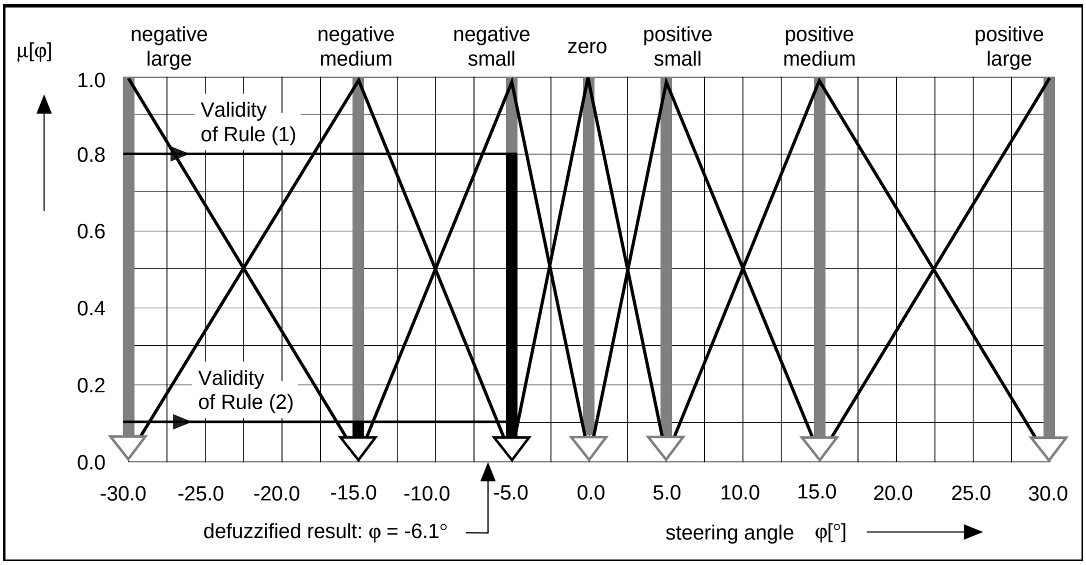


Figure 2-15. Defuzzification According to Center-of-Maximum (CoM)

With φ (*negative medium*) = –15° and φ (*negative small*) = –5° as typical values of the linguistic terms *negative medium* and *negative small*, and with the validity values V (rule 1) = 0.8 and V (rule 2) = 0.1 for the active rules, the possible defuzzification results are:

$$\varphi(\text{out}) = \frac{\varphi(\textit{negative medium}) \bullet V(\text{rule 2}) + \varphi(\textit{negative small}) \bullet V(\text{rule 1})}{V(\text{rule 2}) + V(\text{rule 1})}$$

$$\varphi(\text{out}) = -6.1°$$

The defuzzification method Center-of-Maximum is identical to the Center-of-Gravity method using singleton membership functions.

Figure 2-16 summarizes the fuzzy inference process for the maneuvering situation described above (assuming the CoA method of defuzzification).

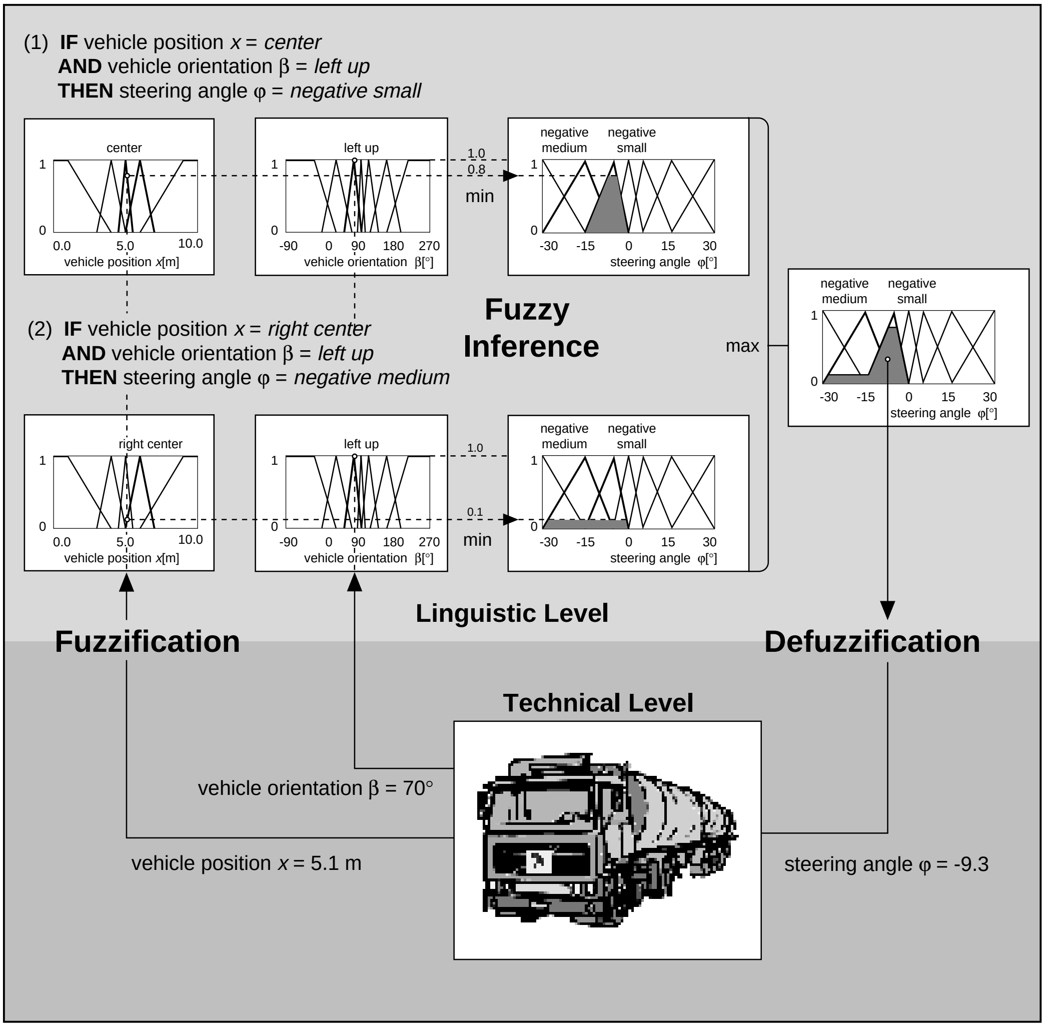

(1) IF vehicle position x = center
AND vehicle orientation β = left up
THEN steering angle φ = negative small
center
left up
1.0
0.8
min
negative medium
negative small
vehicle position x[m]
vehicle orientation β[°]
steering angle φ[°]
(2) IF vehicle position x = right center
AND vehicle orientation β = left up
THEN steering angle φ = negative medium
Fuzzy Inference
max
right center
0.1
Linguistic Level
Fuzzification
Defuzzification
Technical Level
vehicle orientation β = 70°
vehicle position x = 5.1 m
steering angle φ = -9.3

Figure 2-16. Fuzzification, Fuzzy Inference and Defuzzification for a Certain Maneuvering Situation

Without modification, the CoA defuzzification method limits the range of the output value compared to the possible range as shown in Figure 2-17. This problem can be solved easily by a fictitious extension of the left and right side border terms when computing the center-of-area. With this extension, the complete value range of the output variable can be realized (see Figure 2-17). In this case the defuzzification method is called *modified Center-of-Area*.

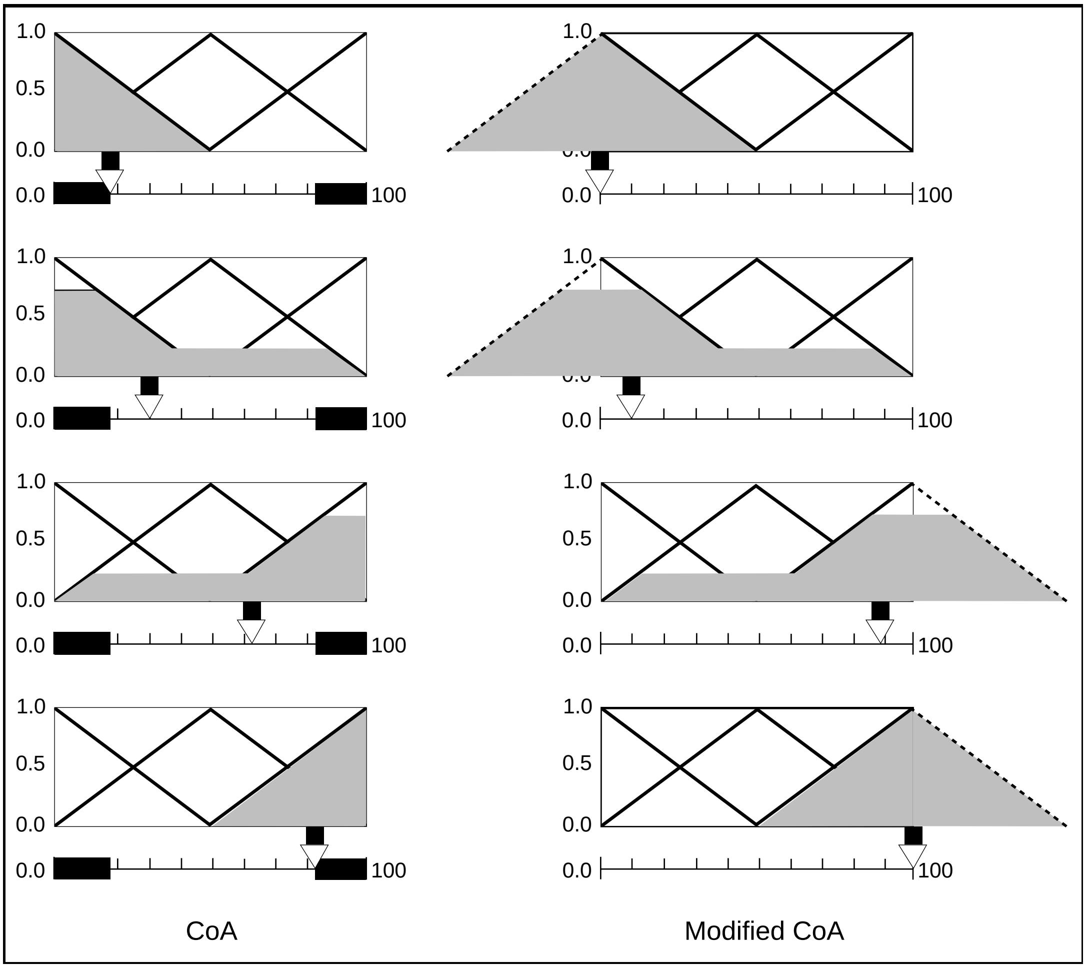


Figure 2-17. Modified CoA for Complete Output Value Range

The defuzzification methods CoM and CoA are commonly applied to closed-loop control applications of fuzzy logic. They usually lead to continuous output signals because the *best compromise* never can jump to a different value with a small change to the inputs.

For pattern recognition applications, the defuzzification method *Mean-of-Maximum* (MoM) must be applied. This defuzzification method calculates the *most plausible* result. Rather than averaging the different inference results, MoM selects the typical value of the output term that is most valid.

In the example situation, the output term *negative small* is the most valid term (see Figures 2-14 and 2-15). Its typical value is φ (*negative small*) = 5°, which would be the immediate defuzzification result. If you want to identify objects by classification of a sensor signal, for example, you are interested in the most plausible result.

In decision support systems, the choice of the defuzzification method depends on the context of the decision to be calculated by the fuzzy system. For quantitative decisions like project prioritization, CoM should be applied. For qualitative decisions, such as an evaluation of credit worthiness, MoM is the correct method.

Fuzzy Controllers

Chapter 3

This chapter describes various implementations and Input/Output (I/O) characteristics of fuzzy controllers.

Structure of a Fuzzy Controller

A fuzzy controller is composed of the three calculation steps Fuzzification, Fuzzy Inference and Defuzzification. The control strategy based on engineering experience with respect to a closed-loop control application is implemented by linguistic rules integrated in the rule base of the controller.

A fuzzy controller has a static and deterministic structure, as shown in Figure 3-1, which can be described with an I/O characteristic curve.

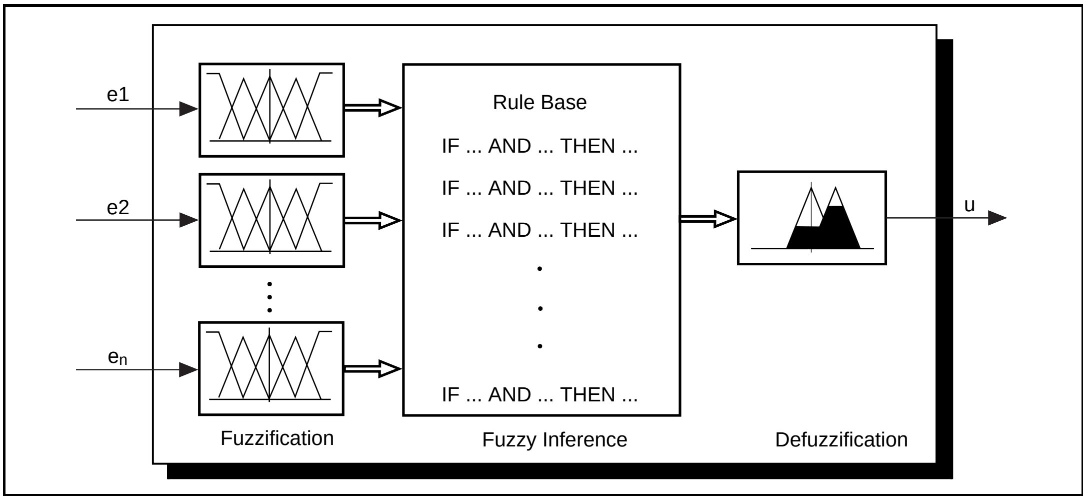


Figure 3-1. Internal Structure of a Fuzzy Controller

In principle, there are two different implementation forms:

- Offline Fuzzy Controller—In this case, the three-step calculation scheme is transformed into a reference table from which the command values can be derived. You can calculate intermediate command values by interpolation.
- Online Fuzzy Controller—In this case, the three-step calculation scheme is evaluated online. This is the standard implementation form of the Fuzzy Logic Toolkit.

Closed-Loop Control Structures with Fuzzy Controllers

There are many different ways to use fuzzy controllers in closed-loop control applications. The most simple structure uses the sensor signals from the process as input signals for the fuzzy controller and its outputs as command values to drive the actuators of the process.
A corresponding control loop structure is shown in Figure 3-2.

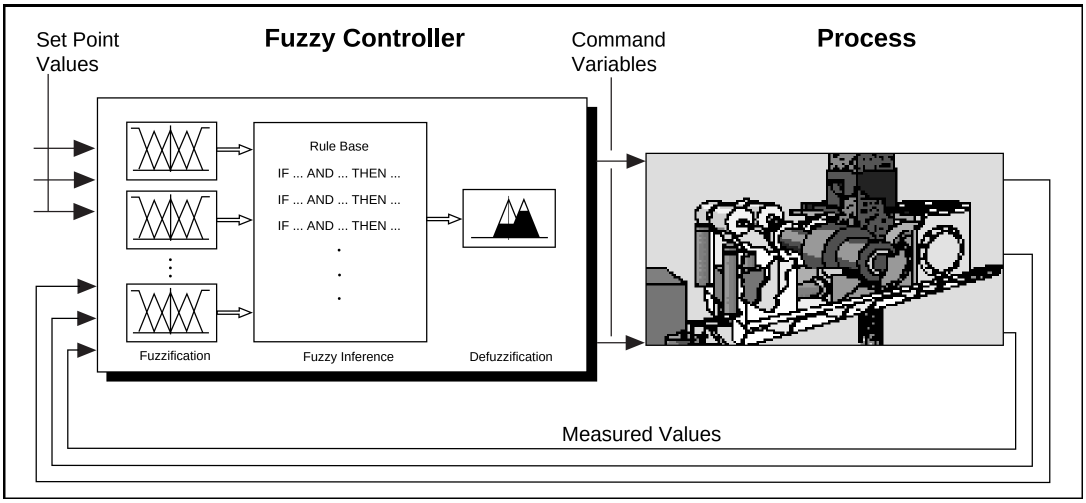


Figure 3-2. Simple Closed-Loop Control Structure with Fuzzy Controller

Pure fuzzy control applications are more the exception than the rule. In most cases the fuzzy controller output serves as reference parameters (such as gains) that are provided to a conventional controller instead of driving actuators in the process directly.

Because you can regard a fuzzy controller as a nonlinear characteristic field controller, it has no internal dynamic aspects. Thus, any dynamic property must be implemented by an appropriate preprocessing of the measured input data.

The Fuzzy-PI Controller, shown in Figure 3-3, uses the error signal e(t) and its derivative *d*e(t)/*d*t from the measured data preprocessing step as inputs. If the output signal describes the necessary difference toward the current output value, a subsequent integrator device is needed to build up the command variable value.

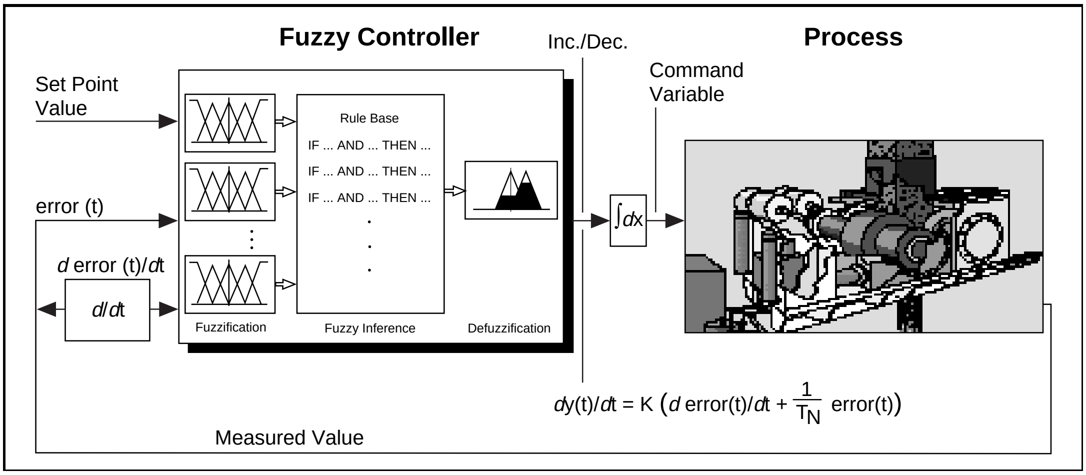


Figure 3-3. Closed-Loop Control Structure with Fuzzy-PI Controller

A fuzzy controller with two inputs and one output that increases because of increasing input values is called a Fuzzy-PI Controller. If an error signal and its derivative are used as input signals it can be regarded as a generalization of the conventional PI controller.

The benefit of the Fuzzy-PI Controller is that it does not have a special operating point. The rules evaluate the difference between the measured value and the set value, the error signal, and the tendency of the error signal to determine whether to increment or decrement the control variable. The absolute value of the command variable has no influence.

The advantage over a conventional PI controller is that a Fuzzy-PI Controller can implement nonlinear control strategies and that it uses linguistic rules. It is possible to take the error tendency into account only when the error becomes small.

Figure 3-4 shows a controller structure that often is used in the chemical industry and process technology. In this application, PID controllers are used to control single process parameters. The operating point of the entire process usually is supervised by human operators.

For automatic operation of such multivariable control problems, you must build a model-based controller. But for most applications, either the process is too complex to be modeled adequately, or the mathematical modeling task requires too much time.

The benefit of fuzzy controllers is that the experience and the knowledge of the operators in supervising the process often can be used to form a linguistic rule base with much less effort.

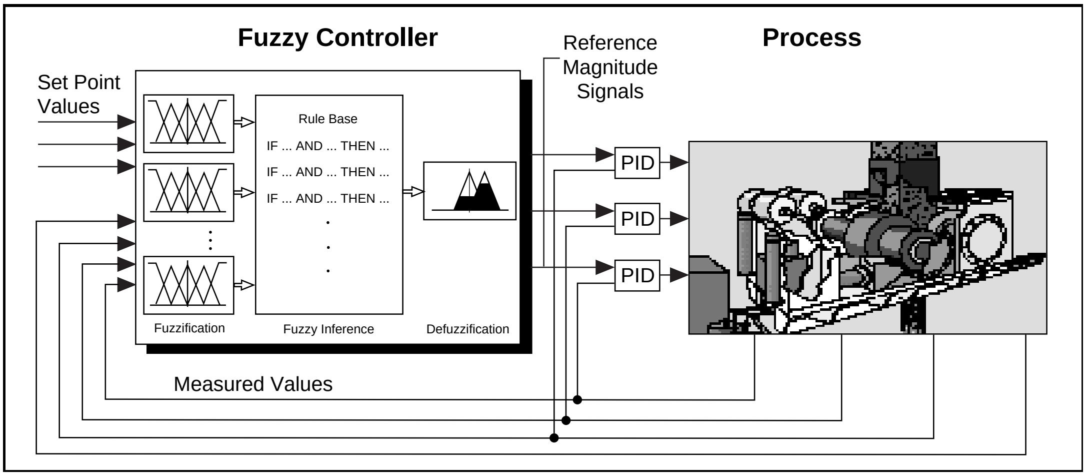


Figure 3-4. Fuzzy Controller with Underlying PID Control Loops

The next example structure shows how you cause a fuzzy controller to tune the parameters of a conventional PID controller automatically. For this, the fuzzy controller constantly interprets the process reaction and calculates the optimal P, I, and D gains. You can apply this control structure to processes that change their characteristics over time. This structure is shown in Figure 3-5.

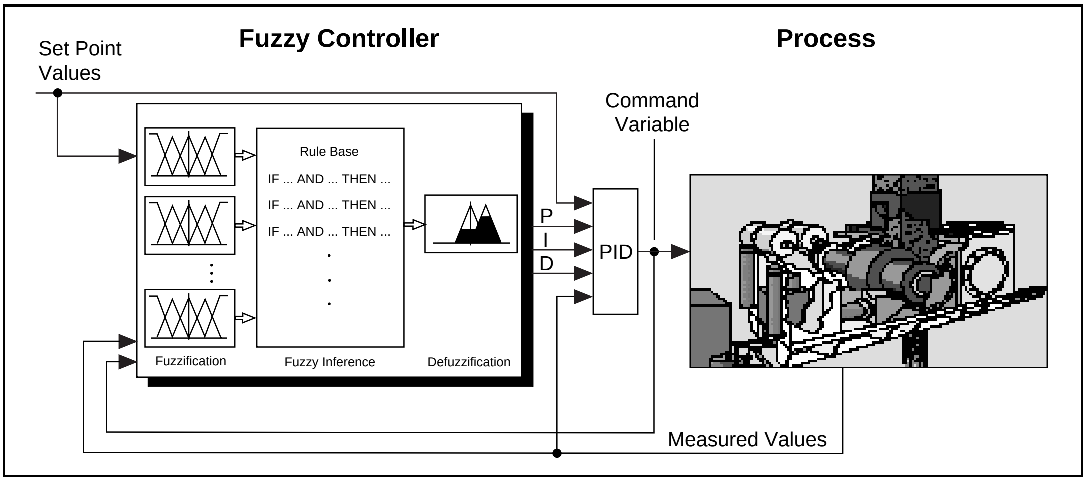


Figure 3-5. Fuzzy Controller for Parameter Adaptation of a PID Controller

Both the fuzzy controller and the PID controller work in parallel. The output signals from both controllers are added, but the output signal from the fuzzy controller is zero under normal operating conditions. The PID controller output leads the process. The fuzzy controller intervenes only when it detects abnormal operating conditions, such as strong disturbances.

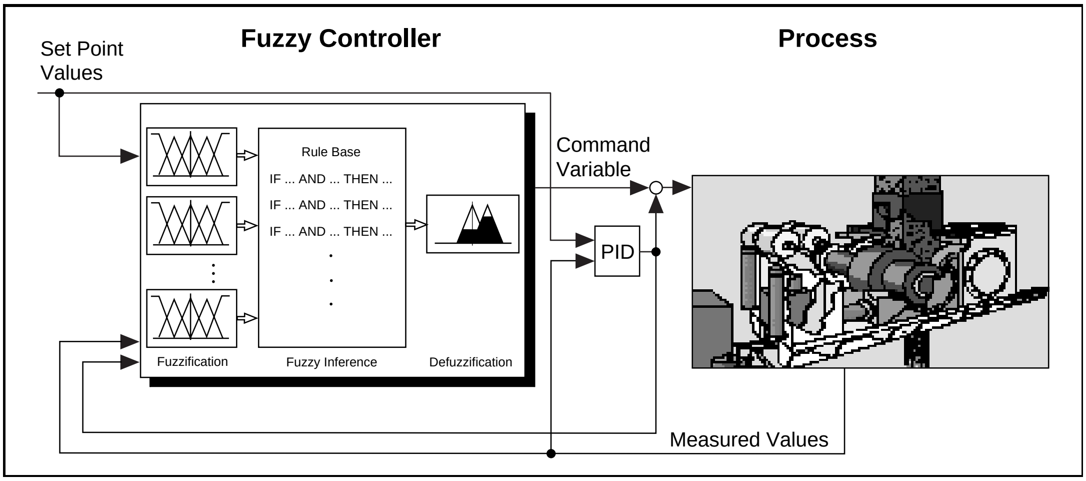


Figure 3-6. Fuzzy Controller for Correction of a PID Controller Output

I/O Characteristics of Fuzzy Controllers

You can consider a fuzzy controller to be a nonlinear characteristic field controller. Its behavior is determined by its rule base and the membership functions that model the terms of the linguistic input and output variables. Because it has no internal dynamic aspects, its transient response can be described entirely by its I/O characteristics.

To illustrate how the I/O characteristic of a fuzzy controller depends on design parameters such as rule base and membership function specification, you must first restrict yourself to a single-input fuzzy controller. Most of these ideas directly apply to fuzzy controllers with two or more inputs.

Figure 3-7 shows the I/O characteristic of a fuzzy controller with only three linguistic terms for the input variable *x* and the output variable *y*. The rule base consists of three rules, indicating that the output increases because of an increasing input value.

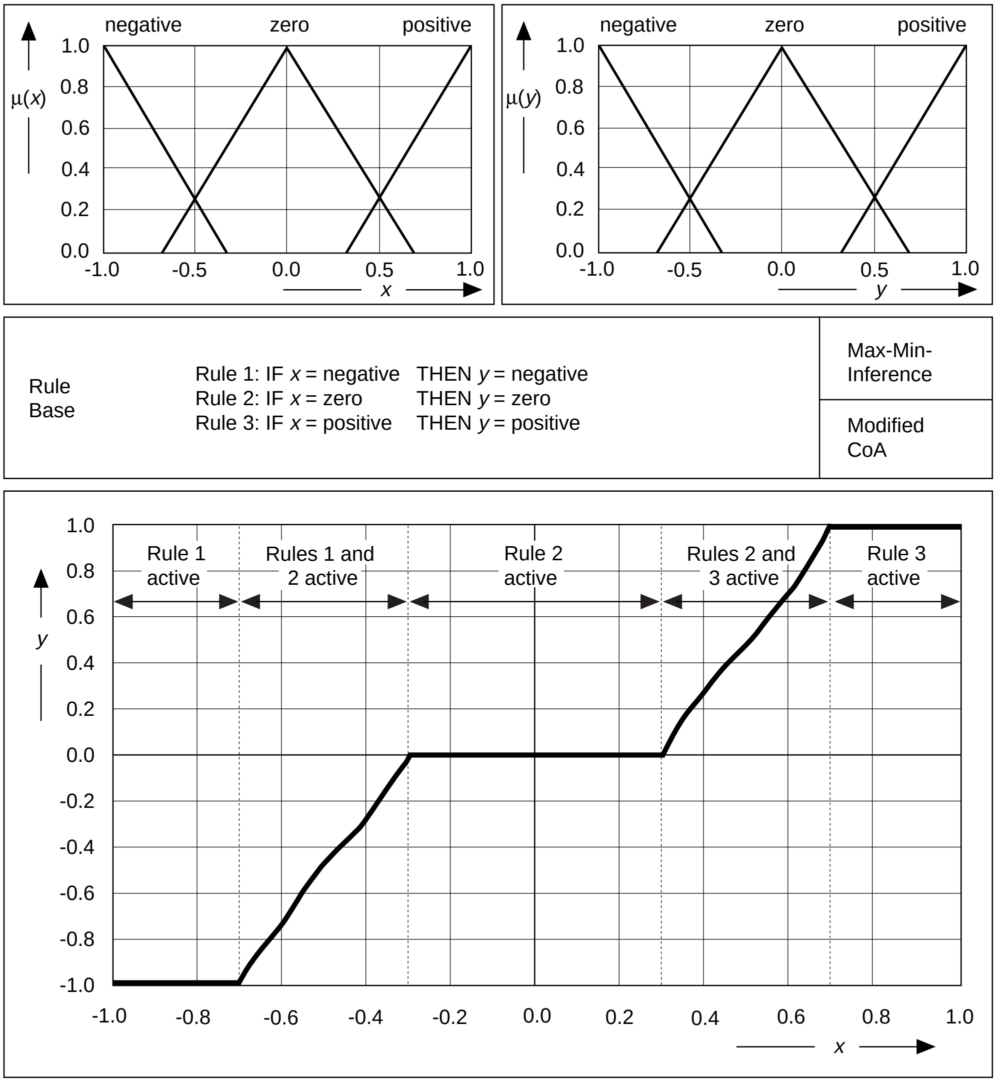


Figure 3-7. I/O Characteristic of a Fuzzy Controller (Partially Overlapping Input Terms)

The resulting controller characteristic shows nonlinear behavior. Because of the partially overlapping input terms (antecedence terms) you obtain different intervals within the controller characteristic. Outside of the overlapping regions, there is only one valid rule. This leads to a constant value for the output value determined by the output term (conclusion) of the output variable, which is independent of the degree of truth for that rule.

The overlapping sections of the antecedence terms lead to the rising intervals of the controller characteristic. Within these parts, two rules are active simultaneously. The output value is determined by the different conclusion terms weighted by the degree of truth of the different active rules. Notice that the rising edges of the controller characteristic are nonlinear because of the overlapping triangular conclusion terms.

Figure 3-8 shows the resulting controller characteristic for entirely overlapping antecedence terms. The conclusion term distribution and the rule base are left unchanged for this case.

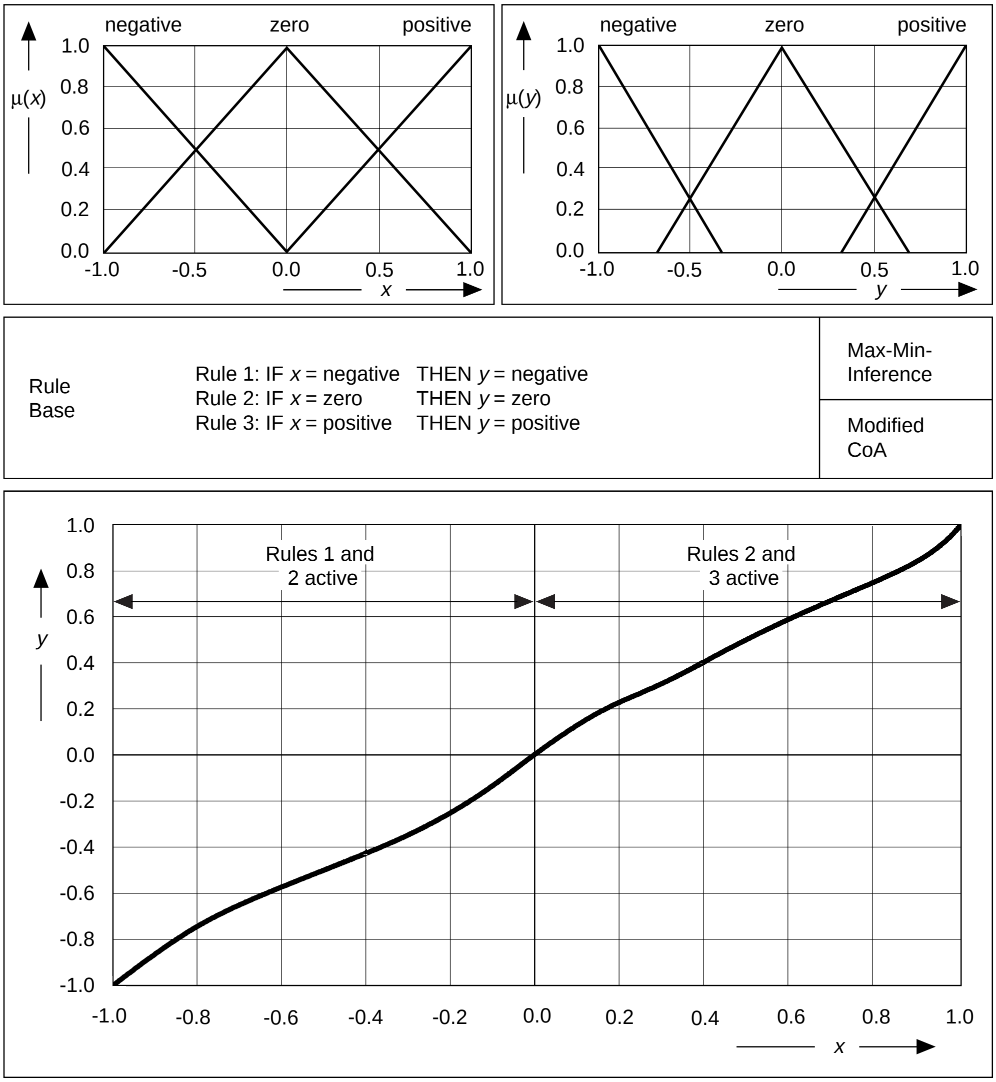


Figure 3-8. I/O Characteristic of a Fuzzy Controller (Entirely Overlapping Input Terms)

Because the antecedence terms completely overlap, there are always two active rules. The output value is determined again by the different conclusion terms weighted by the degree of truth for the different active rules leading to the nonlinear pass of the controller characteristic.

Figure 3-9 shows the resulting controller characteristic for nonoverlapping antecedence terms describing the input variable.

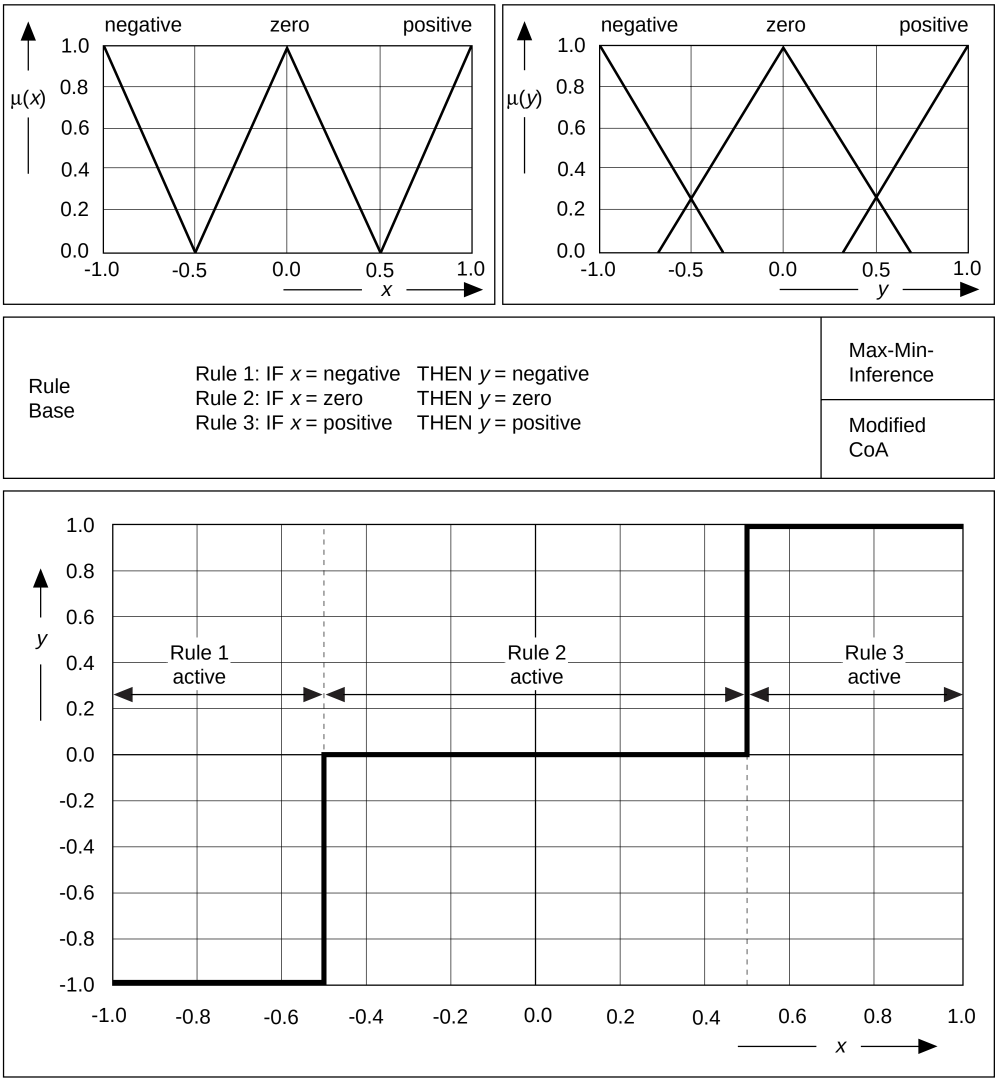
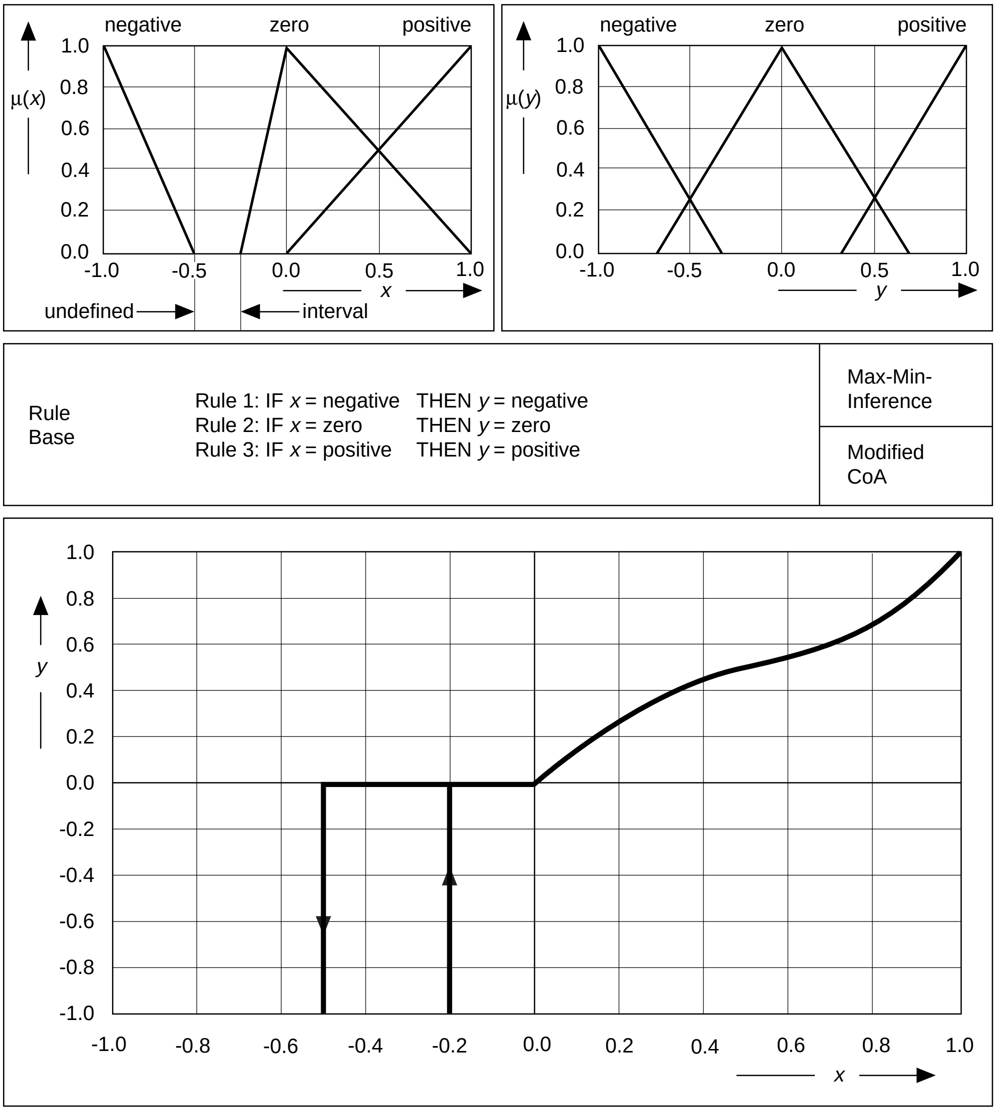


Figure 3-9. I/O Characteristic of a Fuzzy Controller (Nonoverlapping Input Terms)

In this case, only one rule is active for each input situation leading to the stepped controller characteristic shown in Figure 3-9.

If there are undefined intervals within input and output terms, or the rule base is incomplete, you must specify what the fuzzy controller must do. If there is no rule available for a certain situation, the output value is undefined. One way to avoid this problem is to leave the current output value unchanged until the controller encounters a situation that is covered by the rules. The resulting effect on the controller characteristic is shown in Figure 3-10.



Rule Base

Rule 1: IF x = negative THEN y = negative
Rule 2: IF x = zero THEN y = zero
Rule 3: IF x = positive THEN y = positive

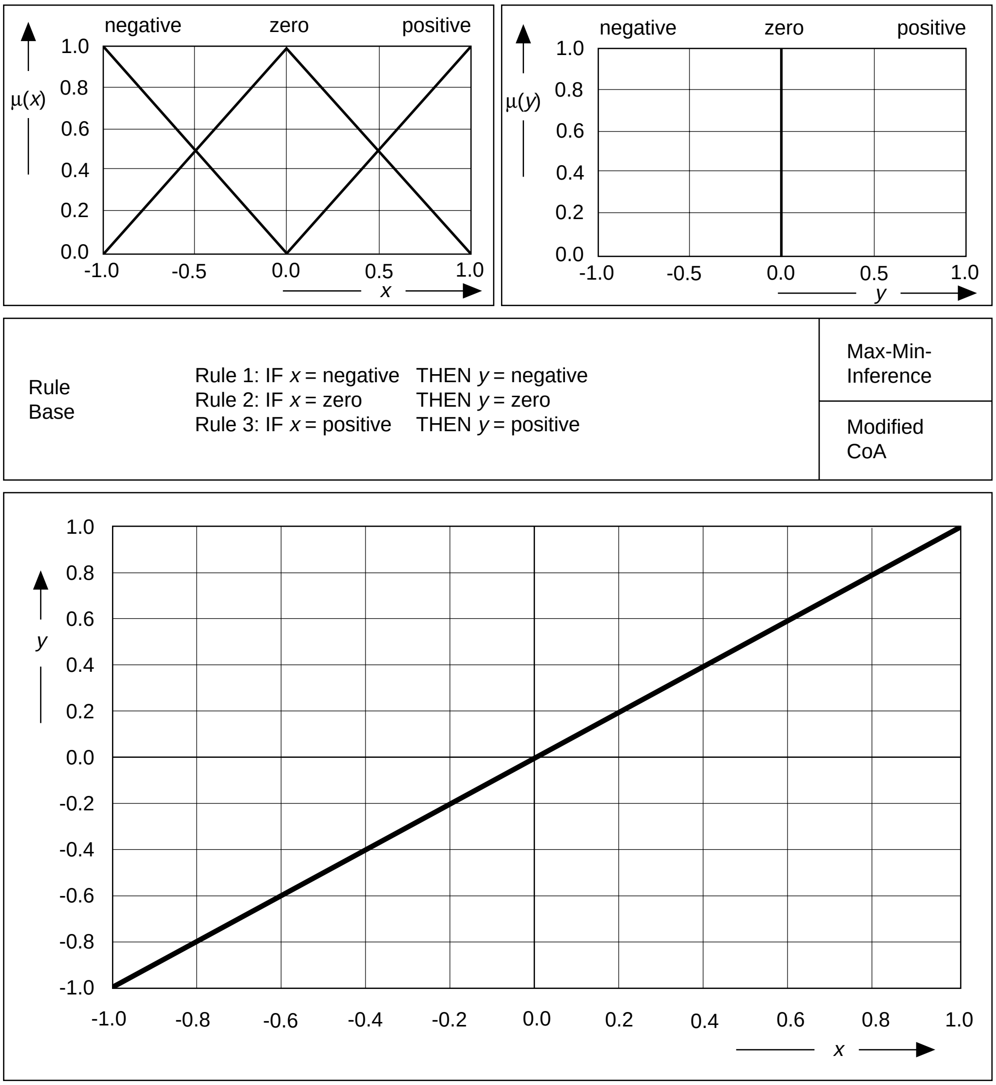
Figure 3-10. I/O Characteristic of a Fuzzy Controller (Undefined Input Term Interval)

If an old output value is used as a default value, undefined intervals or incomplete rule bases lead to hysteresis effects on the controller characteristic

An exact linear controller characteristic can easily be obtained for a single-input controller by using nonoverlapping, rectangular-shaped conclusion terms. In this case both area and momentum vary linearly with the degree of truth, and there is no distortion caused by overlapping regions of the output terms.

The simplest way to obtain a linear controller characteristic is to use *singletons* as conclusion terms with entirely overlapping input terms (see Figure 3-11). Singletons are normalized rectangular membership functions with an infinite small width.

Using singleton membership functions for the conclusion terms makes the CoG defuzzification method identical to the CoM method.



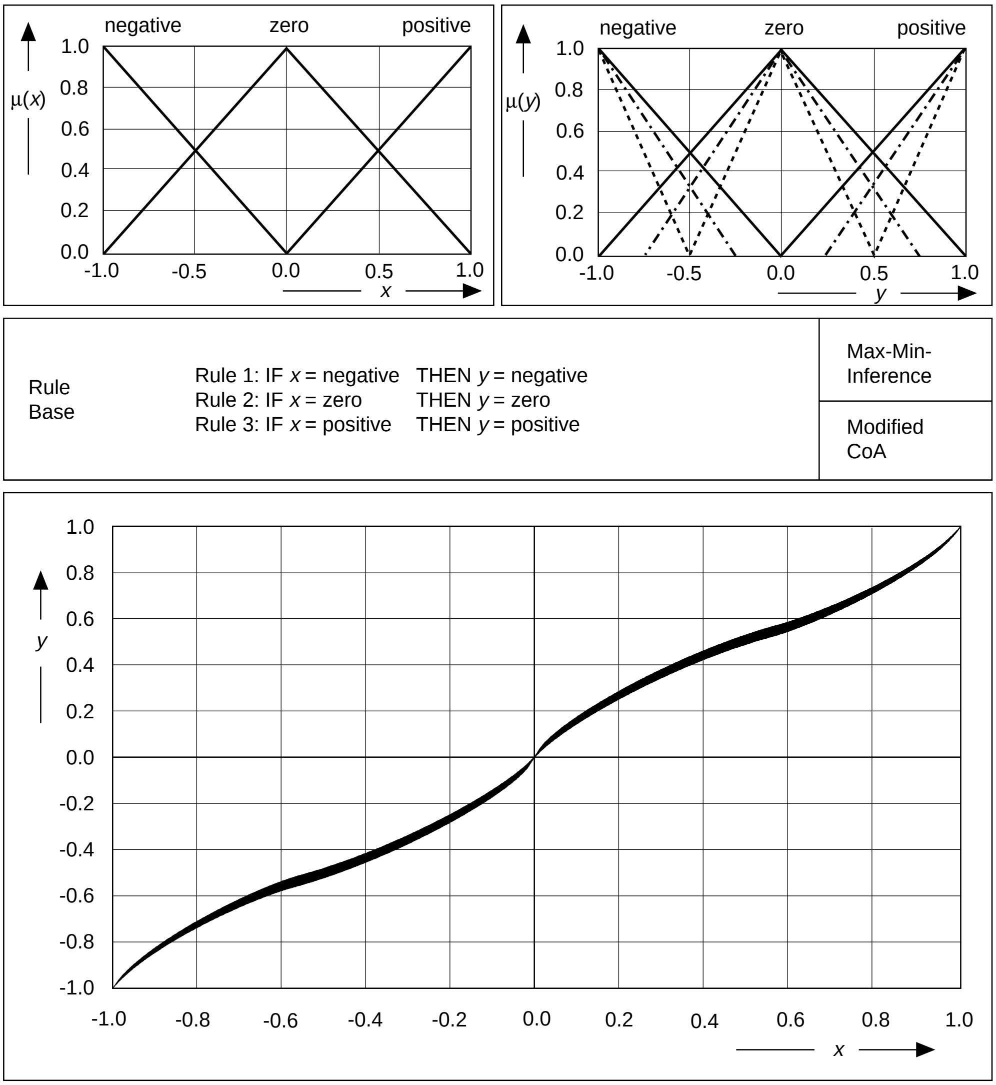
Rule Base

Rule 1: IF x = negative THEN y = negative
Rule 2: IF x = zero THEN y = zero
Rule 3: IF x = positive THEN y = positive

Figure 3-11. I/O Characteristic of a Fuzzy Controller (Singletons as Output Terms, Entirely Overlapping Input Terms)

Varying the overlapping degree of the membership functions for the conclusion terms by leaving the input terms entirely overlapped does not change the controller characteristic very much, especially if all the conclusion terms are equal in width as shown in Figure 3-12. Then, only the typical values of the conclusion terms are important.

Therefore, in most closed-loop control applications the output terms can sufficiently be modeled by singleton membership functions rather than triangular or other membership function types.



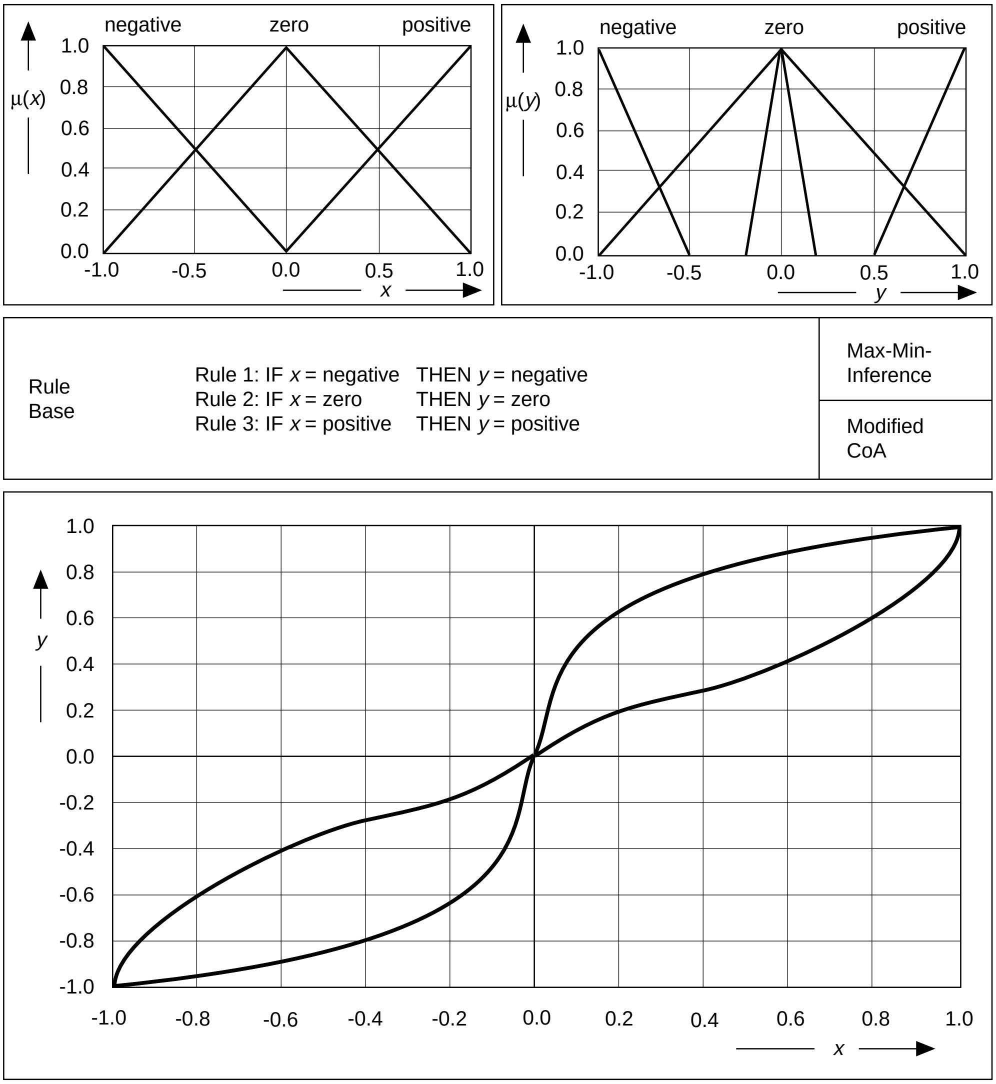
Rule Base

Rule 1: IF x = negative THEN y = negative
Rule 2: IF x = zero THEN y = zero
Rule 3: IF x = positive THEN y = positive

Figure 3-12. I/O Characteristics of a Fuzzy Controller (Different Overlapping Degrees of Membership Functions for the Output Terms)

Figure 3-12 shows that the overlapping degree of the membership functions for the conclusion terms has no significant influence on the controller characteristic, if all the conclusion terms are equal in width.

Instead, using output terms that are modeled by membership functions with equally distributed *typical* values but different scopes of influence, significantly influences the controller characteristic. The different terms have different areas and thus different weights with respect to the defuzzification process. A wide output term has more influence on the inference result than a small neighboring output term. This effect is demonstrated in Figure 3-13.



Rule Base

Rule 1: IF *x* = negative THEN *y* = negative
Rule 2: IF *x* = zero THEN *y* = zero
Rule 3: IF *x* = positive THEN *y* = positive

Figure 3-13. I/O Characteristics of a Fuzzy Controller (Wide and Small Membership Functions for the Output Terms)

Using CoA or CoM as the defuzzification method results in continuous courses of the controller characteristic, especially within those intervals of the input values in which two or more rules are active simultaneously. This is because of the averaging character of the defuzzification methods described in Chapter 2, *Overview of Fuzzy Logic*.

Using the MoM defuzzification method, the most plausible result is calculated. In other words, the typical value of the conclusion term of the most valid rule is taken as a crisp output value. This results in stepped output characteristics as shown in Figure 3-14.

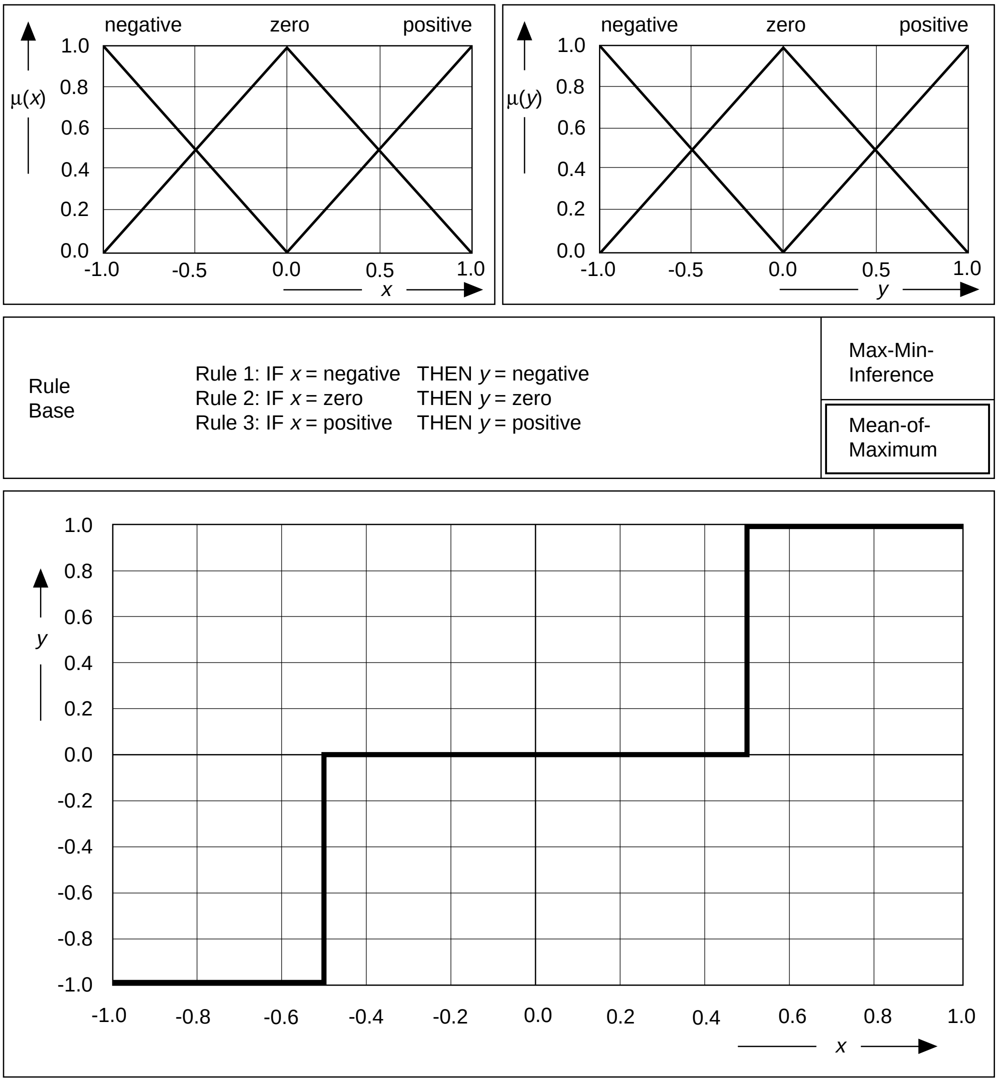
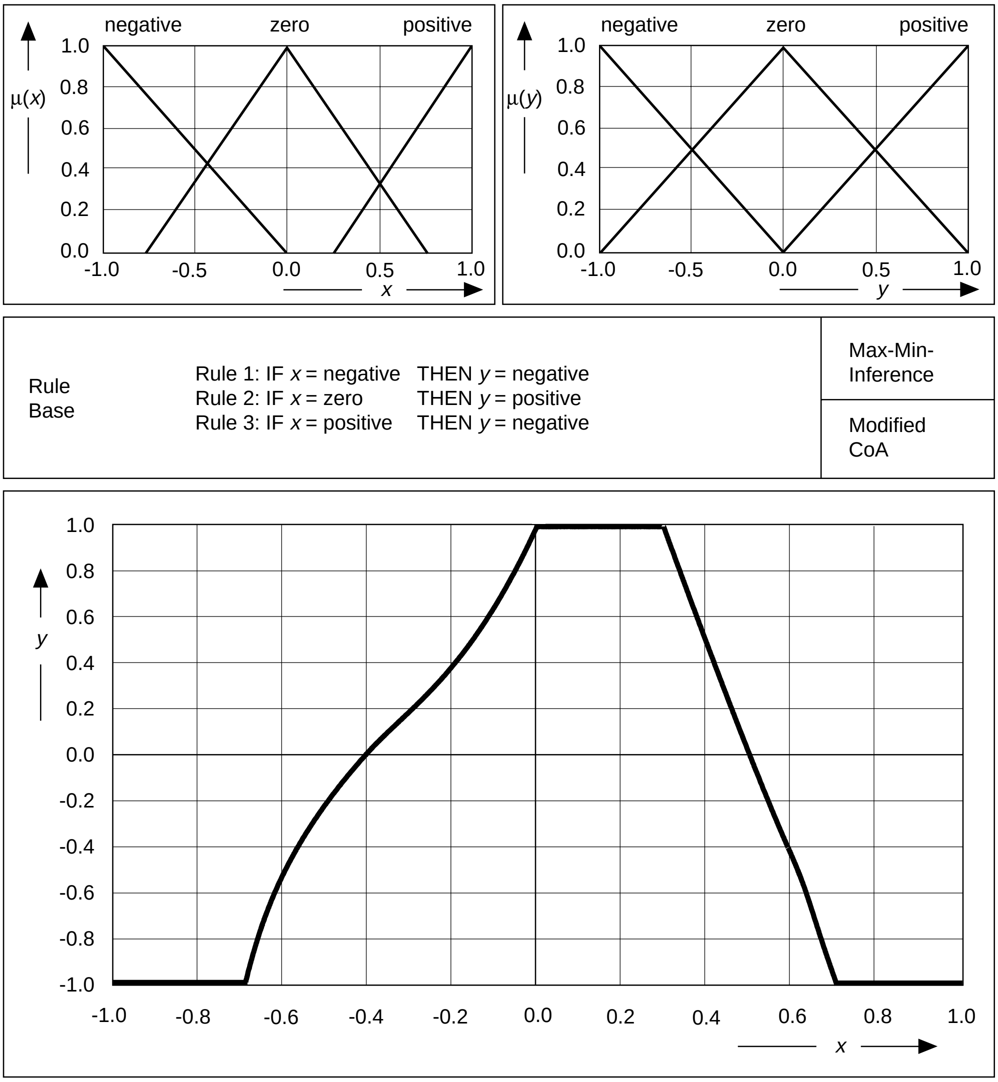


Figure 3-14. I/O Characteristic of a Fuzzy Controller with Mean-of-Maximum (Entirely Overlapping Membership Functions for Input and Output Terms)

The most important influence on the controller characteristic is applied by the rule base itself. The rule base determines the principal functionality of the controller.

Figure 3-15 illustrates how the controller characteristic changes if the rule base of the previous example is changed to the following:

Rule 1:	**IF**	*x* = negative	**THEN**	*y* = negative
Rule 2:	**IF**	*x* = zero	**THEN**	*y* = positive
Rule 3:	**IF**	x = positive	**THEN**	*y* = negative



Rule Base

Rule 1: IF x = negative THEN y = negative
Rule 2: IF x = zero THEN y = positive
Rule 3: IF x = positive THEN y = negative

Max-Min-Inference

Modified CoA

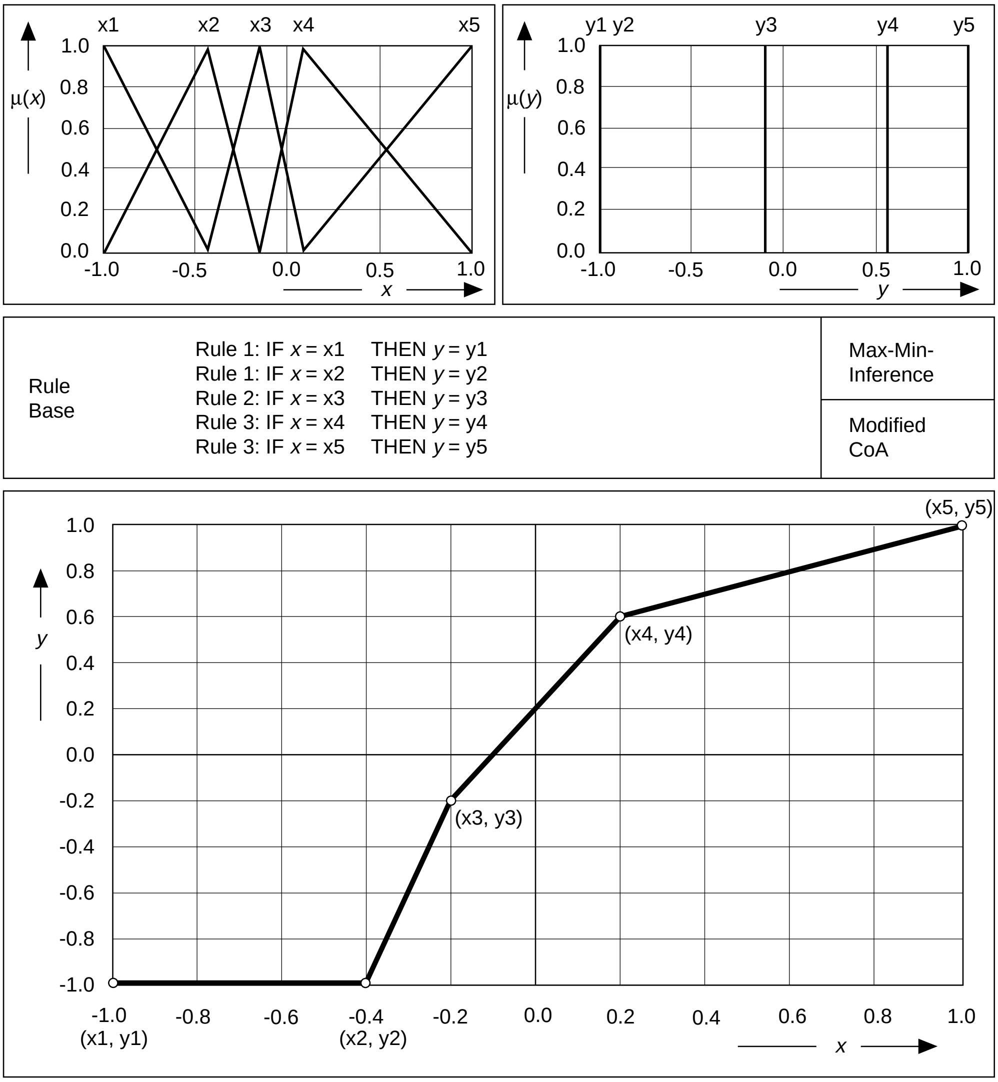
Figure 3-15. I/O Characteristic of a Fuzzy Controller with a Changed Rule Base

The examples show that you can use a fuzzy controller to perform arbitrary I/O operations. The number of linguistic input and output terms depends on the desired characteristic type and the precision to which the given I/O characteristic is approximated.

Consider, for example, the stepped linear characteristic curve shown in Figure 3-16. There are four linear sections that can be described by the five circled base points (x_i, y_i).

To reproduce the given characteristic by a single input fuzzy controller, use five linguistic terms each for the input and output quantities, naming them x_1, x_2, …, x_5 and y_1, y_2, …, y_5, respectively. To obtain the stepped linear sections between the base points, exactly two active rules always must be available. This can be implemented by overlapping triangular membership functions for the input variable entirely, each with a typical value that corresponds to a certain base point component, x_i.

To obtain characteristic sections that are exactly linear, the output variable must be modeled by singleton membership functions, each with a typical value that corresponds to a certain base point component, y_i. The rule base is then a kind of linguistic enumeration of the five base points.



Rule Base

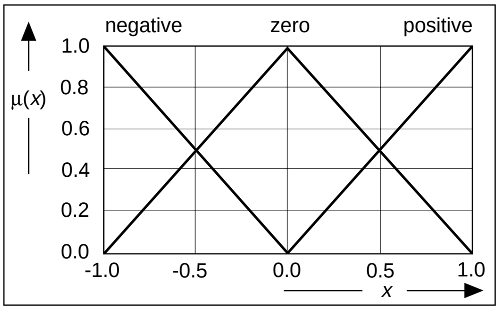
Rule 1: IF *x* = x1 THEN *y* = y1
Rule 1: IF *x* = x2 THEN *y* = y2
Rule 2: IF *x* = x3 THEN *y* = y3
Rule 3: IF *x* = x4 THEN *y* = y4
Rule 3: IF *x* = x5 THEN *y* = y5

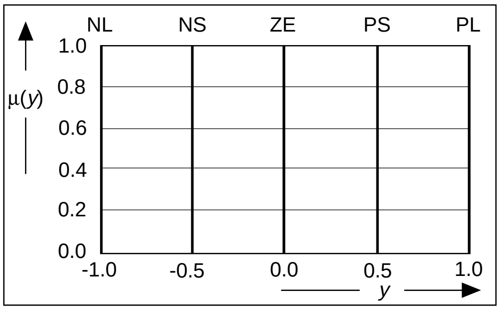
Max-Min-Inference

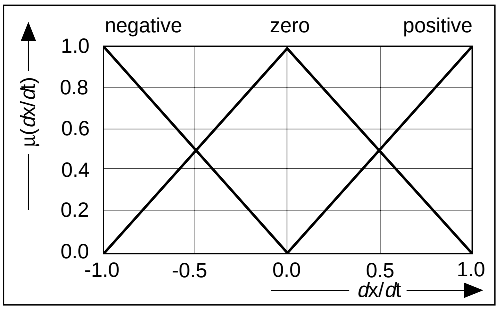
Modified CoA

Figure 3-16. Fuzzy Controller for a Given I/O Characteristic

In principle, these conclusions about I/O characteristics are valid for fuzzy controllers with two or more inputs as well. However, an additional nonlinear effect is raised by the AND-operation combining the different input conditions (also called antecedence terms). Usually the AND-operation is modeled by the minimum operator (see Figure 3-16) that always prefers as a result the antecedence term of the rule with the lowest degree of truth. Figure 3-17 shows the I/O characteristic field for a dual input fuzzy controller.







Rule Base		input x		
		negative	zero	positive
input dx/dt	negative	NL	NS	ZE
	zero	NS	ZE	PS
	positive	ZE	PS	PL

Max-Min- Inference	Modified CoA

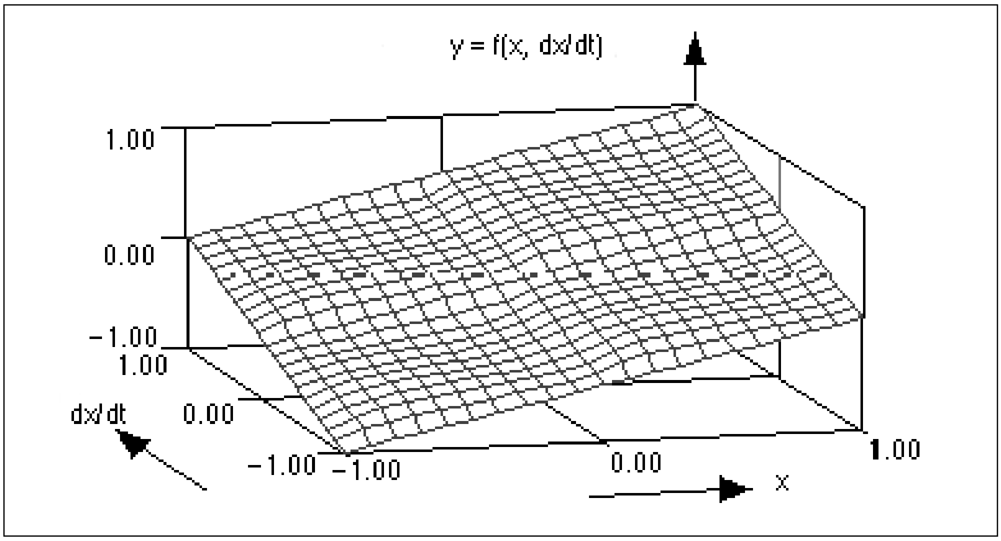
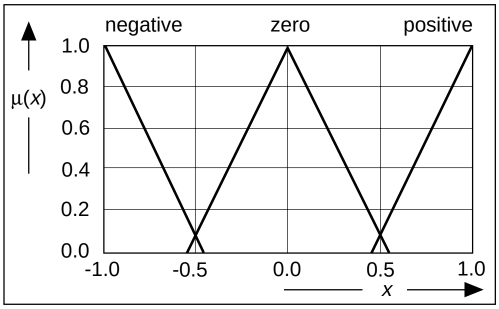
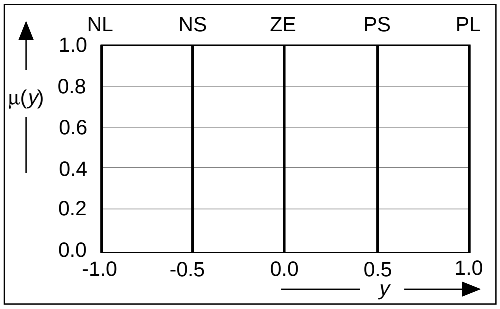
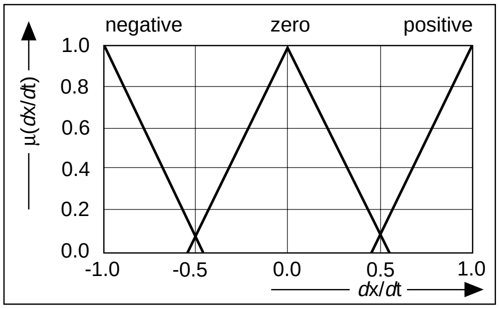


Figure 3-17. I/O Characteristic Field of a Dual Input Fuzzy Controller

Because the minimum operator used in the aggregation step is nonlinear, the characteristic field is not exactly linear despite the entirely overlapping membership functions for both input variables. Nonoverlapping membership functions yield a stepped characteristic field with constant planes, as shown in Figure 3-18.







Rule Base		input x		
		negative	zero	positive
input *dx/dt*	negative	NL	NS	ZE
	zero	NS	ZE	PS
	positive	ZE	PS	PL

Max-Min- Inference	Modified CoA

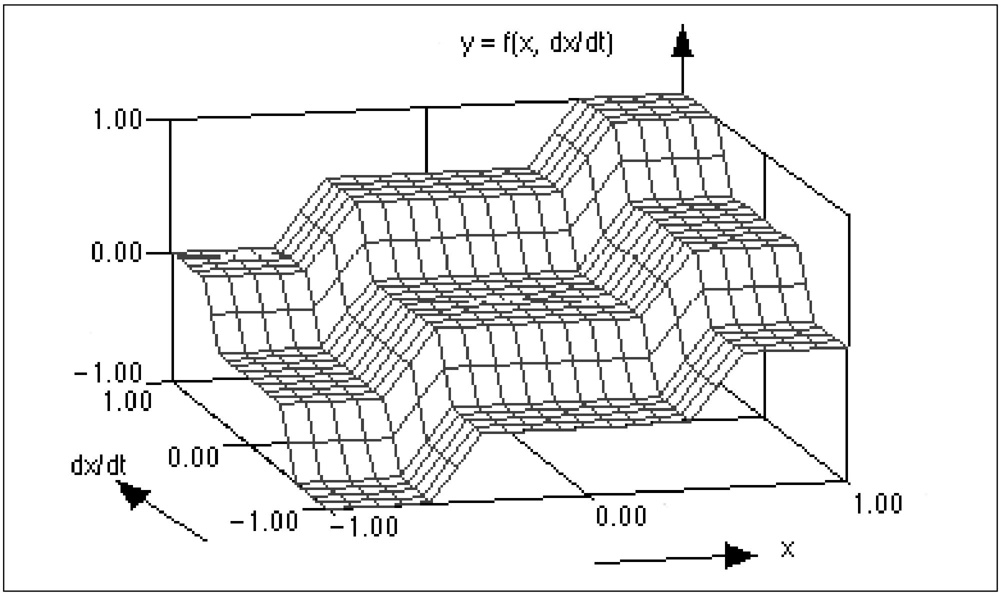


Figure 3-18. I/O Characteristic Field of a Dual Input Fuzzy Controller (Slightly Overlapping Input Terms)

Design Methodology

Chapter 4

This chapter provides an overview of the design methodology of a fuzzy controller.

Design and Implementation Process Overview

Knowledge Acquisition

The knowledge base of a fuzzy controller determines its I/O characteristics and thus the dynamic behavior of the complete closed-loop control circuit. The knowledge base consists of the following:

- Linguistic terms (membership functions) describing the input and output quantities (linguistic variables) of the controller
- Rule base containing the engineering knowledge
- Operators for both the AND and the OR operation
- Fuzzy inference method and the defuzzification method

Within the first system design step, all of the linguistic variables and terms for the given application must be established as the vocabulary of the rule-based system. Use the rule base to formulate the control strategy, then select an appropriate defuzzification method.

Offline Optimization

Within this design step the prototype controller is tested and simulated with either real process data previously recorded from the process or simulation data obtained from a mathematical process model. Transfer characteristics analysis and time response analysis can be performed to observe the system behavior and optimize the controller. LabVIEW and BridgeVIEW support both types of analysis. In this step, Neuro Fuzzy techniques, as well as Genetic or Evolutionary Algorithms, can also be used for system optimization.

Online Optimization

Using the data acquisition capabilities of LabVIEW and BridgeVIEW, you can run the fuzzy controller in conjunction with a process. Then, you can use online optimization techniques to make modifications to the running system.

Implementation

Although you can use the fuzzy controller directly with LabVIEW and BridgeVIEW, real-time performance constraints might make it necessary to download the fuzzy controller to a fast microcontroller board.

Definition of Linguistic Variables

The sensors and actuators of the system to be automated determine the input and output quantities of a fuzzy controller. Each additional quantity measured provides more information about the current process state. Additional sensors can improve accuracy but increase cost.

Fuzzy systems do not require high-precision measurement equipment. In fact, obtaining many values using inexpensive, lower-precision sensors is better than acquiring less data with more expensive, higher-precision sensors. If measuring exact process quantities is too difficult, secondary quantities that reveal less specific process information might be sufficient.

Number of Linguistic Terms

The possible values of a linguistic variable are the linguistic terms which are linguistic interpretations of technical quantities. The quantity vehicle position *x* (usually called the base variable) for example, which is measured in meters, can have the linguistic interpretations *left*, *left center*, *center*, *right center*, and *right*.

When creating a linguistic variable, first determine how many terms define the linguistic variable. In most applications between three and seven terms make up a linguistic variable. It makes no sense to use less than three terms, because most linguistic concepts have at least two extreme terms and a middle term between them. On the other hand, linguistic systems that use more than seven terms are difficult to understand because humans use their short-term memory to interpret technical quantities, and our short-term memory only can compute up to seven symbols simultaneously.

Linguistic variables usually have an odd number of terms because they are defined symmetrically and include a middle term between the extremes.

As a starting point, set up the input variables with at least three or five terms and the output variables with five or seven terms.

Standard Membership Functions

The degree of truth to which a measurement value of a technical quantity satisfies the linguistic concept of a certain term of a linguistic variable is called degree of membership. For a continuous variable the degree of membership can be modeled by a mathematical function.

The normalized standard membership functions illustrated in Figure 4-1 can be applied to most technical processes. These standard functions include Z-type, Λ-type (triangular shape), Π-type (trapezoidal shape), and S-type membership function shapes.

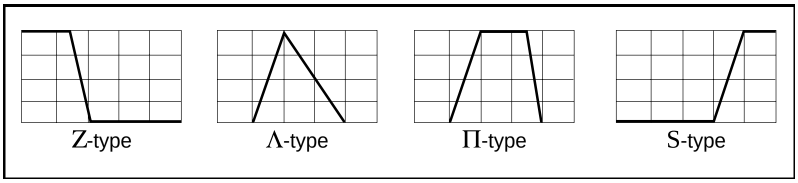


Figure 4-1. Shapes of Standard Membership Functions

To establish standard membership functions, complete the following steps.

1. Define the typical value for each term. This is the value that best fits the linguistic meaning of the term and yields the membership degree $\mu = 1$.
2. For each term, set the membership degree to $\mu = 0$ at the typical values of neighboring terms.
3. Connect the point $\mu = 1$ with the points $\mu = 0$ by straight lines, creating triangular membership function shapes for all inner terms.
4. Because there are no terms beyond the rightmost term and below the leftmost term, all values that fall into this region belong to the respective border term with the membership degree $\mu = 1$.

Figure 4-2 illustrates the design steps mentioned above.

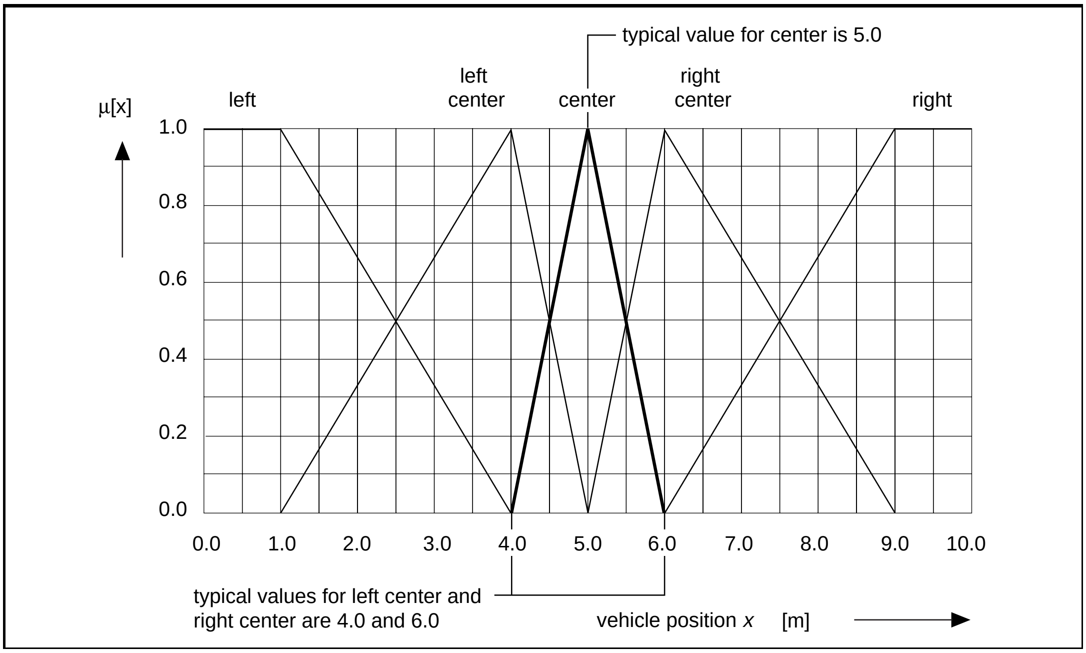


Figure 4-2. Definition of a Triangular Membership Function for the Linguistic Term *Center*

Sometimes the typical value of a term is an interval rather than a crisp value. If, for example, the position *center* is characterized by the statement $x = 5 \pm 0.25$ m, a trapezoidal membership function (Π-type shape) applies, as shown in Figure 4-3.

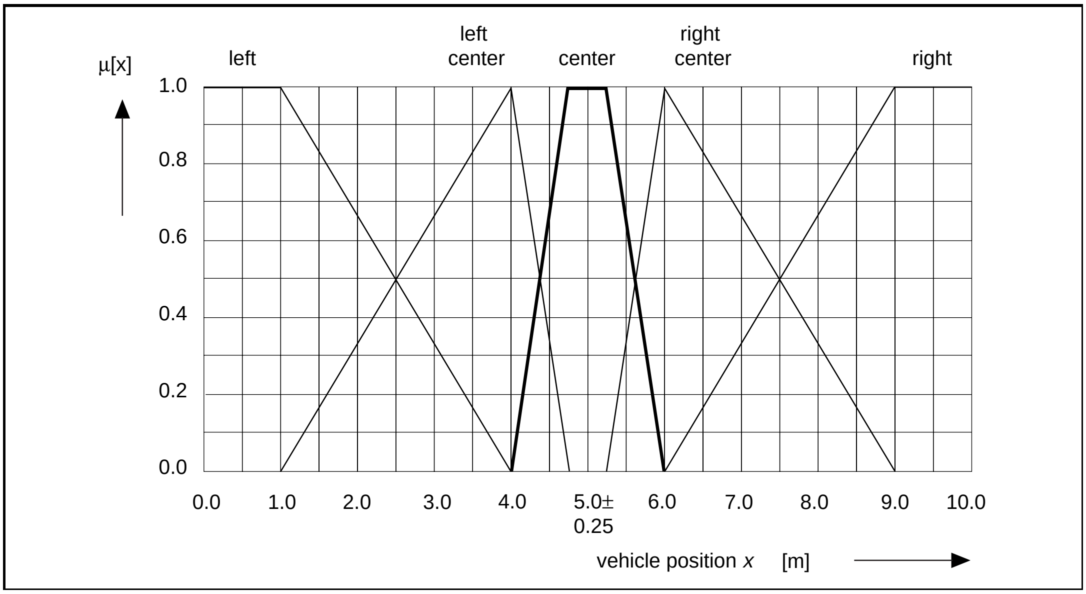


Figure 4-3. Definition of a Trapezoidal Membership Function for the Linguistic Term *Center*

If there is no a priori information available, begin with terms equally spaced within the range of the associated variable, with each term entirely overlapping its neighboring terms. Cover the desired stable region of the system with more linguistic terms that have a small influence interval rather than trying to cover the border regions that have only a few linguistic terms that have a large influence interval. A term distribution like this makes the controller more sensitive within the stable state region of the system.

Disturbance effects on input values such as noise must be taken into account. Do not set up membership functions with an interval of influence that is smaller than the amplitude of the noise signal.

Definition of a Fuzzy Logic Rule Base

The fuzzy logic rule base is the main part of a fuzzy system and contains all the engineering knowledge necessary to control a system. The rule base supplies all the actions to be taken by the fuzzy controller in certain situations. In a sense, the rule base represents the controller's intelligence.

Changes to a single rule have only a local influence on the controller characteristic. Thus you can selectively change the behavior of the fuzzy controller for a certain input situation by modifying a particular rule. Because the modification of a rule usually is carried out in discrete steps by changing its consequence term, it has a much greater influence on the controller characteristic than modifications to the membership functions. To avoid this, implement weight factors (Degrees of Support) for the rules to enhance or reduce the influence of a rule on the controller characteristic.

To build up a rule base, define one rule for each combination of terms (antecedences) of the input variables used in the IF-part of the rule. Then select the most plausible term (conclusion) from the output variable to specify the THEN-part of each rule.

Assuming a fuzzy controller to be built with *m* input variables with *p* terms each, the total number *N* of possible rules is

$$N = p^m$$

p = number of terms for each input variable
m = number of input variables

For example, for three input variables with five terms each, the total number of possible rules is 125. For five input variables with seven terms each, the complete rule base totals 16,807 rules.

Notice that for systems with numerous controller inputs, large rule bases can be avoided by using cascading fuzzy controllers—outputs from fuzzy controllers serve as the inputs to fuzzy controllers of the next layer.

In the case of a fuzzy controller with m input variables, each with an individual number of terms p_i (with $1 \le i \le m$), there are a total of N possible rules according to

$$N = \prod_{i=1}^{m} p_i$$

p_i = number of terms for input variable i
m = number of input variables

This great degree of freedom allows extraordinary design flexibility. However, for large and complex systems it is very difficult to implement the complete rule base. In such cases usually only the rules covering the normal system operation are implemented.

☞ **Note:** ***A fuzzy controller with an incomplete rule base must have a default action value (usually the last command value) for input situations with no active rule.***

A rule base with at least one active rule for each possible combination of crisp input values is called a *complete* rule base. Because there are overlapping regions of the membership functions, an undefined output in a rule base does not necessarily mean that there is no rule active for a certain input situation.

The completeness of a rule base is not the only aspect to be considered when dealing with large rule bases. *Contradicting* rules (rules with the same IF-part but with different THEN-parts) are illogical and should be avoided. Contradicting rules have only a marginal effect on the controller characteristic because of the averaging process that occurs during the defuzzification step. A rule base that is free of contradicting rules is called a *consistent* rule base.

If the rule base is small enough to contain all possible rules, it is not difficult to detect inconsistencies. This is guaranteed for rule bases that can be built in the form of a matrix, as in Figure 2-9, *Complete Linguistic Rule Base*. However, many rule bases are larger and more complex. These are built by beginning with just a few rules to operate input quantities and gradually adding more rules. It is difficult to detect inconsistencies in larger rule bases.

For fuzzy controllers with only two or three input quantities, it is possible to estimate the qualitative controller characteristic just by looking at the rule base. Neighboring terms within a rule matrix with strongly differing meanings like *positive large* and *negative small* indicate steeply sloped edges in the control surface, which usually are not desired. This is referred to as the *continuity* of a rule base. If neighboring rules have the same or similar conclusions, the rule base is said to be *continuous*.

Within large rule bases it is possible to have multiple definitions of the same rule. This is called *redundancy*. It has no influence on the inference result at all if the Max-Min inference method is implemented. But there are other inference methods (not discussed in this manual), such as the Sum-Product method, in which multiple rules can effect the inference result.

Operators, Inference Mechanism, and Defuzzification Method

In closed-loop control applications using fuzzy logic, the standard common operators for the AND- and the OR-operation are the Min- and Max-operators discussed in Chapter 2, *Overview of Fuzzy Logic* (see Figure 2-13, *Default Set of Fuzzy Logic Operators*). Within certain control applications in the field of process technology, however, it might be necessary to use a compensatory AND-operator rather than the pure AND. The most important compensatory AND-operator is the γ-operator (not discussed in detail here) that allows a continuous tuning between AND (no compensation) and OR (full compensation). In real situations the word AND sometimes is used to combine two antecedences more like “as well as,” indicating that a little less of one quantity may be compensated. This is exactly what can be modeled with the γ-operator (also called compensatory AND). Refer to Appendix A, *References*, for a list of documents with more information about this topic.

The standard inference mechanism is the **Max-Min method**. Other inference methods have only a marginal influence on the controller characteristic.

The defuzzification method derives a crisp output value that best represents the linguistic result obtained from the fuzzy inference process. As explained in Chapter 2, *Overview of Fuzzy Logic*, there are generally two different linguistic meanings of the defuzzification process:

- Calculating the **best compromise**—CoM or CoA
- Calculating the **most plausible result**—MoM

An important aspect of the defuzzification method is the continuity of the output signal. Consider a fuzzy logic system with a complete rule base and overlapping membership functions. A defuzzification method is continuous if an arbitrary small change of an input value can never cause an abrupt change in the output signal.

In this respect, the defuzzification methods CoM and CoA are continuous because the best compromise can never jump to a different value with a small change to the inputs (assuming overlapping output membership functions). To the contrary, the defuzzification method MoM is discontinuous because there is always a point at which an arbitrary small change in the input situation of the system will cause a switch to another more plausible result.

Table 4-1. Comparison of Different Defuzzification Methods

Assessment Criteria	**Method**		
	Center-of-Gravity (CoG) **Center-of-Area (CoA)**	**Center-of-Maximum (CoM)**	**Mean-of-Maximum (MoM)**
Linguistic Characteristic	Best Compromise	Best Compromise	Most Plausible Result
Fit with Intuition	Implausible with varying membership function shapes and strong overlapping membership functions	Good	Good
Continuity	Yes	Yes	No
Computational Effort	Very High	Low	Very Low
Application Field	Closed-loop Control, Decision Support, Data Analysis	Closed-loop Control, Decision Support, Data Analysis	Pattern Recognition, Decision Support, Data Analysis

Chapter 5

Using the Fuzzy Logic Controller Design VI

This chapter describes how to design a fuzzy controller using the Fuzzy Logic Controller Design VI.

Overview

The VIs that make up the Fuzzy Logic Controller Design are arranged in four layers of abstraction:

- Project Manager—Maintains a fuzzy logic project
- Fuzzy Set Editor—Defines and modifies linguistic variables including their linguistic terms
- Rule Base Editor—Defines and modifies the rule base of a fuzzy system to be designed
- Testing and project maintenance utilities

Online Help is available by clicking the **Help** button on the active panel.

The following restrictions are valid:

- The maximum number of linguistic variables (controller inputs) is 4.
- The maximum number of linguistic terms for each linguistic variable is 9.
- The types of membership functions are: normalized triangular and trapezoidal membership functions (Z-, Λ-, Π- and S-Type) and singletons.

Project Manager

Start the Project Manager by running the Fuzzy Logic Controller Design VI. Its active front panel is shown in Figure 5-1. This VI differs from most VIs in that it is a standalone application with a graphical user interface for designing and editing a fuzzy controller. The Fuzzy Logic Controller Design VI is set up to run immediately when opened. Although this VI has no inputs or outputs, you can use it as a subVI by placing the icon on your application diagram to allow your user to edit the fuzzy controller programmatically.

The menu bar contains the four topics **File**, **Edit**, **Test**, and **Help**. In the **File** menu (see Figure 5-1) there are several commands for handling the project data. You can activate each command from the **File** menu by double-clicking it. Notice that certain commands are dimmed when unavailable.

As shown in Figure 5-1 the Project Manager front panel has a Project Description Field (indicated by the keyword **description**) into which you can enter important project information. This description contains development ideas and other a priori information for the fuzzy controller to be developed.

In addition to this, there is a Project Identification Field into which the developer can enter his name (input box marked with the key word **developer**). The other entries (controller, date, and time) are processed by the Fuzzy Logic Toolkit. When the project is closed or saved for the first time, the user is prompted to enter a project name which is indicated within the controller indication line of the Project Identification Field when the project is opened later. The default project name is `untitled`, as shown in Figure 5-1.

The **File»New** command creates a new fuzzy logic project. Selecting this command automatically calls the Fuzzy-Set-Editor.

The **File»Open** command opens an existing fuzzy controller for further modifications, and the **File»Close** command closes the current project.

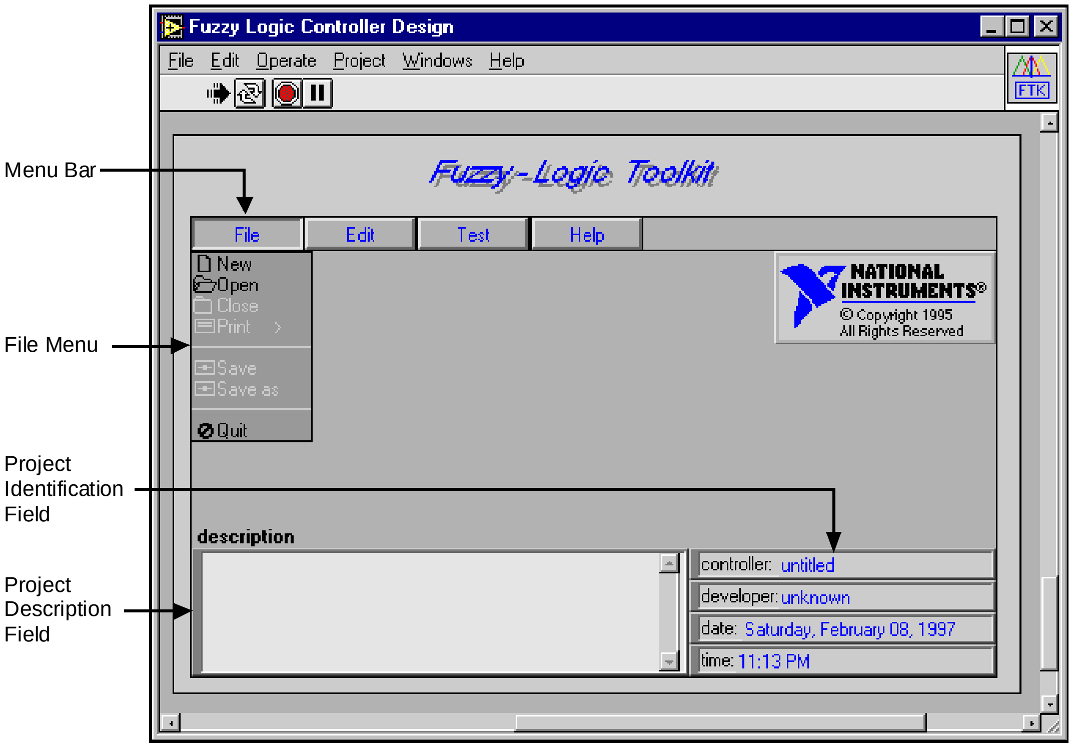


Figure 5-1. Project Manager Front Panel

The **File»Print** command prints out the fuzzy controller documentation. You can choose different print layouts from the **Print** submenu, which opens when the **Print** command is selected. The existence of a submenu is indicated by a > in the **Print** menu.

Many of the commands in the toolkit work similarly to those in LabVIEW or BridgeVIEW. The **File»Save** and **File»Save as** commands store the project data to a file with a `.fc` extension. The **Quit** command exits the application. The **Quit** command checks for unsaved project data and prompts you to save the project if necessary before leaving the application. You can access online help by selecting **Help»Help**.

Fuzzy-Set-Editor

Now, consider designing a fuzzy controller for the truck maneuvering example described in the *Rule-Based Systems* section of Chapter 2, *Overview of Fuzzy Logic*. When you begin a new project, it is best to enter at least a short project description and the name of the developer into the Project Identification Field.

Invoke the Fuzzy-Set-Editor by choosing **File»New**. If there is an existing project already loaded, switch to the Fuzzy-Set-Editor by selecting **Edit»Set-Editor**. The Fuzzy-Set-Editor front panel is shown in Figure 5-2.

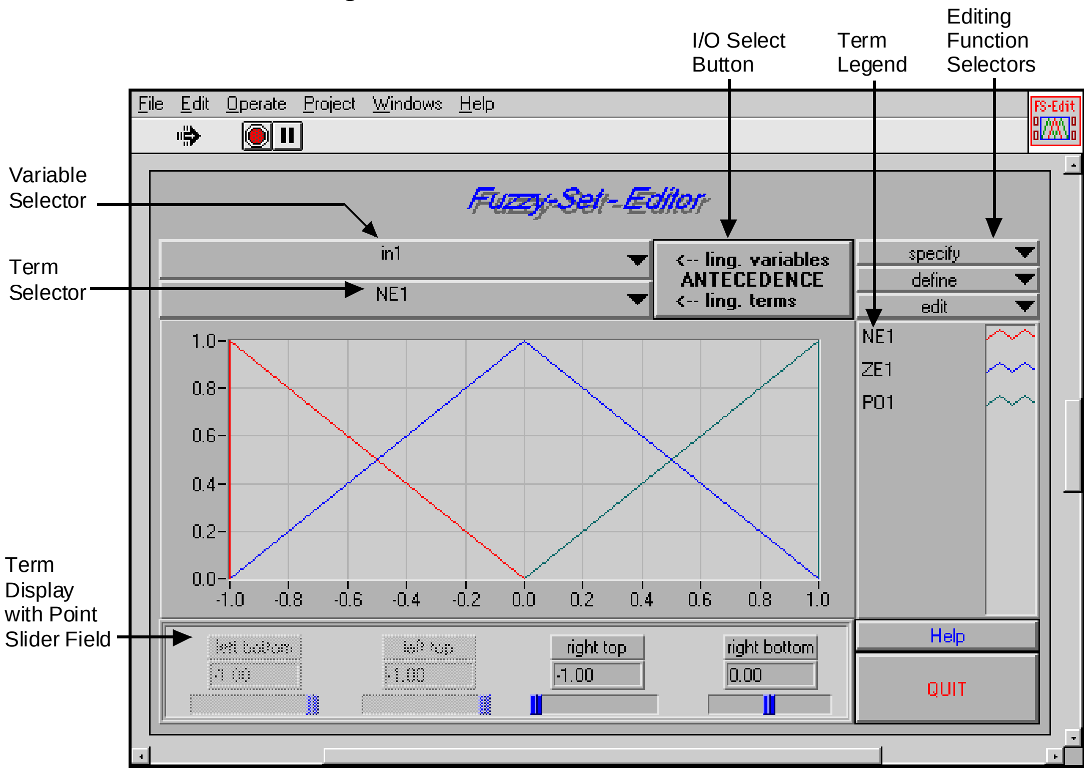


Figure 5-2. Default Fuzzy Controller Settings

A new project always is started with the following default settings:

- Two normalized linguistic input variables (**I/O Select** button switched to ANTECEDENCE) assigned by the default description identifiers input1 and input2. Each input variable ranges from –1.0 to 1.0.
- Each linguistic input variable is composed of three entirely overlapping linguistic terms. For input1, the linguistic terms NE1 (negative), ZE1 (zero), and PO1 (positive); and for input2, the linguistic terms NE2 (negative), ZE2 (zero), and PO2 (positive), are predefined.
- One normalized linguistic output variable (**I/O Select** button switched to CONSEQUENCE) which is assigned to by the default identifier output and composed of the three entirely overlapping linguistic terms NEo (negative), ZEo (zero), and POo (positive). The output variable ranges from –1.0 to 1.0.

All linguistic terms of the linguistic variable that is activated by the Variable Selector are shown in the Term Display, while the term description identifiers are displayed in the Term Legend. (See Figure 5-2.)

You can modify the linguistic term being activated by the Term Selector interactively by adjusting the sliders or input controls from the Point Slider Field.

The Fuzzy-Set-Editor controls modifications to terms with respect to plausibility restrictions. To prevent the user from making implausible term arrangements, all input sliders of term points that cannot be modified because of plausibility restrictions are dimmed. As the example in Figure 5-3 illustrates, you cannot move the left-bottom point or left-top point of the term NE1 below the left-hand range limit of the input variable.

When modifying a term shape by moving a particular point slider, all input sliders are controlled and updated by the Fuzzy-Set-Editor according to plausibility restrictions, too. Thus, the right top value of the term NE1 as shown in Figure 5-3 might not override the left top value of the term ZE1. When moving the right top slider, the Fuzzy-Set-Editor constantly updates this slider according to the plausibility restriction mentioned above so that this point (right top of NE1) cannot exceed the left top of ZE1.

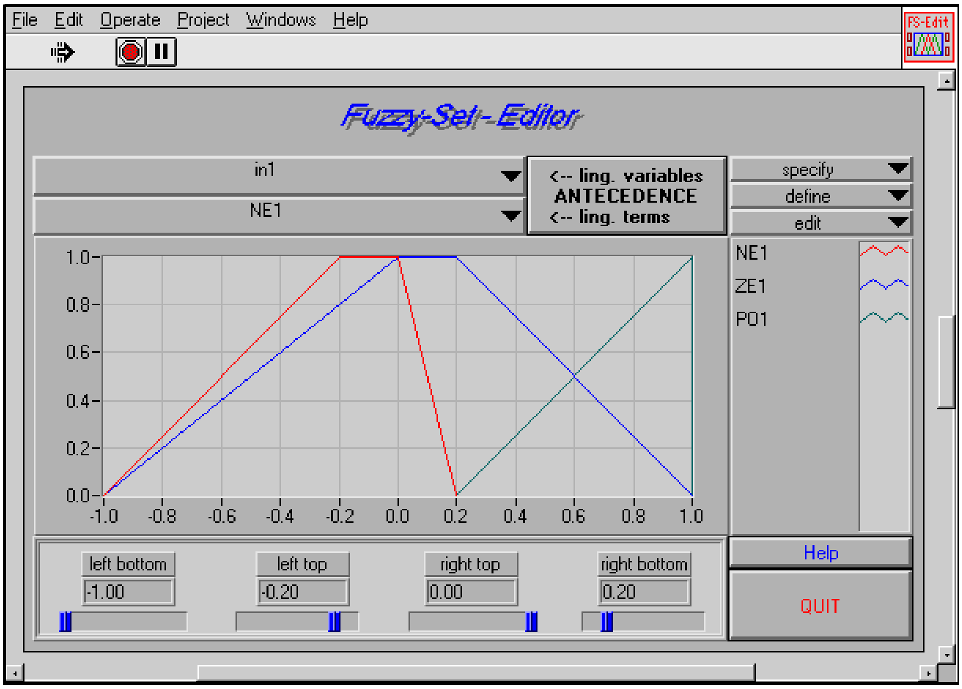


Figure 5-3. Plausibility Checking and Point Slider Movement

In the truck maneuvering example in the *Rule-Based Systems* section of Chapter 2, *Overview of Fuzzy Logic*, there are two linguistic input variables (*vehicle position x* and *vehicle orientation* β) and the linguistic output variable (*steering angle* φ). It is a good idea to use descriptive variable names instead of the default identifiers offered by the Fuzzy-Set-Editor.

To rename the input variables *input1* and *input2* as *vehicle-position* and *vehicle-orientation*, select **specify»rename variable** as shown in Figure 5-4.

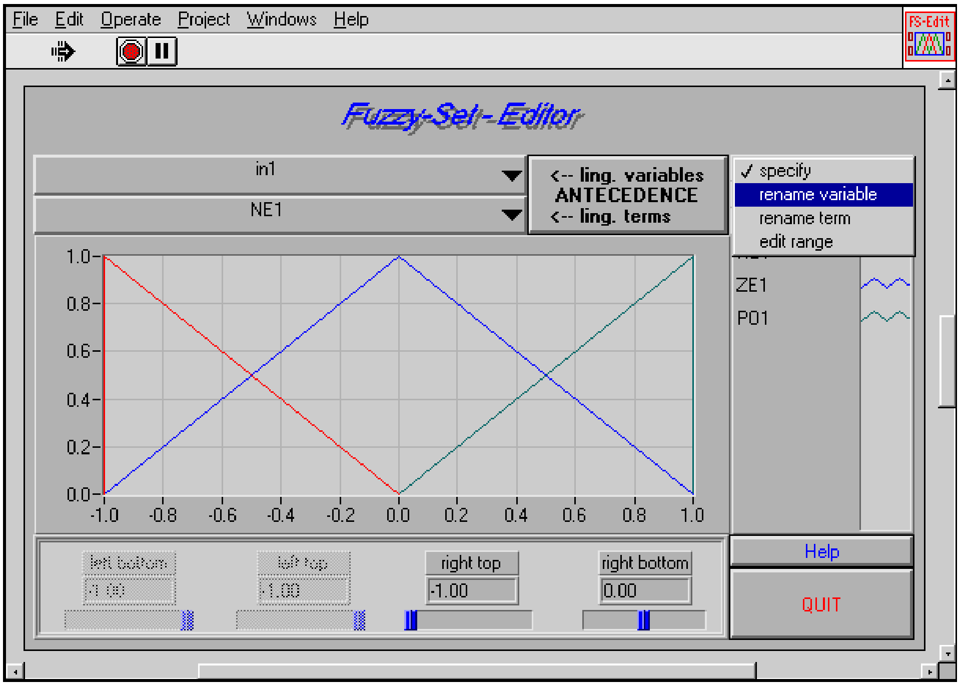


Figure 5-4. Selecting the Rename Variable Command

Now you can change the selected variable identifier **in1** by entering the new description identifier `vehicle-position` into the text input box above the **OK** button (see Figure 5-5). The new variable identifier is saved by either clicking the **OK** button or pressing <Enter>.

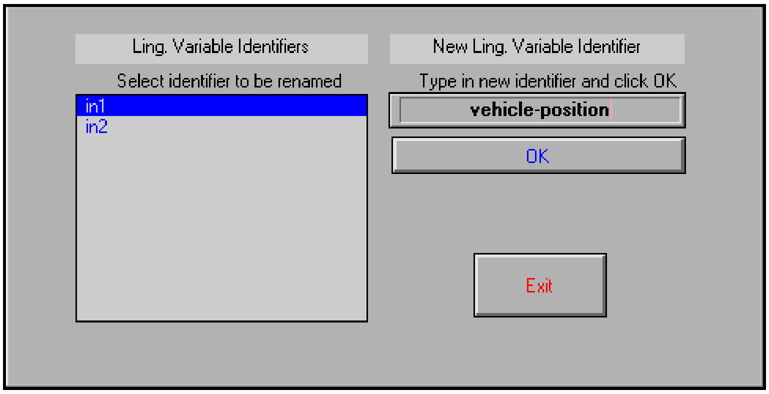

Ling. Variable Identifiers
Select identifier to be renamed
in1
in2
New Ling. Variable Identifier
Type in new identifier and click OK
vehicle-position
OK
Exit

Figure 5-5. Rename Variable Dialog Box

After this, select the variable identifier **in2** and enter the description identifier `vehicle-orientation` into the text input box. Again, the new variable identifier is saved by either clicking the **OK** button or pressing <Enter>.

Complete the **rename variable** command by clicking the **Exit** button on the dialog panel.

To rename the output variable *out* as *steering-angle*, select **ANTECEDENCE/CONSEQUENCE** on the **I/O Select** button to access the output variable. You can rename the variable according to the steps demonstrated above. To finish this step, return the button to the ANTECEDENCE position to be able to access the input variables using the Variable Selector.

The Fuzzy-Set-Editor starts a new project with two input variables, each of which having the default data range interval [–1.0, +1.0]. The variable data ranges must be changed for the truck application example. The vehicle-position ranges from 0.0 to 10.0 meters and the vehicle-orientation from –90.0 to +270.0 degrees.

To change the data range of the input variable vehicle-position, select **specify»edit range,** as shown in Figure 5-6.

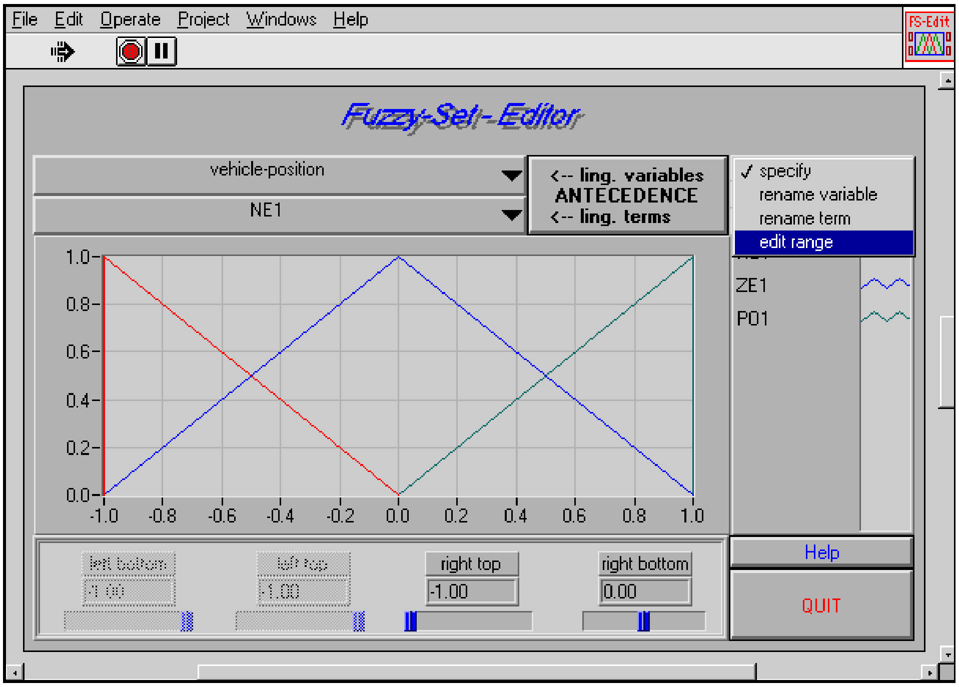


Figure 5-6. Selecting the Edit Range Command

Open the Edit Range dialog box to enter the range boundaries as shown in Figure 5-7.

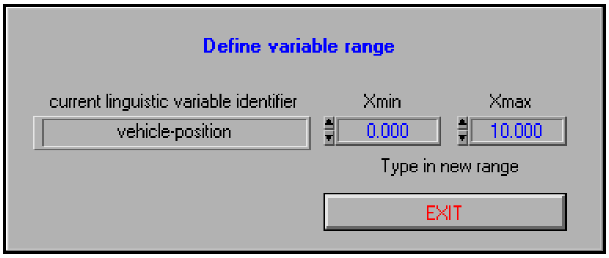


Figure 5-7. Edit Range Dialog Box

Close the dialog box by clicking the **OK** button. Notice that all linguistic terms of the linguistic variable are adapted to the new data range proportionally, as shown in Figure 5-8.

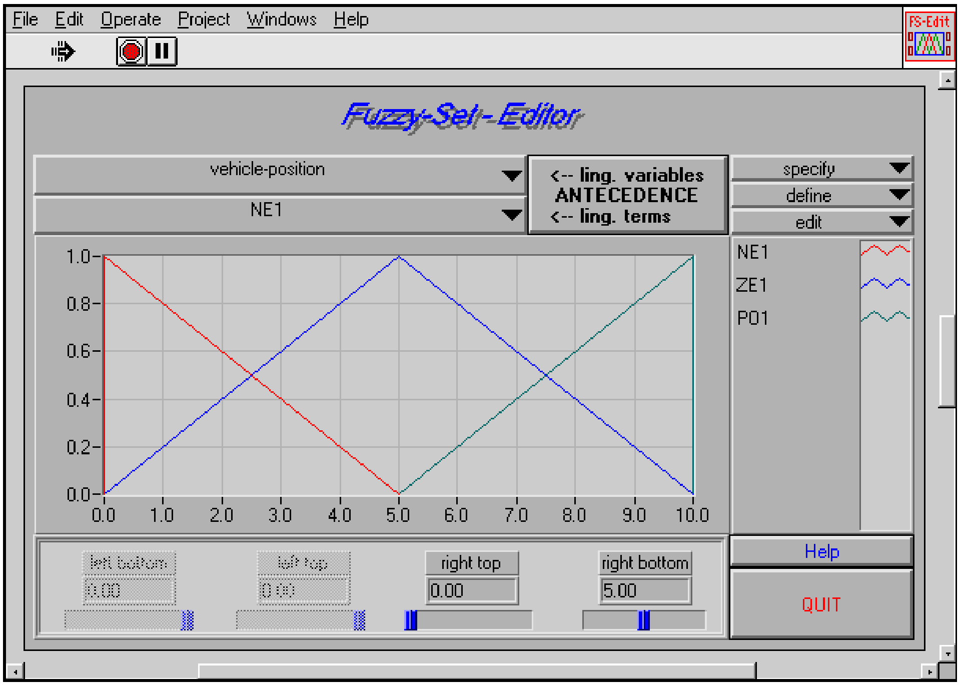


Figure 5-8. Current Input Variable Data Range Changed

For the application example, repeat the steps discussed above to set up the correct data range for the second input variable *vehicle-orientation* and for the output variable *steering-angle***,** which ranges from –30.0 to +30.0 degree.

Figure 5-9 shows the Fuzzy-Set-Editor front panel after setting the correct data range for the output variable. Notice that the **I/O Select** button is in the CONSEQUENCE position.

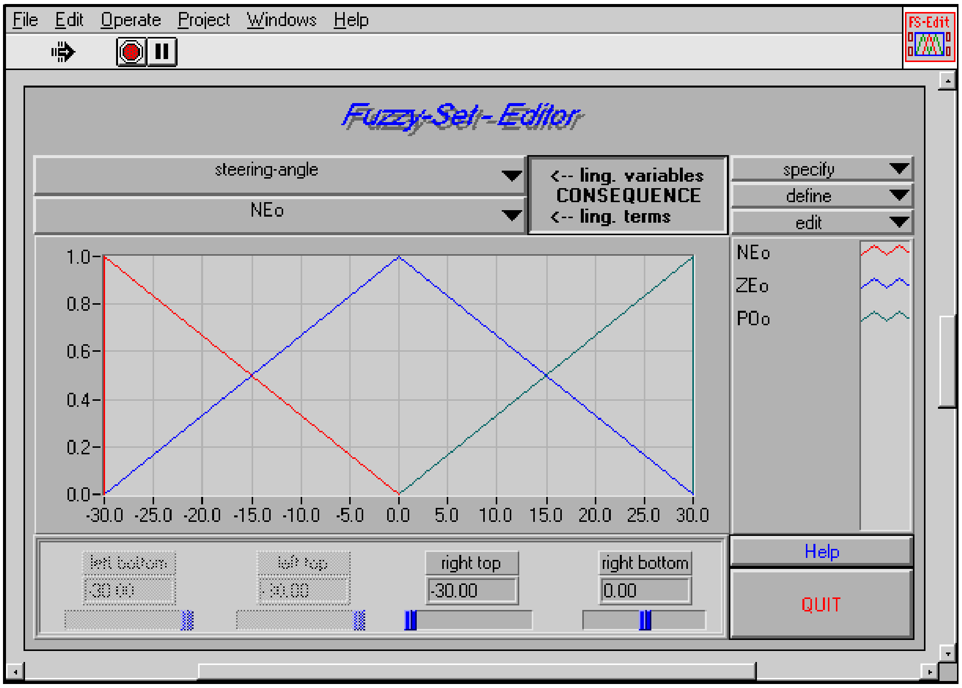


Figure 5-9. Output Variable Data Range Changed

For the next step, you must have access to the input variable *vehicle-position*. Do this by clicking the **I/O Select** button until it is in the ANTECEDENCE position and selecting the desired input variable from the Variable Selector.

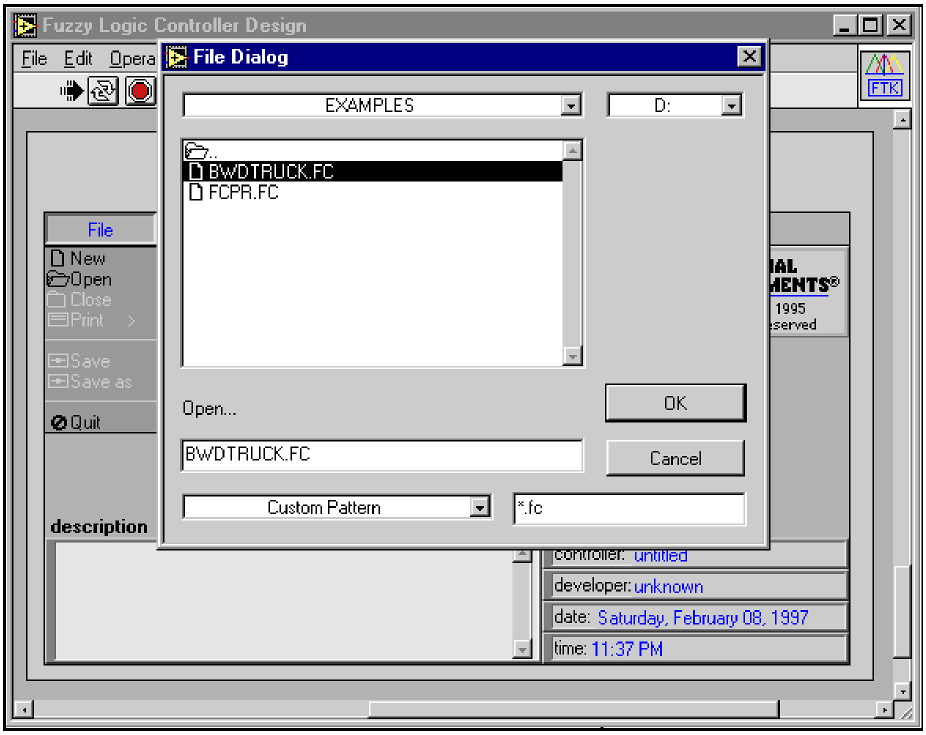
Any modifications made during the Fuzzy-Set-Editor session might have a significant influence on the rule base. It is always a good idea to open the Rulebase-Editor immediately after you close the Fuzzy-Set-Editor. Because you started your Fuzzy-Set-Editor session with a new project, the Rulebase-Editor is called automatically by the Fuzzy Logic Toolkit to create a rule base.

Because you still have to do additional work on the knowledge base, you should add and set up all linguistic terms according to the application example. You do not need to work with the Rulebase-Editor at this point in the project, so exit the Rulebase-Editor by clicking the **QUIT** button immediately after it opens.

When working on an existing project, the Rulebase-Editor is not called automatically when the Fuzzy-Set-Editor is closed. Regardless, closing the Fuzzy-Set-Editor as well as closing the Rulebase-Editor activates the Project Manager.

You can save your project with the **File»Save** or **File»Save as** command. When prompted to enter a file name, type in `FuzzyTruck` as the project name. Notice that fuzzy controller project files always have the extension `.fc`.

Load an existing project that has not yet been loaded using **File»Open** as shown in Figure 5-10.



BWDTRUCK.FC

*.fc

controller: untitled

developer: unknown

date: Saturday, February 08, 1997

time: 11:37 PM

Figure 5-10. Open Command and File Dialog Box

Immediately after a project is loaded by the Project Manager, call the Fuzzy-Set-Editor by selecting **Edit»Set-Editor**. Now the input and output variables have the correct names and data ranges.

The input variable, *vehicle-position*, still is set up by the three entirely overlapping default terms NE1, ZE1, and PO1, as shown in Figure 5-11. Because *vehicle-position* must be composed of the five linguistic terms shown in Figure 2-6, *Linguistic Variable Vehicle Position x and Its Linguistic Terms*, you must add two new linguistic terms. (See the *Rule-Based Systems* section in Chapter 2, *Overview of Fuzzy Logic*.) All linguistic terms must have the same names and shapes so that the complete term arrangement corresponds to that in Figure 2-6.

To add a new linguistic term between the terms NE1 and ZE1, select **define»add term after**, as shown in Figure 5-11.

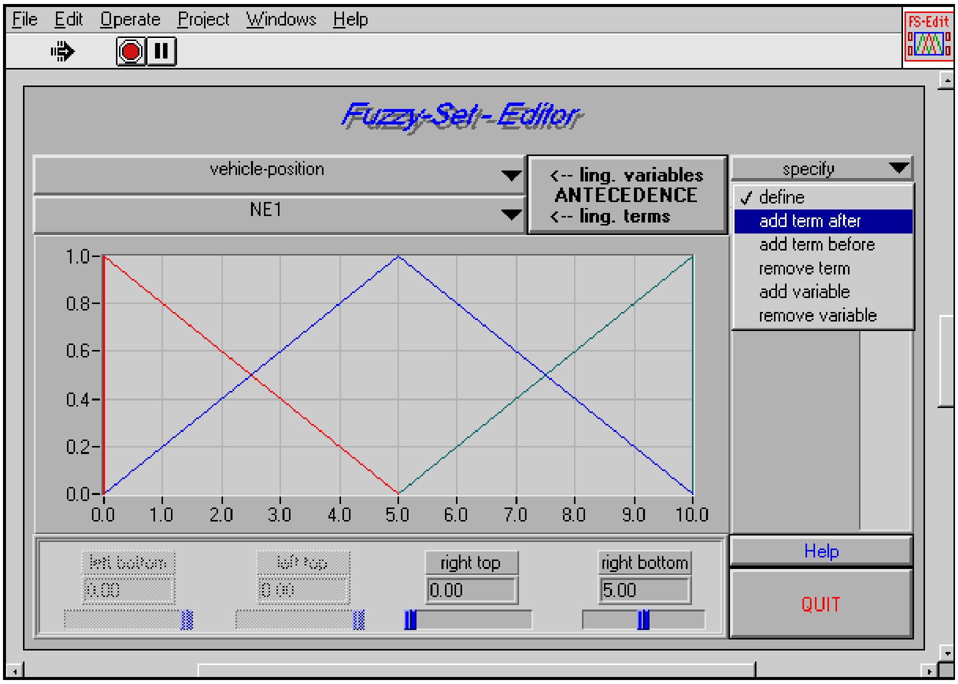
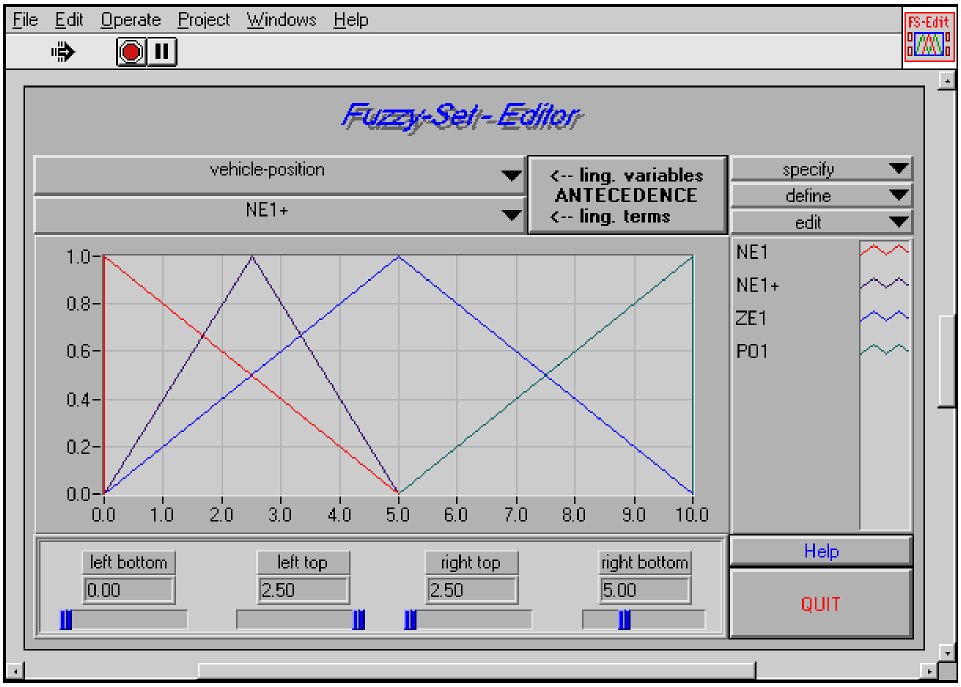


Figure 5-11. Selecting the Add Term After Command

The new linguistic term is located below the active term, as shown in Figure 5-12. The new term identifier is built from the term identifier of the referred term with a + symbol added to the right side.

NE1 is the term identifier of the active term, and the new term is NE1+. Notice that the new term becomes the active term and can be modified immediately.

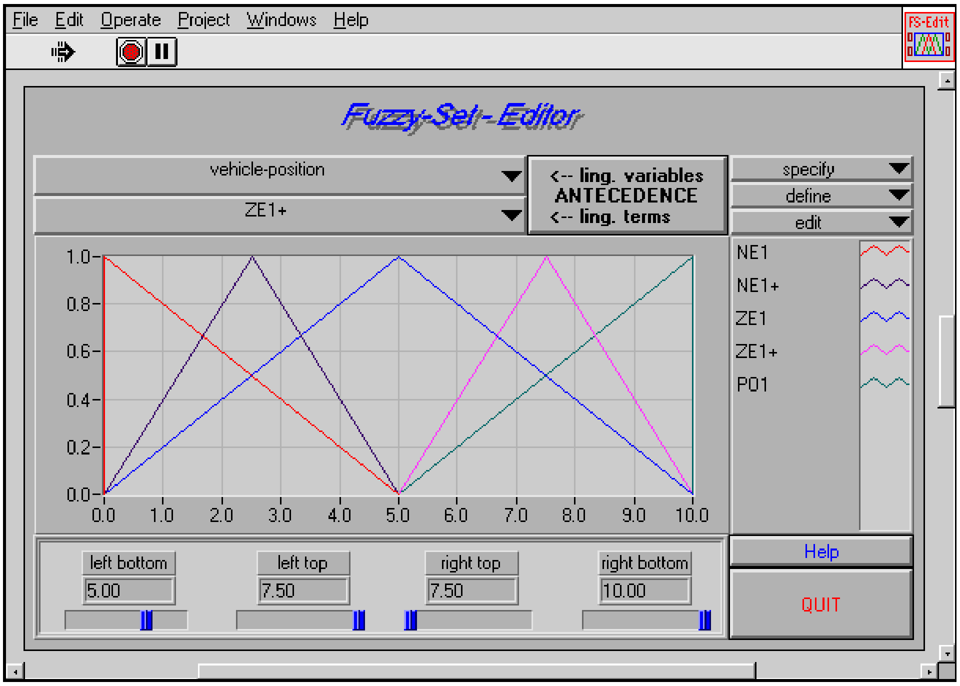


left bottom	left top	right top	right bottom
0.00	2.50	2.50	5.00

Figure 5-12. New Term Added to the Vehicle-Position Variable

☞ **Note:** ***Adding a new term to an input variable, especially of an existing project, causes significant changes to the rule base. The rule base is automatically extended by additional rules, each with a conclusion predefined as* none. *Adding a new consequence term only extends the possibility to select conclusion terms within the Rulebase-Editor. Remember that each input and output variable can have a maximum of nine linguistic terms (checked automatically).***

To add the second new term between ZE1 and PO1, first select **ZE1** from the Term Selector. With ZE1 as the active term, you can select **define»add term after** again. The new term, ZE1+, is added to the Term Display, as shown in Figure 5-13.



left bottom	left top	right top	right bottom
5.00	7.50	7.50	10.00

Figure 5-13. Another New Term Added to the Vehicle-Position Variable

Before rearranging the linguistic terms according to the desired pattern (see Figure 2-6, *Linguistic Variable Vehicle Position x and Its Linguistic Terms*), you can assign the correct term identifiers first by selecting **specify»rename term**. Figure 5-14 shows an intermediate state and Figure 5-15 shows the final result of this renaming.

Note: ***The specify menu also can be used to add or remove linguistic variables (controller inputs).***

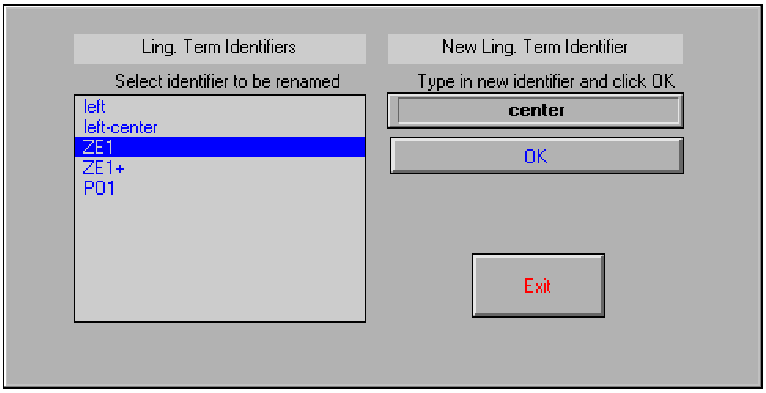

Ling. Term Identifiers
Select identifier to be renamed
left
left-center
ZE1
ZE1+
PO1
New Ling. Term Identifier
Type in new identifier and click OK
center
OK
Exit

Figure 5-14. Rename Term Dialog Box

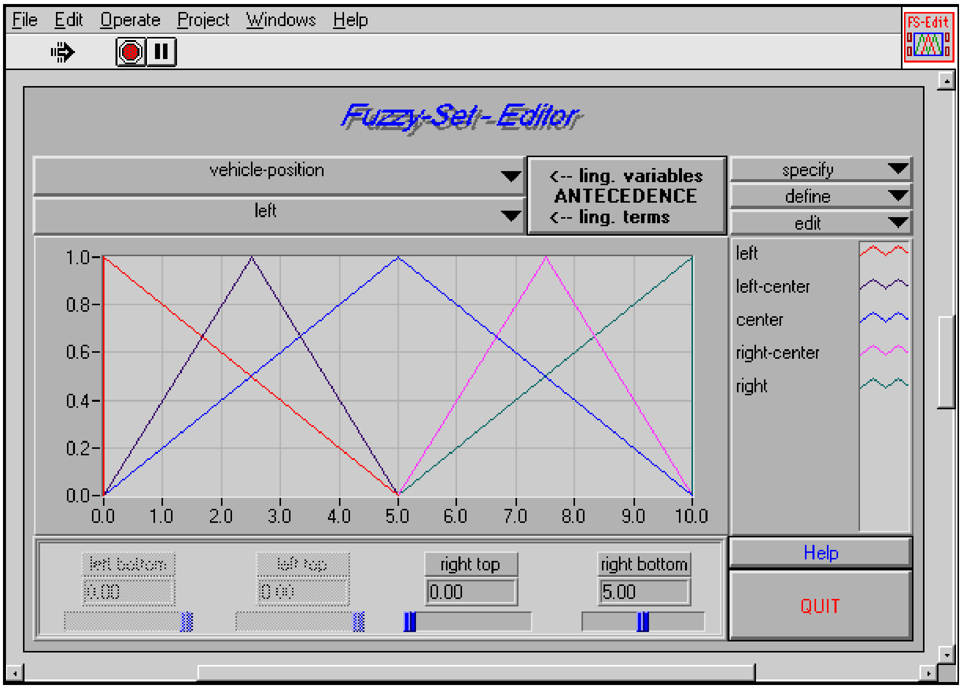


Figure 5-15. All Vehicle-Position Terms Named Correctly

The Fuzzy-Set-Editor offers many functions to modify single terms or the whole term arrangement of the active variable. It is a good idea to experiment with this function at this point in your project because you must modify the whole term arrangement according to the desired term arrangement shown in Figure 2-6, *Linguistic Variable Vehicle Position x and Its Linguistic Terms*. Figures 5-16 and 5-17 show the term arrangement obtained by selecting **edit»full term-overlap all**, resulting in a term arrangement with all terms of the active linguistic variable completely overlapping each other.

The **edit** menu also has several other functions for editing membership functions automatically. You can change individual membership functions, or all of them, to singleton fuzzy sets (typically used for controller output only). The tolerance function changes a trapezoidal membership to a triangular function. In addition, there are options to set the overlap between functions and to make all functions symmetric. The left side of the left-most term and the right side of the right-most term are not changed by this command.

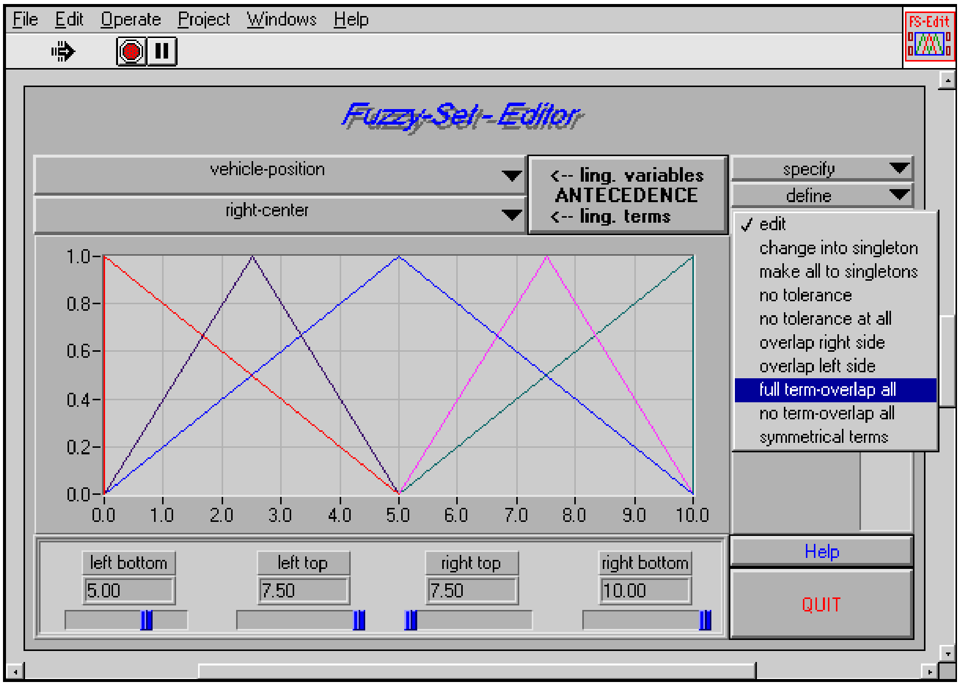


Figure 5-16. Selecting the Full Term-Overlap All Command

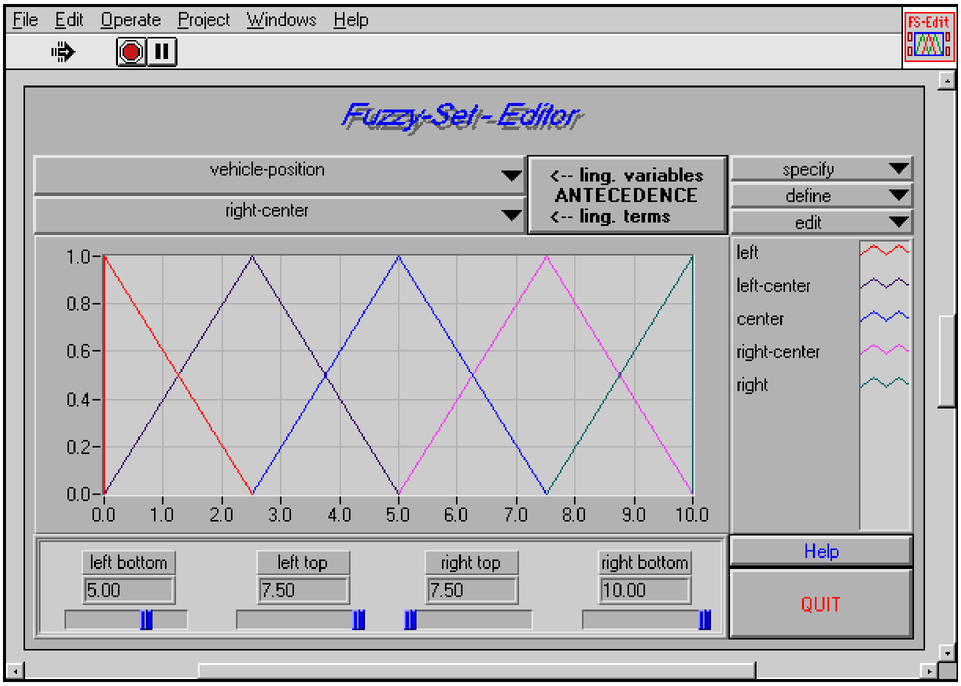
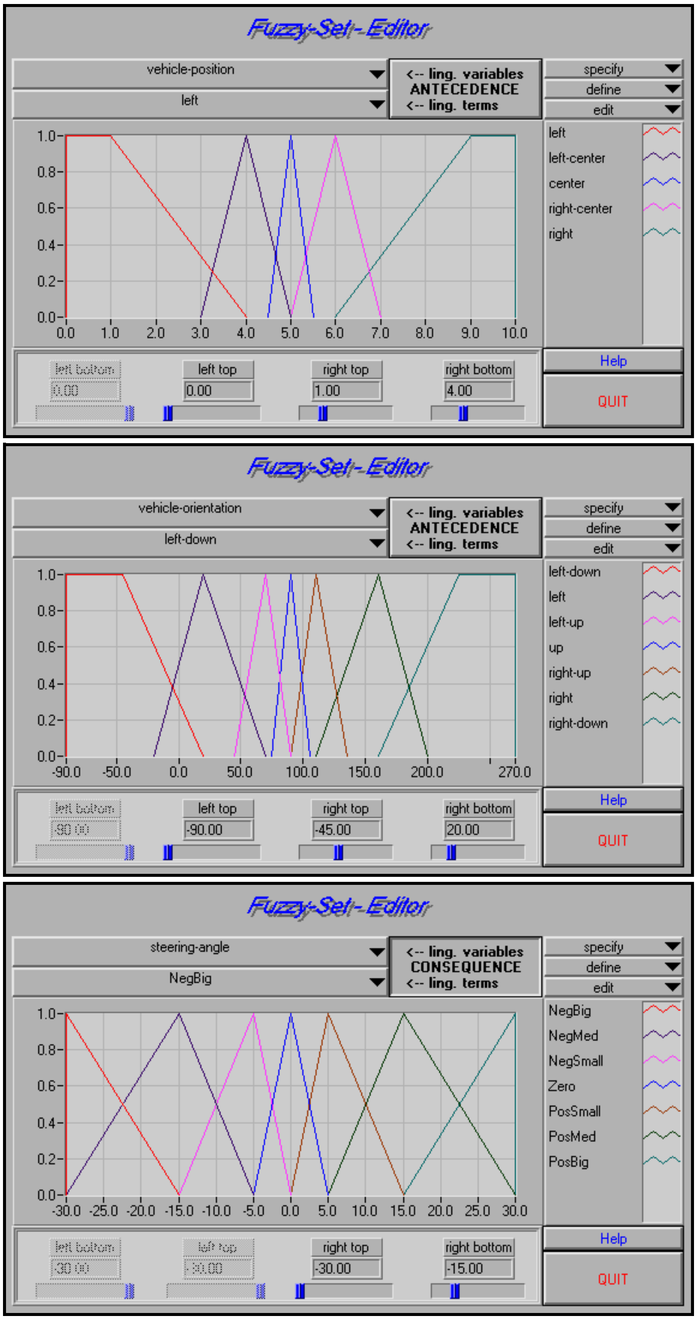


Figure 5-17. A Term Arrangement of Completely Overlapping Terms

With the Fuzzy-Set-Editor functions described above, you can edit all linguistic variables, including the desired term arrangements for the `FuzzyTruck` example project. Figure 5-18 shows the result of the complete editing session.



vehicle-position
left

left bottom	left top	right top	right bottom
0.00	0.00	1.00	4.00

vehicle-orientation
left-down

left bottom	left top	right top	right bottom
-90.00	-90.00	-45.00	20.00

steering-angle
NegBig

left bottom	left top	right top	right bottom
-30.00	-30.00	-30.00	-15.00

Figure 5-18. Results of the Complete Editing Session Example

Rulebase-Editor

After you have entered all the linguistic information of the application example to your `FuzzyTruck` project, you can begin editing. The rule base represents expert knowledge about the vehicle maneuvering process.

If it is not active already, load the example project, `FuzzyTruck`, by selecting **File»Open**. Open the Rulebase-Editor by selecting **Edit»Rulebase**.

Because you have not explicitly entered or modified a rule at this point in the example project, the Rulebase-Editor starts with a project-specific complete default rule base, as shown in Figure 5-19.

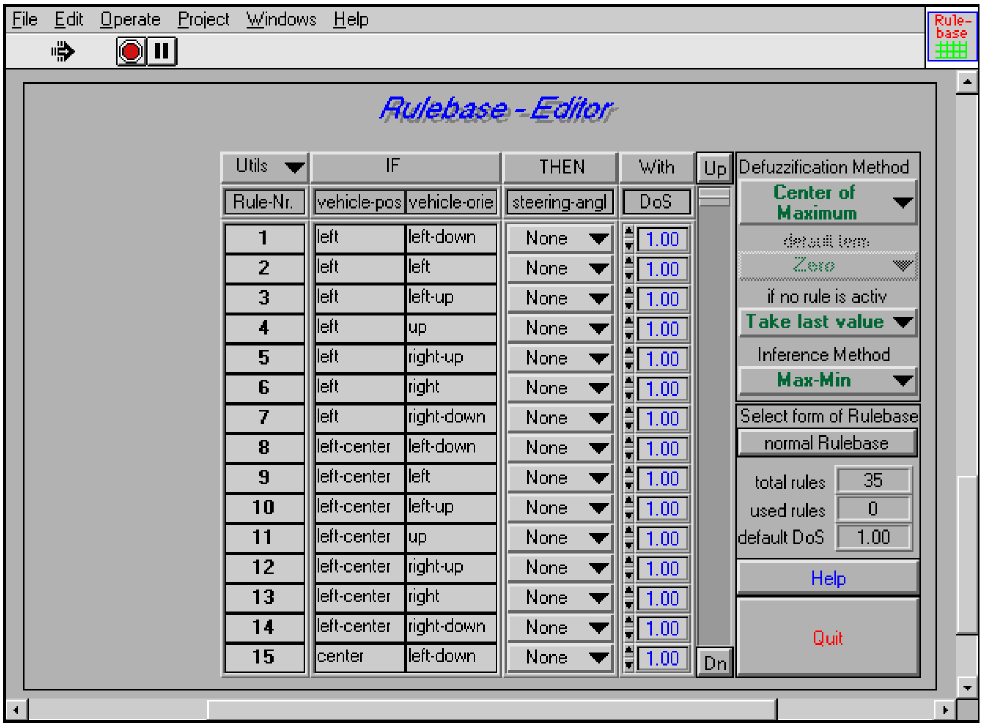
Each possible combination of linguistic terms (antecedences) of the input variables is assigned to a single rule with its *consequence* part (also called conclusion) set to **none**.

Because there are five terms for the first input variable (*vehicle-position*) and seven for the second input variable (*vehicle-orientation*) the rule base offered by the Rulebase-Editor contains 35 rules.

If there are more than fifteen rules available, a scrollbar, as shown in Figure 5-20, is activated to access the rules not currently displayed on the Rulebase-Editor front panel.

Each rule is associated with a weight factor, or Degree of Support (DoS), to enhance or reduce the influence of a rule on the controller characteristic. The DoS ranges between 0.0 and 1.0. In a default rule base, all DoS values are set to 1.0 automatically. You can use the **Utils** menu to set weights for all rules.

You can use weight factors in combination with techniques, such as genetic algorithms, to optimize controller performance.



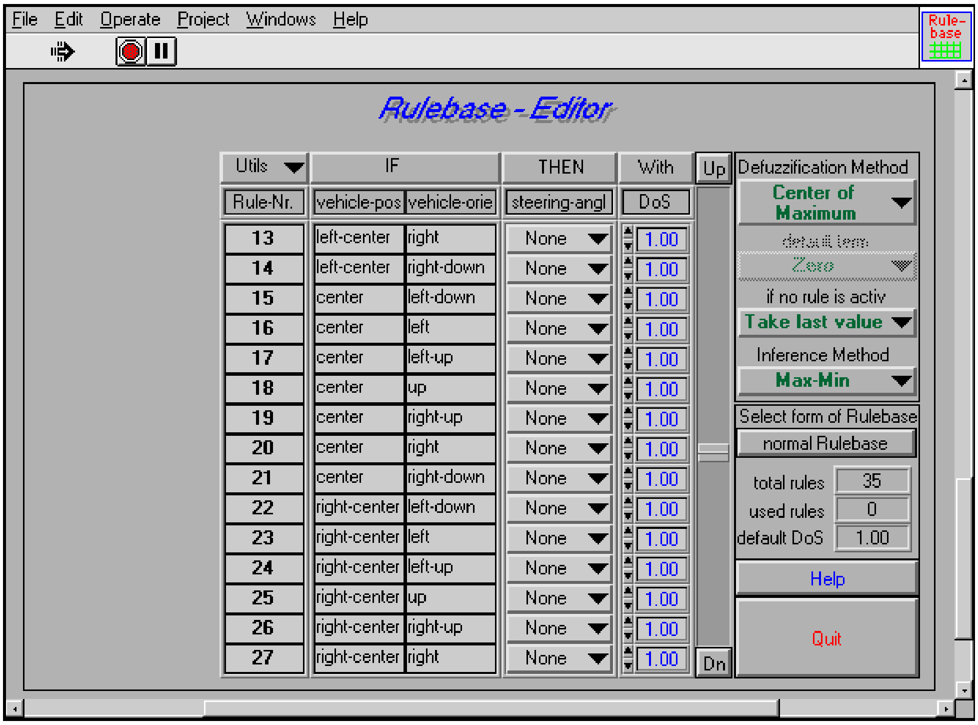
IF			THEN	With
Rule-Nr.	vehicle-pos	vehicle-orie	steering-angl	DoS
1	left	left-down	None	1.00
2	left	left	None	1.00
3	left	left-up	None	1.00
4	left	up	None	1.00
5	left	right-up	None	1.00
6	left	right	None	1.00
7	left	right-down	None	1.00
8	left-center	left-down	None	1.00
9	left-center	left	None	1.00
10	left-center	left-up	None	1.00
11	left-center	up	None	1.00
12	left-center	right-up	None	1.00
13	left-center	right	None	1.00
14	left-center	right-down	None	1.00
15	center	left-down	None	1.00

Defuzzification Method: Center of Maximum
default term: Zero
if no rule is activ: Take last value
Inference Method: Max-Min
Select form of Rulebase: normal Rulebase

total rules	35
used rules	0
default DoS	1.00

Figure 5-19. Project-Specific Complete Default Rule Base

The Rulebase-Editor panel also contains menu buttons for selecting the defuzzification method and inference method interactively. The default controller output also can be changed for situations with no active rules, if the default setting does not fit the application needs.



Utils	IF		THEN	With
Rule-Nr.	vehicle-pos	vehicle-orie	steering-angl	DoS
13	left-center	right	None	1.00
14	left-center	right-down	None	1.00
15	center	left-down	None	1.00
16	center	left	None	1.00
17	center	left-up	None	1.00
18	center	up	None	1.00
19	center	right-up	None	1.00
20	center	right	None	1.00
21	center	right-down	None	1.00
22	right-center	left-down	None	1.00
23	right-center	left	None	1.00
24	right-center	left-up	None	1.00
25	right-center	up	None	1.00
26	right-center	right-up	None	1.00
27	right-center	right	None	1.00

Defuzzification Method: Center of Maximum
default term: Zero
if no rule is activ: Take last value
Inference Method: Max-Min
Select form of Rulebase: normal Rulebase

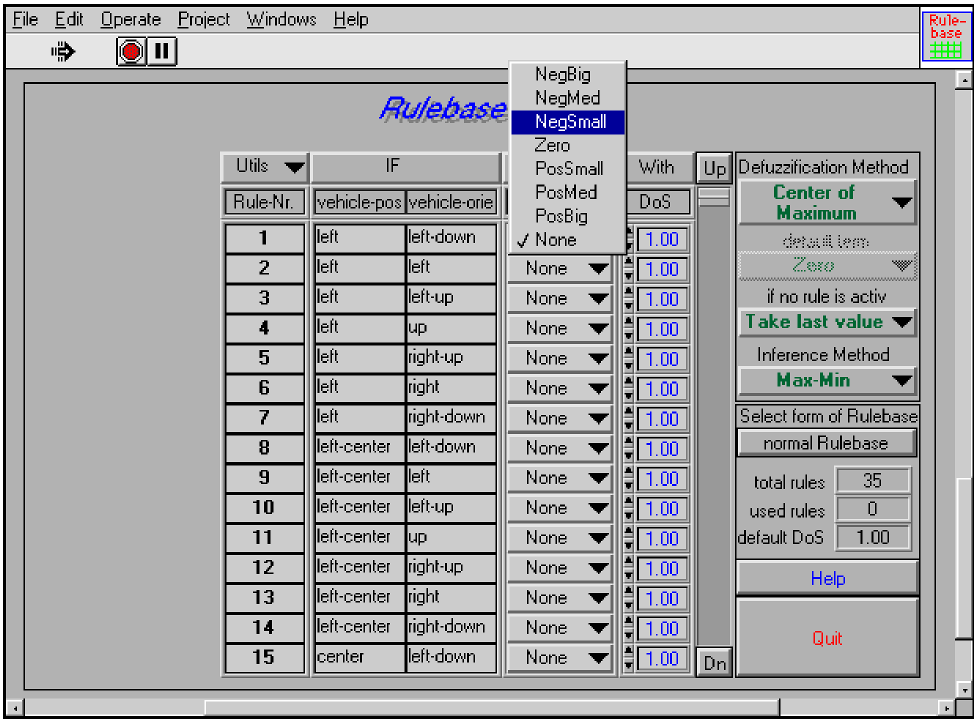
total rules	35
used rules	0
default DoS	1.00

Figure 5-20. Using the Rulebase-Editor Scrollbar

Start editing the rule base by entering the desired consequence to each rule. The consequence part of each rule is implemented as a term selection box containing all possible consequence terms. You can specify the consequence of a particular rule by selecting the desired consequence term from the term selection box.

According to the rule base specified in Figure 2-9, *Complete Linguistic Rule Base*, if the vehicle position is *left* and the vehicle orientation is *left down*, the consequence term is *negative small*. By selecting **NegSmall** from the term selection box of the consequence part (THEN part), as shown in Figure 5-21, the first rule of the rule base is now specified as

IF vehicle-position is *left* **AND** vehicle-orientation is *left down*, **THEN** set steering-angle to *negative small*.



Utils	IF			With
Rule-Nr.	vehicle-pos	vehicle-orie		DoS
1	left	left-down		1.00
2	left	left	None	1.00
3	left	left-up	None	1.00
4	left	up	None	1.00
5	left	right-up	None	1.00
6	left	right	None	1.00
7	left	right-down	None	1.00
8	left-center	left-down	None	1.00
9	left-center	left	None	1.00
10	left-center	left-up	None	1.00
11	left-center	up	None	1.00
12	left-center	right-up	None	1.00
13	left-center	right	None	1.00
14	left-center	right-down	None	1.00
15	center	left-down	None	1.00

Defuzzification Method: Center of Maximum
default term: Zero
if no rule is activ: Take last value
Inference Method: Max-Min
Select form of Rulebase: normal Rulebase

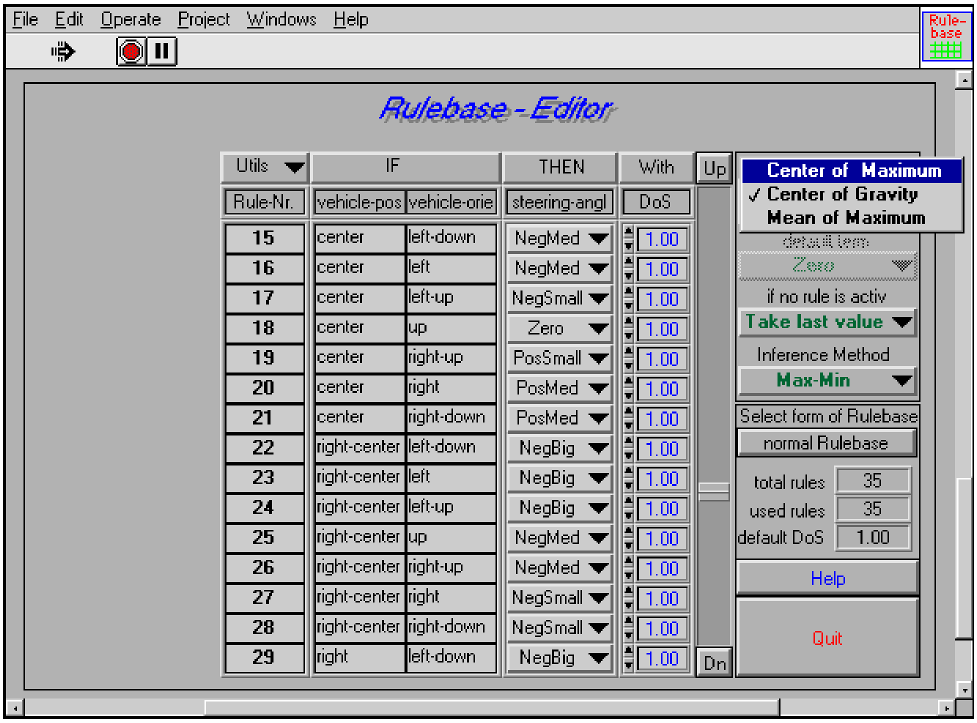
total rules	35
used rules	0
default DoS	1.00

Figure 5-21. Selecting Negative Small as the Consequence Term of the First Rule

The complete rule base can be entered this way. The IF part of the Rulebase-Editor panel automatically accommodates the number of input variables used in the fuzzy controller.

Next, select an appropriate defuzzification method. Because there must be a continuous output signal for the steering angle control, you must select a defuzzification method that calculates the best compromise. Following the guidelines in Table 4-1, *Comparison of Different Defuzzification Methods*, in Chapter 4, *Design Methodology*, you can use either the Center-of-Maximum method or the Center-of-Area method.

Select the defuzzification method from the appropriate selector on the Rulebase-Editor panel, as shown in Figure 5-22.

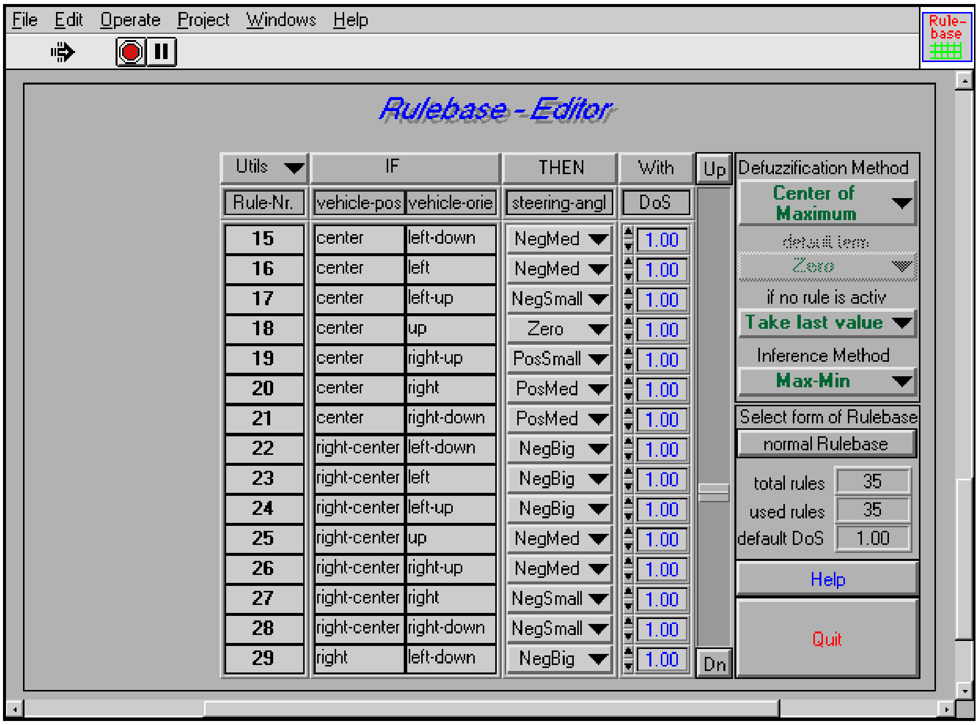


Utils	IF		THEN	With
Rule-Nr.	vehicle-pos	vehicle-orie	steering-angl	DoS
15	center	left-down	NegMed	1.00
16	center	left	NegMed	1.00
17	center	left-up	NegSmall	1.00
18	center	up	Zero	1.00
19	center	right-up	PosSmall	1.00
20	center	right	PosMed	1.00
21	center	right-down	PosMed	1.00
22	right-center	left-down	NegBig	1.00
23	right-center	left	NegBig	1.00
24	right-center	left-up	NegBig	1.00
25	right-center	up	NegMed	1.00
26	right-center	right-up	NegMed	1.00
27	right-center	right	NegSmall	1.00
28	right-center	right-down	NegSmall	1.00
29	right	left-down	NegBig	1.00

default term: Zero
if no rule is activ: Take last value
Inference Method: Max-Min
Select form of Rulebase: normal Rulebase
total rules 35
used rules 35
default DoS 1.00

Figure 5-22. Selecting a Defuzzification Method

The default setting shown in Figure 5-23 can be used as the default controller output. The default setting does not affect the application example because the fuzzy controller has a complete rule base and overlapping term arrangements. In the example, no input variables have definition gaps or undefined intervals. See Figure 3-10, *I/O Characteristic of a Fuzzy Controller (Undefined Input Term Interval).*



Rule-Nr.	IF vehicle-pos	IF vehicle-orie	THEN steering-angl	With DoS
15	center	left-down	NegMed	1.00
16	center	left	NegMed	1.00
17	center	left-up	NegSmall	1.00
18	center	up	Zero	1.00
19	center	right-up	PosSmall	1.00
20	center	right	PosMed	1.00
21	center	right-down	PosMed	1.00
22	right-center	left-down	NegBig	1.00
23	right-center	left	NegBig	1.00
24	right-center	left-up	NegBig	1.00
25	right-center	up	NegMed	1.00
26	right-center	right-up	NegMed	1.00
27	right-center	right	NegSmall	1.00
28	right-center	right-down	NegSmall	1.00
29	right	left-down	NegBig	1.00

Defuzzification Method: Center of Maximum
default term: Zero
if no rule is activ: Take last value
Inference Method: Max-Min
Select form of Rulebase: normal Rulebase

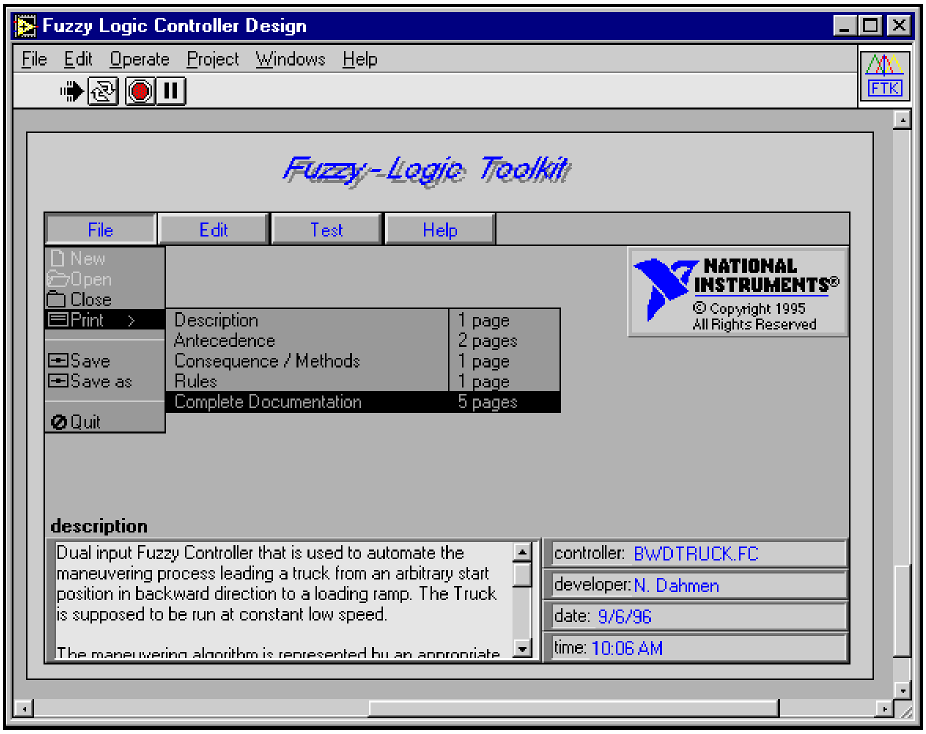
total rules	35
used rules	35
default DoS	1.00

Figure 5-23. Default Settings for Default Controller Output and Inference Method

Now you have completed the design work for the example project. It is time to save the project and to see what documentation features are available within the toolkit.

Documenting Fuzzy Control Projects

The **File»Print** submenu offers documentation facilities for printing information about the active project. Select **Print»Complete Documentation** to print the complete controller documentation for the example project, as shown in Figure 5-24.



Description	1 page
Antecedence	2 pages
Consequence / Methods	1 page
Rules	1 page
Complete Documentation	5 pages

description

Dual input Fuzzy Controller that is used to automate the maneuvering process leading a truck from an arbitrary start position in backward direction to a loading ramp. The Truck is supposed to be run at constant low speed.

The maneuvering algorithm is represented by an appropriate

controller: BWDTRUCK.FC
developer: N. Dahmen
date: 9/6/96
time: 10:06 AM

Figure 5-24. Selecting Complete Documentation from the File Menu

You can leave the Fuzzy Logic Toolkit by selecting **File»Quit** without having saved the project since the last changes.

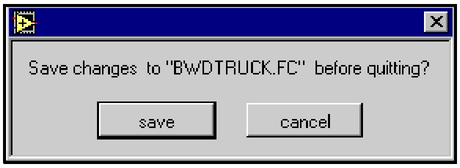
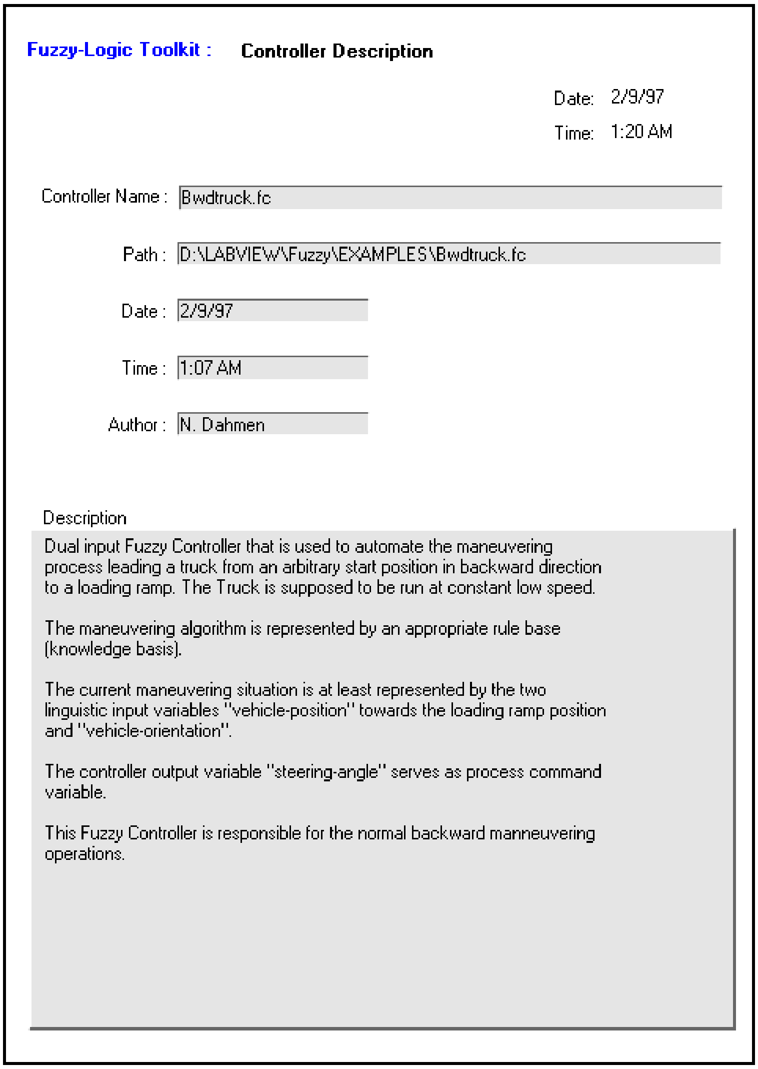

Save changes to "BWDTRUCK.FC" before quitting?

Figure 5-25. Save Changes Dialog Box

Figures 5-26 through 5-30 show printed pages from the example project.



Fuzzy-Logic Toolkit : **Controller Description**

Date: 2/9/97

Time: 1:20 AM

Controller Name :	Bwdtruck.fc
Path :	D:\LABVIEW\Fuzzy\EXAMPLES\Bwdtruck.fc
Date :	2/9/97
Time :	1:07 AM
Author :	N. Dahmen

Description

Dual input Fuzzy Controller that is used to automate the maneuvering process leading a truck from an arbitrary start position in backward direction to a loading ramp. The Truck is supposed to be run at constant low speed.

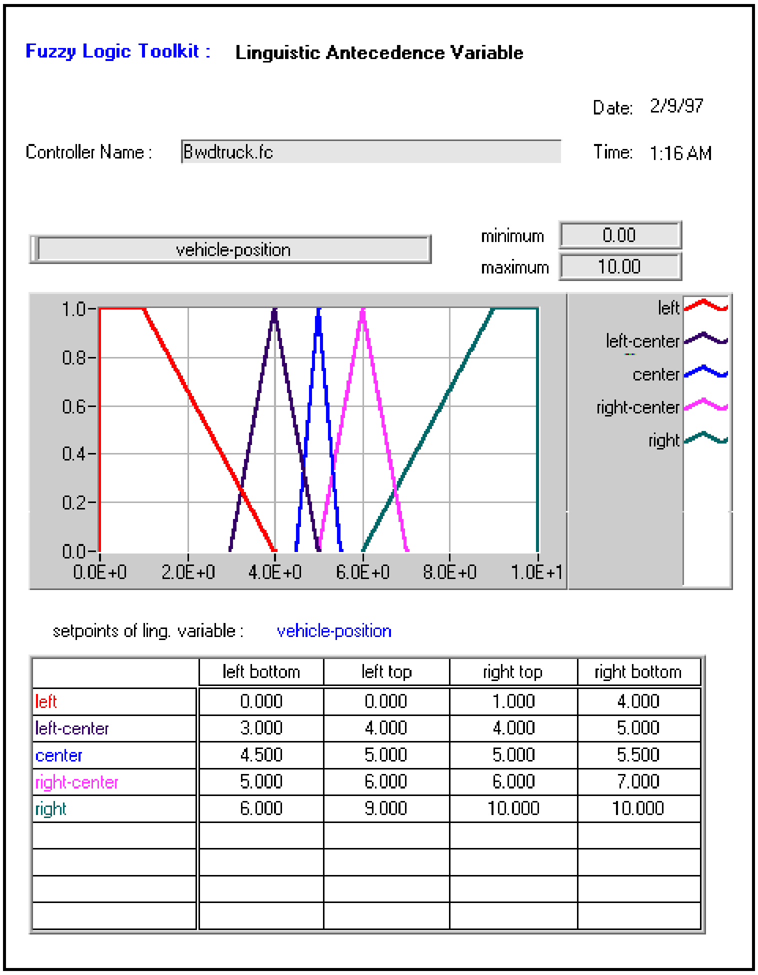
The maneuvering algorithm is represented by an appropriate rule base (knowledge basis).

The current maneuvering situation is at least represented by the two linguistic input variables "vehicle-position" towards the loading ramp position and "vehicle-orientation".

The controller output variable "steering-angle" serves as process command variable.

This Fuzzy Controller is responsible for the normal backward manneuvering operations.

Figure 5-26. Print Page Project Description



Fuzzy Logic Toolkit : **Linguistic Antecedence Variable**

Date: 2/9/97

Controller Name : Bwdtruck.fc

Time: 1:16 AM

vehicle-position

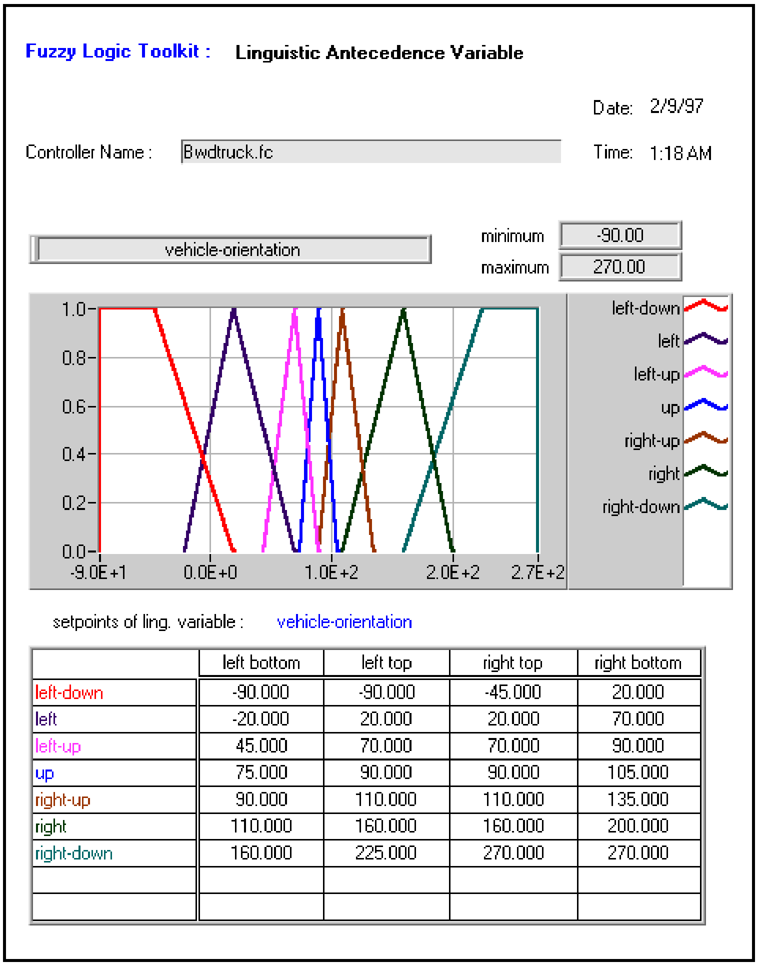
minimum 0.00

maximum 10.00

setpoints of ling. variable : vehicle-position

	left bottom	left top	right top	right bottom
left	0.000	0.000	1.000	4.000
left-center	3.000	4.000	4.000	5.000
center	4.500	5.000	5.000	5.500
right-center	5.000	6.000	6.000	7.000
right	6.000	9.000	10.000	10.000

Figure 5-27. Print Page Antecedence, Vehicle Position Variable



Fuzzy Logic Toolkit : **Linguistic Antecedence Variable**

Date: 2/9/97

Controller Name : Bwdtruck.fc

Time: 1:18 AM

vehicle-orientation

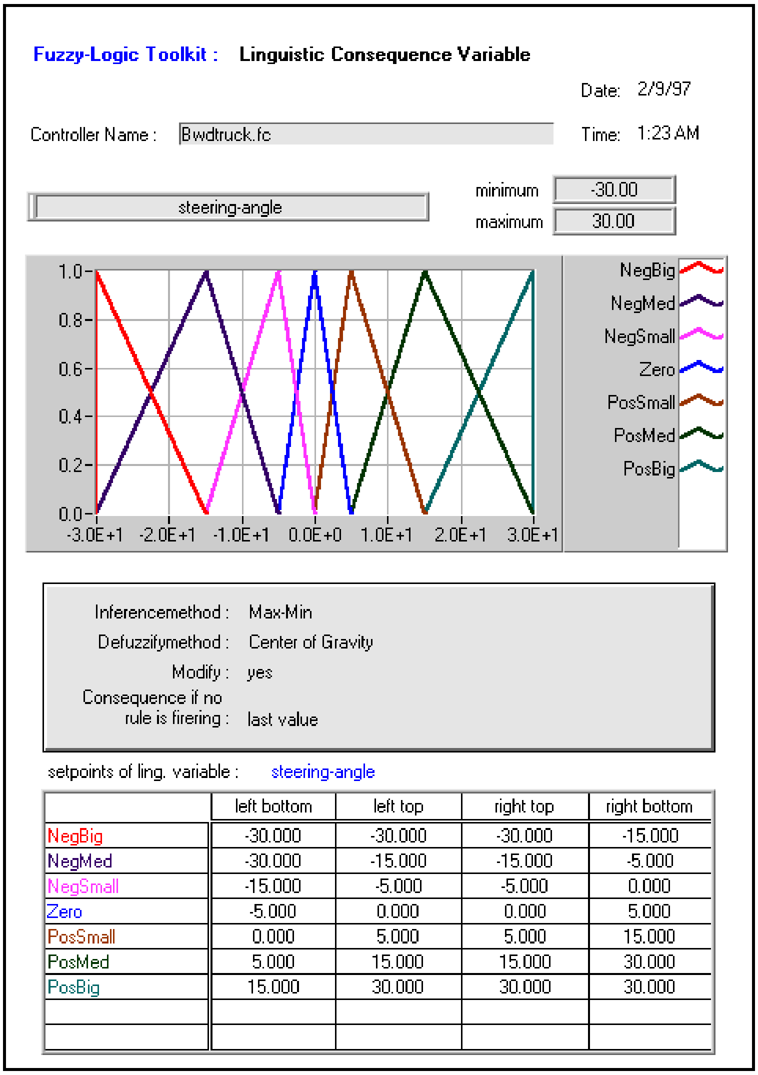
minimum -90.00

maximum 270.00

setpoints of ling. variable : vehicle-orientation

	left bottom	left top	right top	right bottom
left-down	-90.000	-90.000	-45.000	20.000
left	-20.000	20.000	20.000	70.000
left-up	45.000	70.000	70.000	90.000
up	75.000	90.000	90.000	105.000
right-up	90.000	110.000	110.000	135.000
right	110.000	160.000	160.000	200.000
right-down	160.000	225.000	270.000	270.000

Figure 5-28. Print Page Antecedence, Vehicle Orientation Variable



Fuzzy-Logic Toolkit : **Linguistic Consequence Variable**

Date: 2/9/97

Controller Name : Bwdtruck.fc

Time: 1:23 AM

steering-angle

minimum -30.00

maximum 30.00

Inferencemethod : Max-Min

Defuzzifymethod : Center of Gravity

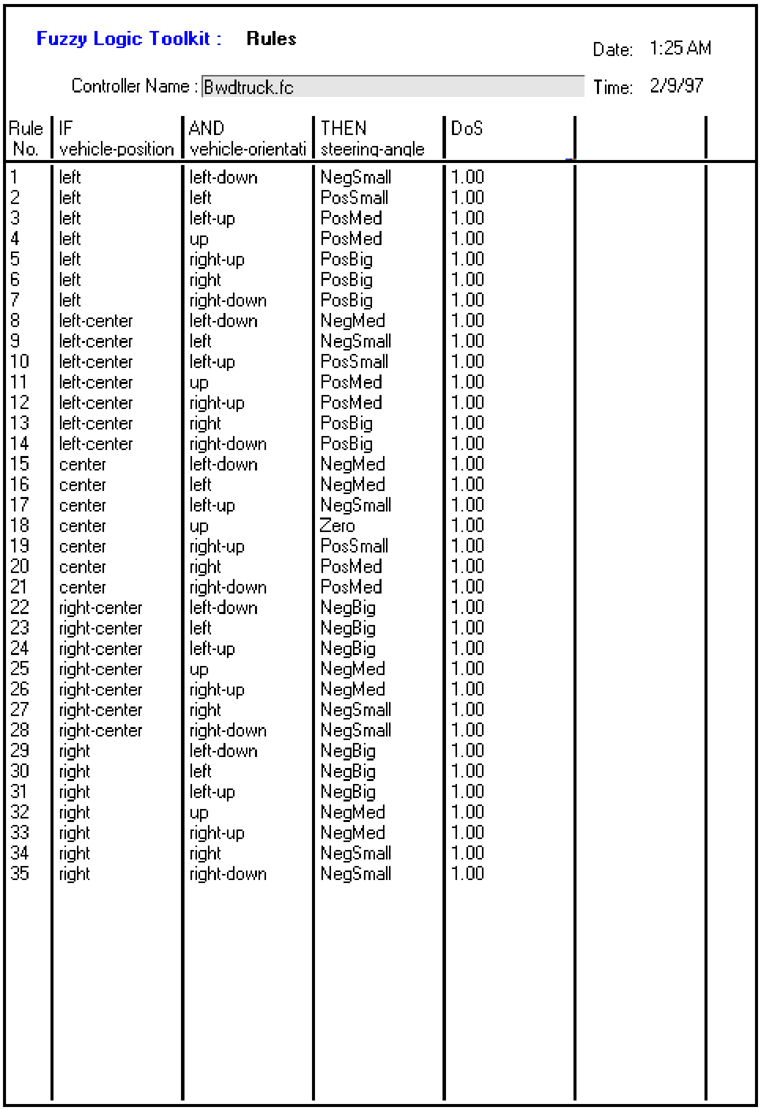
Modify : yes

Consequence if no rule is firering : last value

setpoints of ling. variable : steering-angle

	left bottom	left top	right top	right bottom
NegBig	-30.000	-30.000	-30.000	-15.000
NegMed	-30.000	-15.000	-15.000	-5.000
NegSmall	-15.000	-5.000	-5.000	0.000
Zero	-5.000	0.000	0.000	5.000
PosSmall	0.000	5.000	5.000	15.000
PosMed	5.000	15.000	15.000	30.000
PosBig	15.000	30.000	30.000	30.000

Figure 5-29. Print Page Consequence/Methods



Fuzzy Logic Toolkit : Rules

Date: 1:25 AM

Controller Name : Bwdtruck.fc

Time: 2/9/97

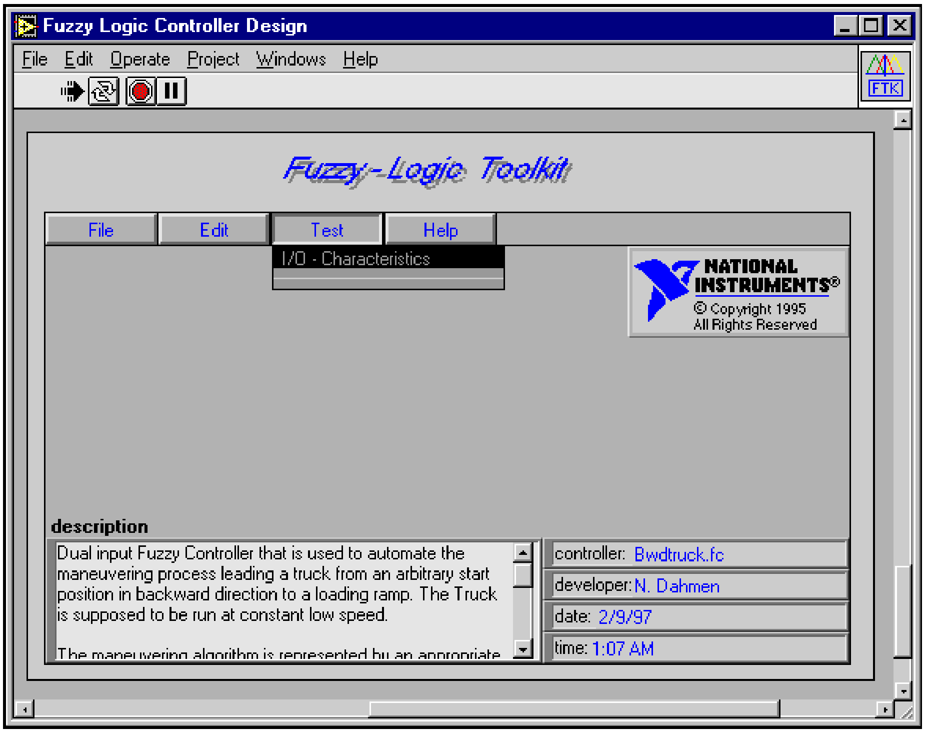
Rule No.	IF vehicle-position	AND vehicle-orientati	THEN steering-angle	DoS		
1	left	left-down	NegSmall	1.00		
2	left	left	PosSmall	1.00		
3	left	left-up	PosMed	1.00		
4	left	up	PosMed	1.00		
5	left	right-up	PosBig	1.00		
6	left	right	PosBig	1.00		
7	left	right-down	PosBig	1.00		
8	left-center	left-down	NegMed	1.00		
9	left-center	left	NegSmall	1.00		
10	left-center	left-up	PosSmall	1.00		
11	left-center	up	PosMed	1.00		
12	left-center	right-up	PosMed	1.00		
13	left-center	right	PosBig	1.00		
14	left-center	right-down	PosBig	1.00		
15	center	left-down	NegMed	1.00		
16	center	left	NegMed	1.00		
17	center	left-up	NegSmall	1.00		
18	center	up	Zero	1.00		
19	center	right-up	PosSmall	1.00		
20	center	right	PosMed	1.00		
21	center	right-down	PosMed	1.00		
22	right-center	left-down	NegBig	1.00		
23	right-center	left	NegBig	1.00		
24	right-center	left-up	NegBig	1.00		
25	right-center	up	NegMed	1.00		
26	right-center	right-up	NegMed	1.00		
27	right-center	right	NegSmall	1.00		
28	right-center	right-down	NegSmall	1.00		
29	right	left-down	NegBig	1.00		
30	right	left	NegBig	1.00		
31	right	left-up	NegBig	1.00		
32	right	up	NegMed	1.00		
33	right	right-up	NegMed	1.00		
34	right	right	NegSmall	1.00		
35	right	right-down	NegSmall	1.00		

Figure 5-30. Print Page Rules

Test Facilities

Before running a fuzzy controller within a designated system environment, it is useful to study its I/O characteristics within the toolkit. Optimization and necessary modifications can be carried out that way. An appropriate test environment is available within the Fuzzy Logic Toolkit.

You can call the test facility to perform the I/O characteristic studies of a fuzzy controller by selecting **Test»I/O-Characteristics,** as shown in Figure 5-31.



description

Dual input Fuzzy Controller that is used to automate the maneuvering process leading a truck from an arbitrary start position in backward direction to a loading ramp. The Truck is supposed to be run at constant low speed.

The maneuvering algorithm is represented by an appropriate

controller: Bwdtruck.fc
developer: N. Dahmen
date: 2/9/97
time: 1:07 AM

Figure 5-31. Selecting the I/O-Characteristics Command from the Test Menu

For the application example, `FuzzyTruck`, previously loaded into the Fuzzy Logic Toolkit, the I/O-Characteristic test facility starts with a front panel, similar to the one shown in Figure 5-32.

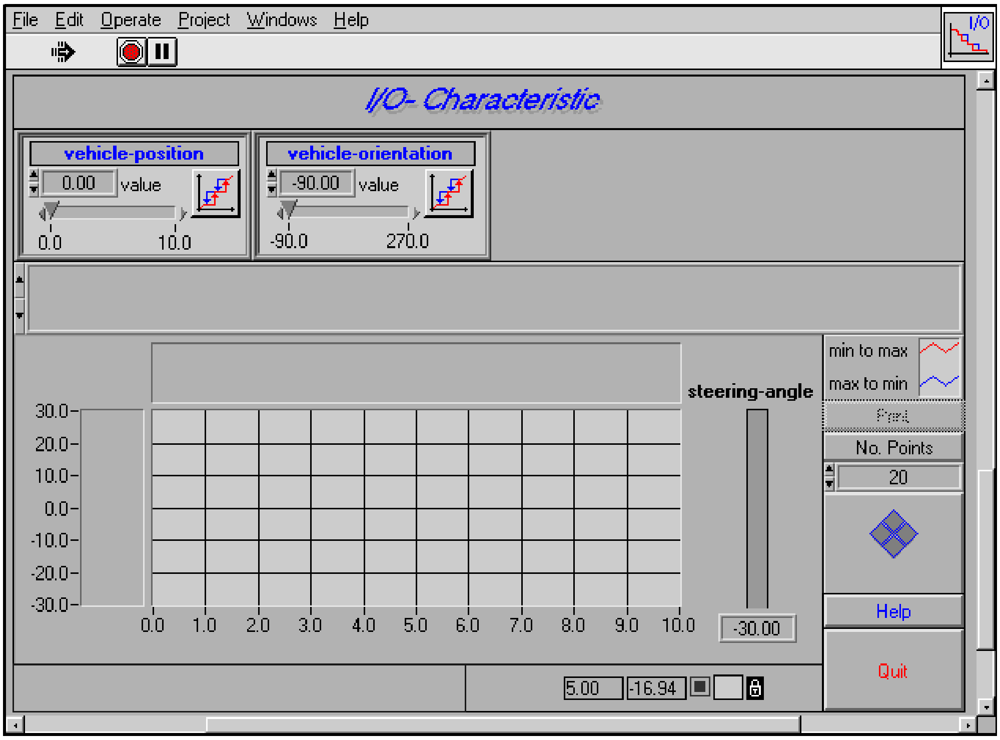


Figure 5-32. I/O-Characteristic Project-Specific Front Panel

Each input variable of the fuzzy controller is represented by a specific parameter control block within the Input Parameter Field of the I/O-Characteristic front panel. It is used to set up the desired test conditions for the different controller inputs.

Suppose you want to observe the behavior of the controller output variable, *steering-angle*, depending on the vehicle-position and the vehicle-orientation by varying the vehicle-position within the whole input data range and keeping the vehicle-orientation constant at 0°.

To set up these test conditions, first enter the desired test value into the parameter control block for vehicle-orientation, as shown in Figure 5-33.

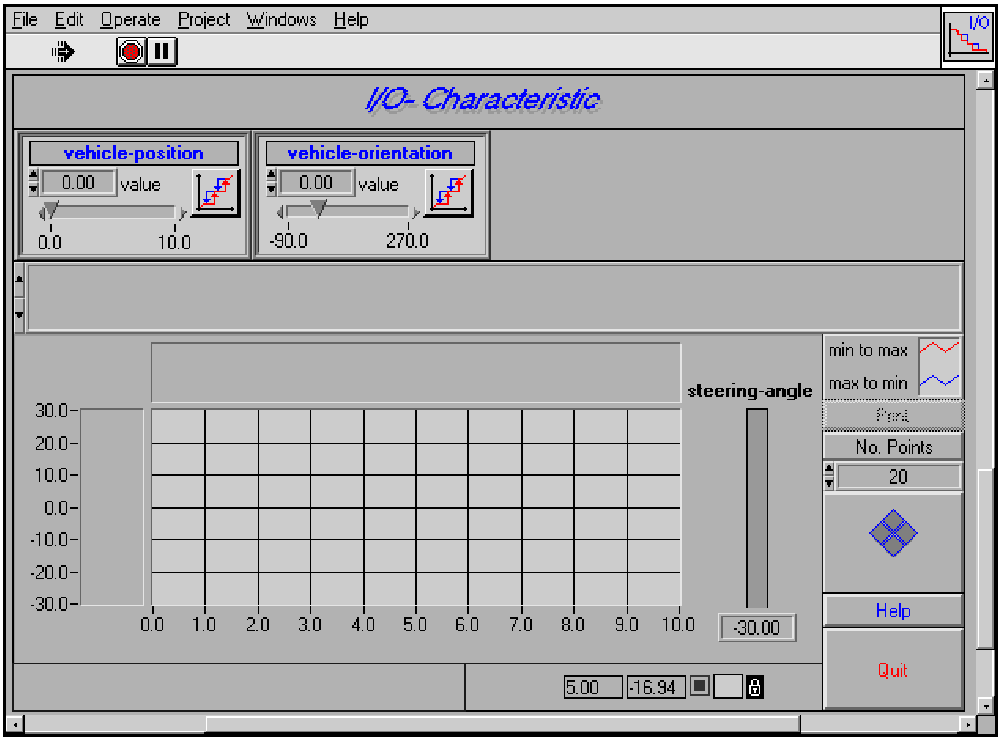
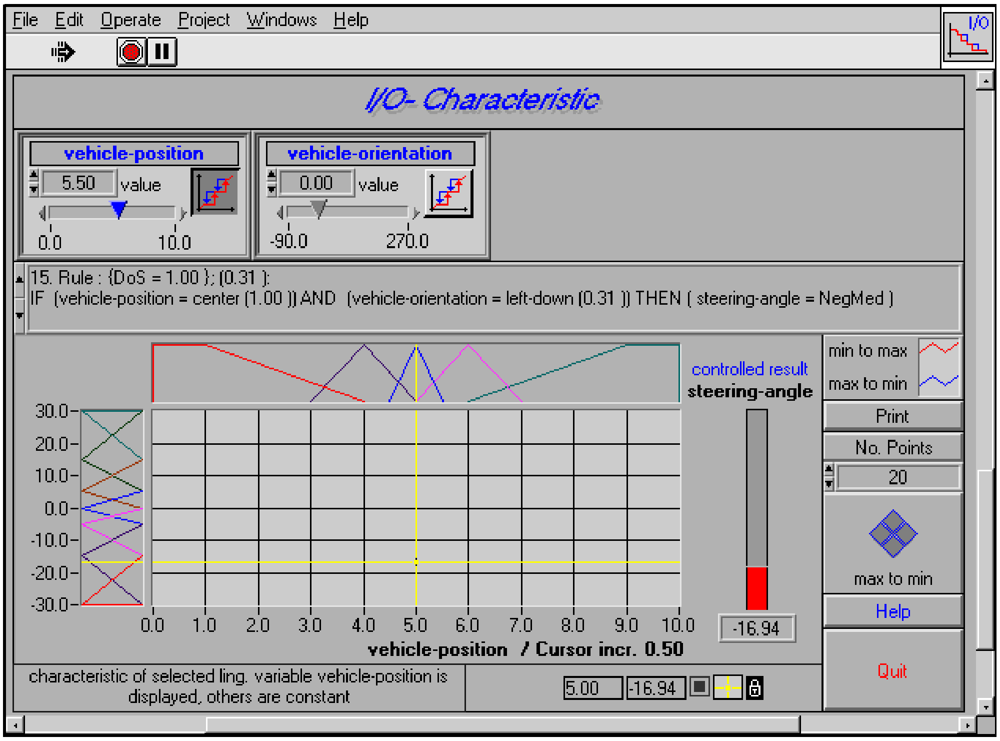


Figure 5-33. Entering a Test Condition into a Parameter Control Block of the I/O-Characteristic Front Panel

Then begin calculating the I/O characteristic by clicking the **Run** button within the parameter control block for vehicle-position, as shown in Figure 5-34.



15. Rule : {DoS = 1.00 }; (0.31):
IF (vehicle-position = center (1.00)) AND (vehicle-orientation = left-down (0.31)) THEN (steering-angle = NegMed)

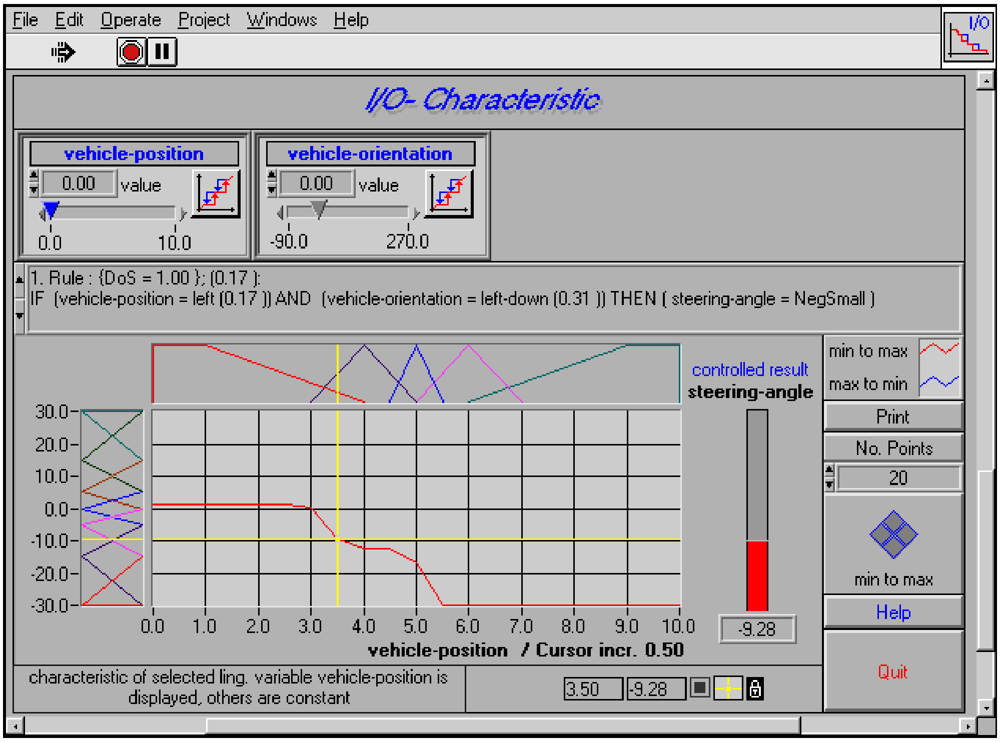
characteristic of selected ling. variable vehicle-position is displayed, others are constant

Figure 5-34. Activating a Test Calculation

The I/O characteristics calculation is carried out according to the number of points specified in the No. Points control box. The calculation process is animated by moving the slider of the varying input variable.

Note: ***The controller characteristic is calculated twice, varying the activated input variable (vehicle-position in the example) from the minimum value up to the maximum value, and vice versa. This happens because of possible hysteresis effects that occur with incomplete rule bases. It can also be caused by definition gaps in the term arrangement of the input variable, causing the controller to use the default output value or the last originally-computed value.***

As soon as the characteristic calculation is finished, the characteristic curve is drawn in the I/O-Characteristic display, as shown in Figure 5-35.



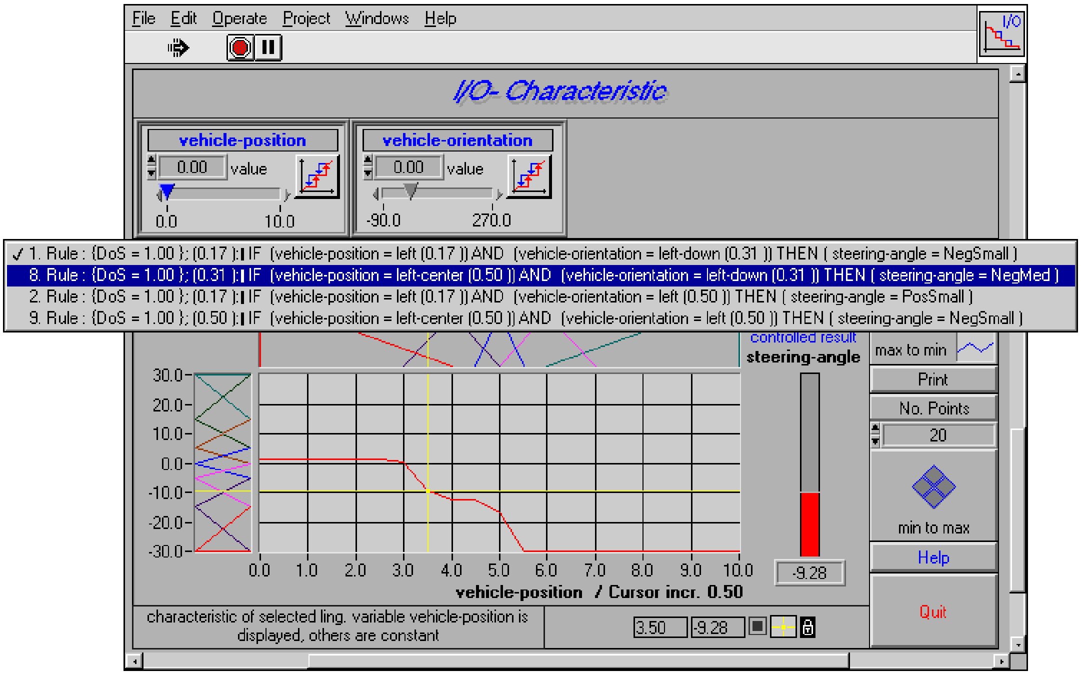
1. Rule : {DoS = 1.00 }; [0.17]:
IF (vehicle-position = left [0.17]) AND (vehicle-orientation = left-down [0.31]) THEN (steering-angle = NegSmall)

characteristic of selected ling. variable vehicle-position is displayed, others are constant

Figure 5-35. Controller Characteristic Displayed

The I/O-Characteristic display contains a cursor that you can control with the Cursor Navigation block, also shown in Figure 5-35. The cursor can travel along the characteristic curve and identify the active rules for the input situation at each cursor position. The current input values and the controller output value are displayed on the I/O-Characteristic function panel.

All active rules within the input situation determined by the cursor position are displayed in the Active Rules display, including the degree of truth for each antecedence term. Each active rule can be selected as shown in Figure 5-36.

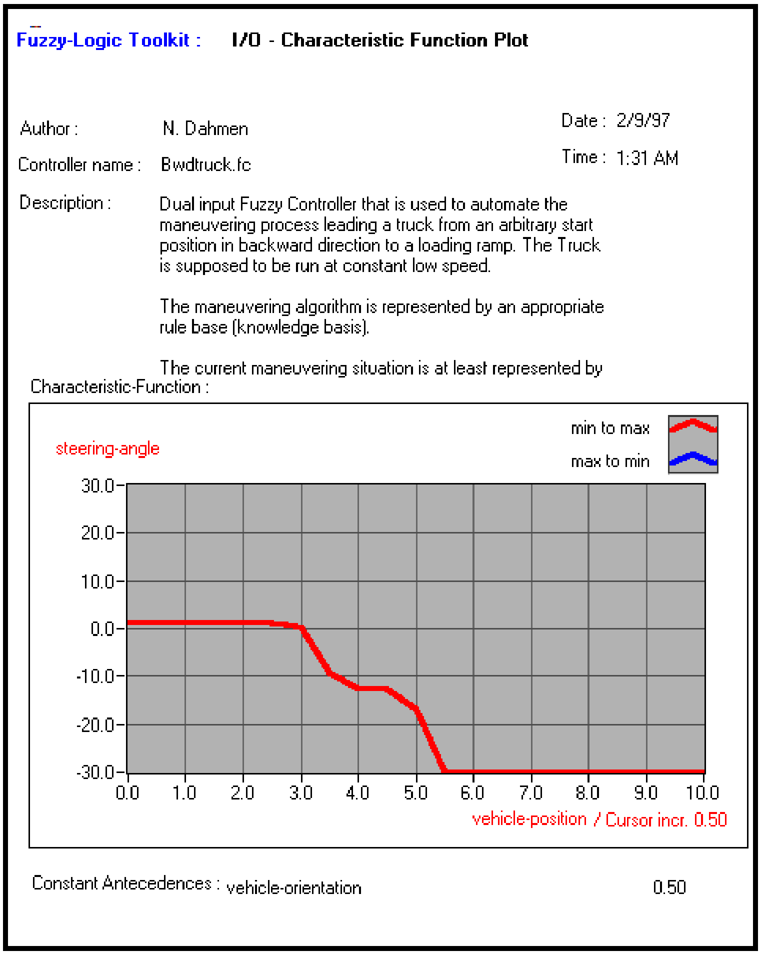


✓ 1. Rule : {DoS = 1.00 }; (0.17): IF (vehicle-position = left (0.17)) AND (vehicle-orientation = left-down (0.31)) THEN (steering-angle = NegSmall)
8. Rule : {DoS = 1.00 }; (0.31): IF (vehicle-position = left-center (0.50)) AND (vehicle-orientation = left-down (0.31)) THEN (steering-angle = NegMed)
2. Rule : {DoS = 1.00 }; (0.17): IF (vehicle-position = left (0.17)) AND (vehicle-orientation = left (0.50)) THEN (steering-angle = PosSmall)
9. Rule : {DoS = 1.00 }; (0.50): IF (vehicle-position = left-center (0.50)) AND (vehicle-orientation = left (0.50)) THEN (steering-angle = NegSmall)

characteristic of selected ling. variable vehicle-position is displayed, others are constant

Figure 5-36. Selecting One of the Active Rules from the Active Rules Display

The current situation can be printed out for documentation purposes by clicking the **Print** button above the cursor control block. An example printout is shown in Figure 5-37.



Fuzzy-Logic Toolkit : I/O - Characteristic Function Plot

Author : N. Dahmen

Date : 2/9/97

Controller name : Bwdtruck.fc

Time : 1:31 AM

Description : Dual input Fuzzy Controller that is used to automate the maneuvering process leading a truck from an arbitrary start position in backward direction to a loading ramp. The Truck is supposed to be run at constant low speed.

The maneuvering algorithm is represented by an appropriate rule base (knowledge basis).

The current maneuvering situation is at least represented by

Characteristic-Function :

Constant Antecedences : vehicle-orientation 0.50

Figure 5-37. Printing Results of a Characteristic Curve

Implementing a Fuzzy Controller

Chapter 6

This chapter describes how to implement a fuzzy controller and includes a pattern recognition application example. There are a few different ways to implement a fuzzy controller using the Fuzzy Logic Toolkit. The easiest way is to use the Fuzzy Controller VI, as demonstrated in the following example.

Pattern Recognition Application Example

Suppose you must develop and implement a fuzzy controller that identifies the shape of different-sized triangular, hexagonal, and rectangular plastic parts moving on a conveyor belt through a simple reflex light barrier, as sketched in Figure 6-1.

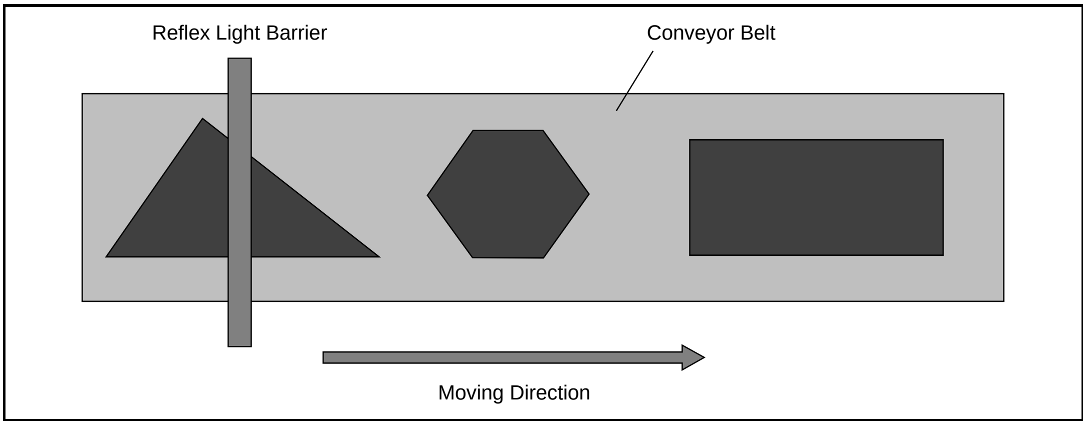


Figure 6-1. Sensor Facility

The plastic parts might be symmetric or asymmetric. The reflex light barrier reads a characteristic voltage signal for each plastic part. The signal depends on the resistances set up on the light barrier. Measuring these signals with a real sensor shows that even the signals of identical plastic parts vary to a certain extent. Different environmental conditions such as scattered light can affect the signal. Some typical voltage drop curves derived from an asymmetric triangle (lefthand-shaped triangle) are shown in Figure 6-2.

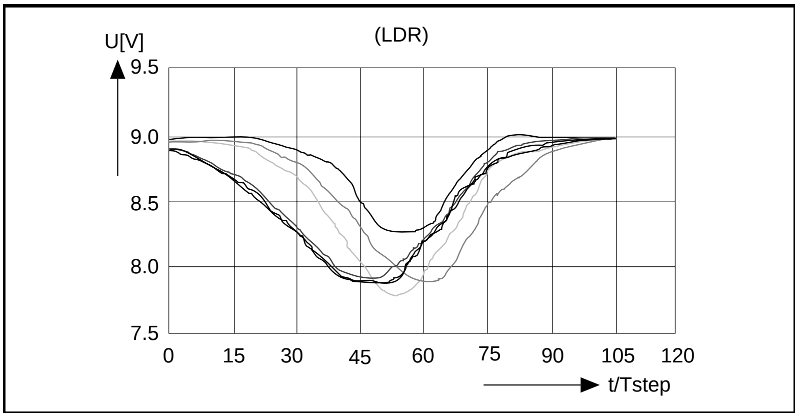


Figure 6-2. Typical Voltage Drop Curves Obtained from a Lefthand-Shaped Triangle

To obtain a simple but efficient controller, abstract the curves shown in Figure 6-2 into the idealized curve outline that is shown in Figure 6-3.

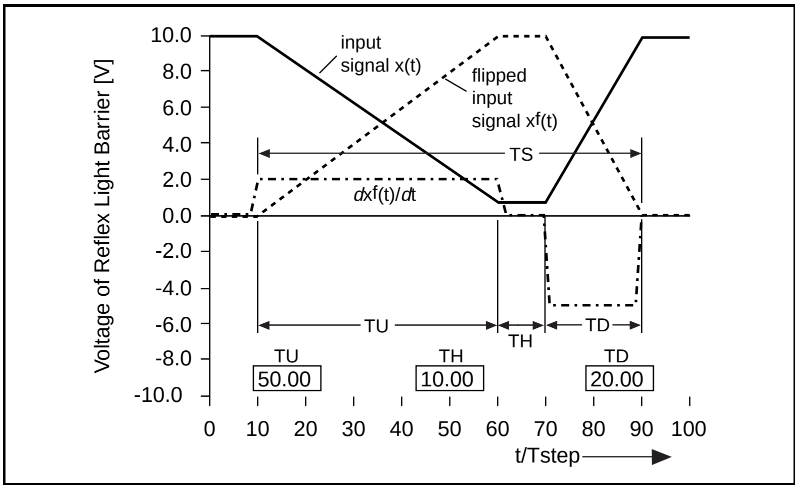


Figure 6-3. Abstract Voltage Drop Curve for Feature Extraction

There are three distinguishable parts of the flipped input signal represented by the dashed curve $x^f(t)$ in Figure 6-3. There is a rising curve part, a constant part, and a falling curve part. Differentiation of the flipped input signal yields the dash-dotted curve, $dx^f(t)/dt$, from which you can derive the time intervals TU (up), TH (hold) and

TD (down). With TS (signal) representing complete operation time, you can extract the following features for the desired pattern recognition:

TH / TS ≈ 0	==> Triangle	(TU–TD) / TS > 0 ==> lefthand-shaped
0 < TH / TS > 1	==> Hexagon	(TU–TD) / TS ≈ 0 ==> symmetrical
TH / TS ≈ 1	==> Rectangle	(TU–TD) / TS < 0 ==> righthand-shaped

☞ **Note:** ***All the signal processing steps described above, can be performed by existing functions or by functions you can write in G.***

Because the real sensor signal is not an idealized signal as shown above, the characteristic features derived from it are not precise. You can model them directly by the appropriate linguistic terms for the two linguistic input variables TH/TS and (TU–TD)/TS. Using the toolkit as described in Chapter 5, *Using the Fuzzy Logic Controller Design VI*, the term arrangements shown in Figures 6-4 and 6-5 exist for the input variables TH/TS and (TU–TD)/TS.

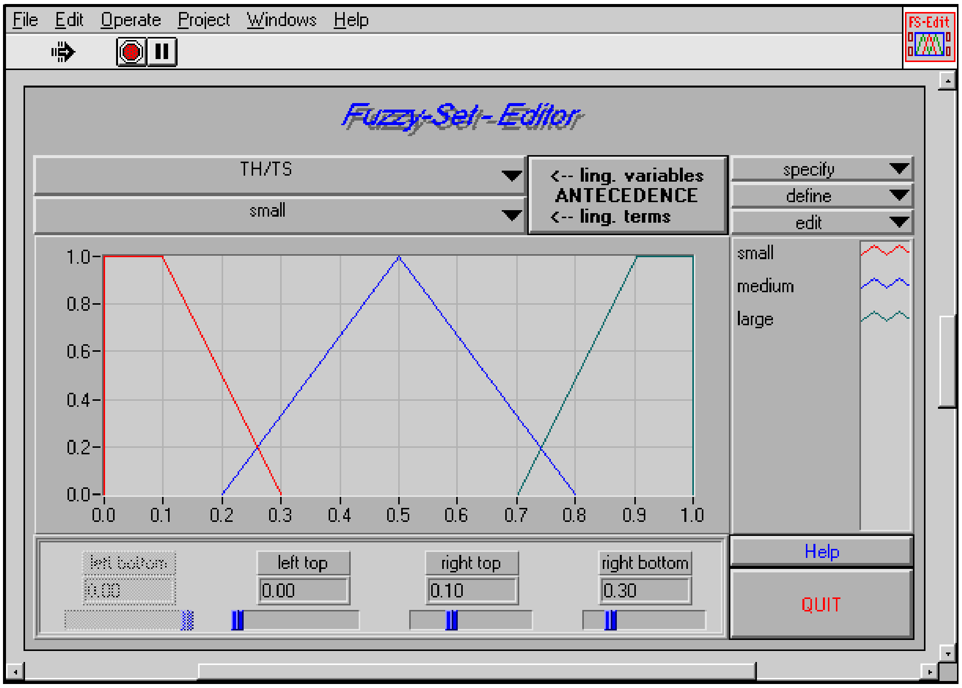


Figure 6-4. Linguistic Term Arrangement of Input Variable TH/TS

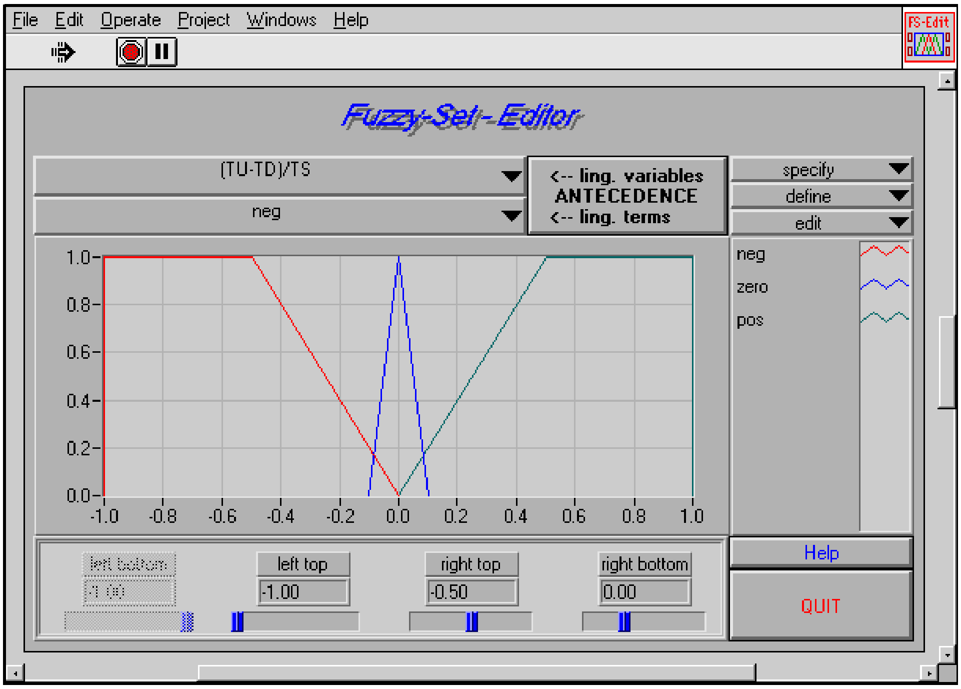


Figure 6-5. Linguistic Term Arrangement of Input Variable (TU–TD)/TS

The linguistic output variable *object* can be composed of singletons, each of which represents a specific shape. The term arrangement is shown in Figure 6-6, and the rule base is shown in Figure 6-7.

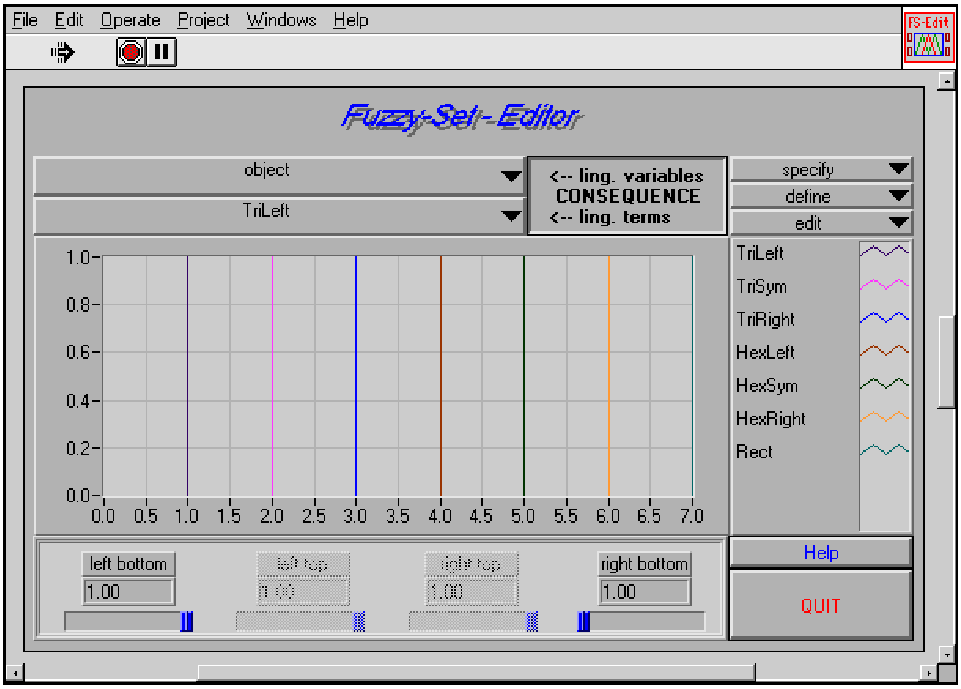
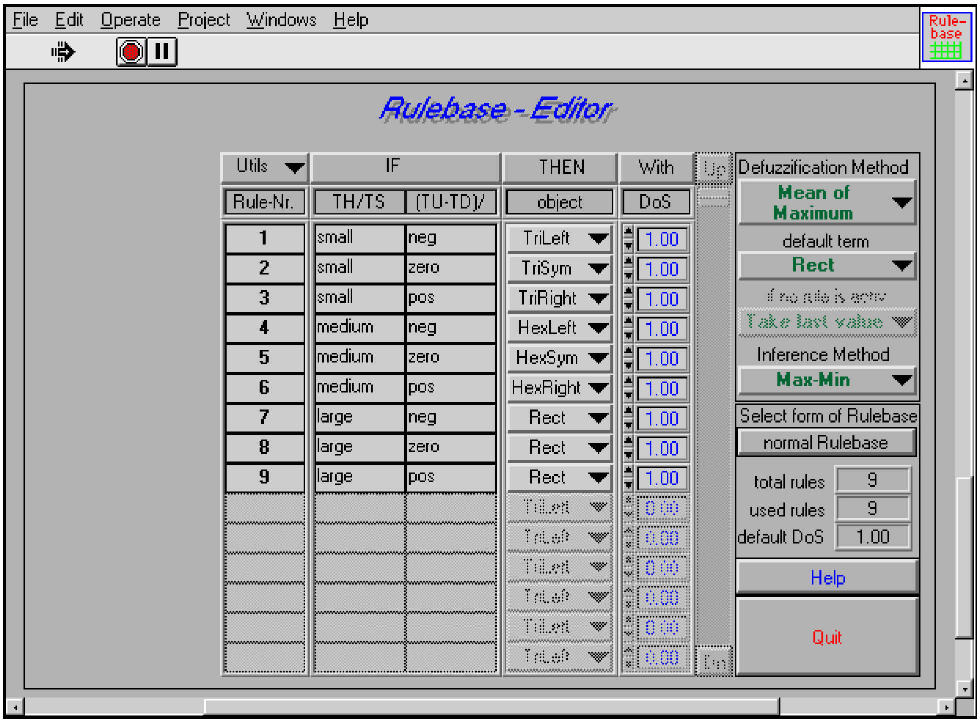


Figure 6-6. Linguistic Term Arrangement of the Output Variable, Object



Rule-Nr.	IF TH/TS	IF (TU-TD)/	THEN object	With DoS
1	small	neg	TriLeft	1.00
2	small	zero	TriSym	1.00
3	small	pos	TriRight	1.00
4	medium	neg	HexLeft	1.00
5	medium	zero	HexSym	1.00
6	medium	pos	HexRight	1.00
7	large	neg	Rect	1.00
8	large	zero	Rect	1.00
9	large	pos	Rect	1.00
			TriLeft	0.00
			TriLeft	0.00
			TriLeft	0.00
			TriLeft	0.00
			TriLeft	0.00
			TriLeft	0.00

Defuzzification Method: Mean of Maximum
default term: Rect
if no rule is activ: Take last value
Inference Method: Max-Min
Select form of Rulebase: normal Rulebase
total rules: 9
used rules: 9
default DoS: 1.00

Figure 6-7. Complete Rule Base Describing the Pattern Recognition Process

The principal program structure of the pattern recognition facility is simply a loop structure, which repeatedly takes the input signal from a data acquisition board using the easy I/O VIs, for example, and processing it according to the conditions described above. To experiment with the fuzzy controller independently from specific data acquisition equipment, consider the following simulation environment.

The SignalGen VI on the left side of the block diagram shown in Figure 6-8 corresponds to the input side of a process controller. The NumtoString VI on the right side of the diagram can be regarded as the output side of a process controller. It supplies all necessary output signals, including the signals used for process animation.

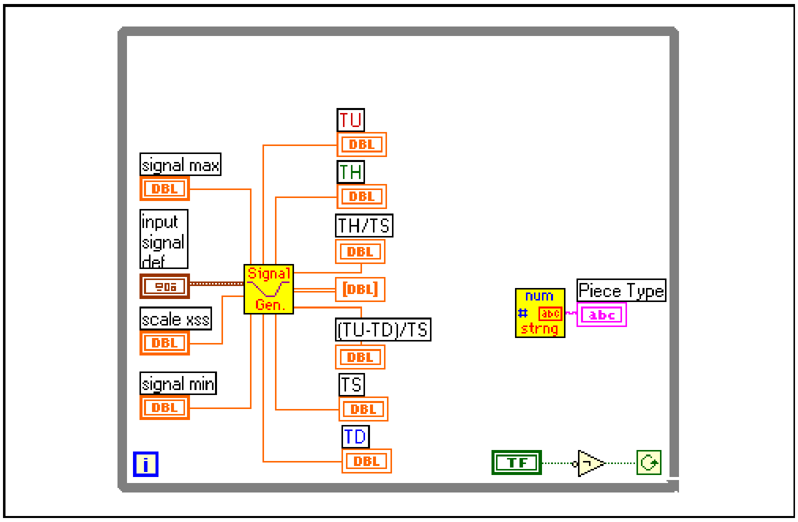


Figure 6-8. Block Diagram of the Pattern Recognition Application Prepared for Entering the Pre-Defined Fuzzy Controller VI

The data acquisition part, including all the data pre-processing activities, is replaced by the SignalGen VI, which directly supplies the necessary input signals, TH/TS and (TU – TD)/TS, for the example application. All other input and output signals used in the block diagram are part of the user interface that includes all the controls and indicators you can use to adjust the pattern recognition application example. The front panel is shown in Figure 6-9.

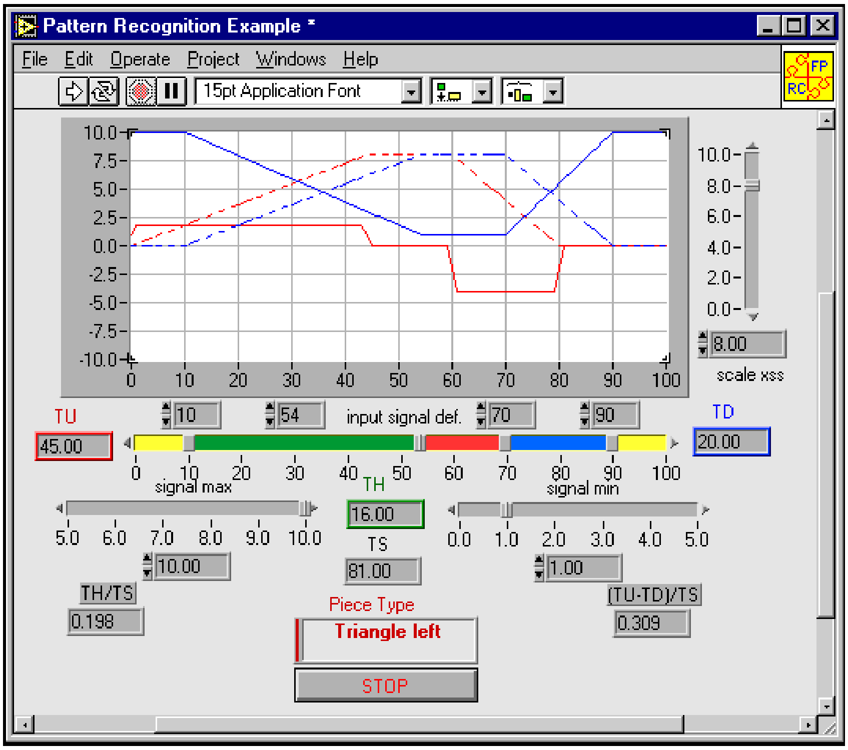


Figure 6-9. Front Panel of the Pattern Recognition Application

You can use the **input signal def.** sliders to simulate the signal from the reflex light barrier of the real system. The signal conditions also can be modified by the **signal max** and **signal min** sliders to test how the fuzzy controller works despite having a signal with a very small amplitude The **scale xss** slider serves to model a kind of gain factor towards the signal that is performed by the data pre-processing step. It also can be used to study how different signal conditions can affect the result of the pattern recognition process.

Fuzzy Controller Implementation

Fuzzy Controller VI

Now incorporate the fuzzy controller into the application block diagram. You do not need to program the fuzzy controller, just use the pre-defined Fuzzy Controller VI available with the Fuzzy Logic Toolkit package, shown in Figure 6-10.

The pre-defined Fuzzy Controller VI can be connected with as many as four input signals from a process and one output signal used as a control value. Although the Fuzzy Controller VI has many different inputs and outputs, at this time you only need those inputs and outputs shown in bold in Figure 6-10.

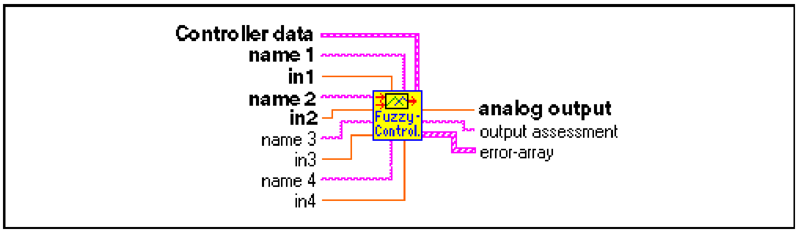


Figure 6-10. Fuzzy Controller VI

The input signals TH/TS and (TU–TD)/TS can be connected to the Fuzzy Controller VI inputs **input1** and **input2**. The output signal of the Fuzzy Controller VI, called **analog output,** also can be connected to the input side of the NumtoString VI. There are a few inputs of the Fuzzy Controller VI being left unconnected at this time.

Loading Fuzzy Controller Data

Load Fuzzy Controller VI

The Fuzzy Controller VI can be compared to a microprocessor that does not have an executable program loaded. To obtain the specific data for the fuzzy controller, you must use the Load Fuzzy Controller VI to load the required data into the Fuzzy Controller VI. This VI also is included in the Fuzzy Logic Toolkit package.

Because the controller data must be loaded into the Fuzzy Controller VI when the pattern recognition application is started, place it outside the While-Loop, as shown in Figure 6-12. Although the Load Fuzzy

Controller VI has many outputs, at this time you only need those outputs shown in bold in Figure 6-11.

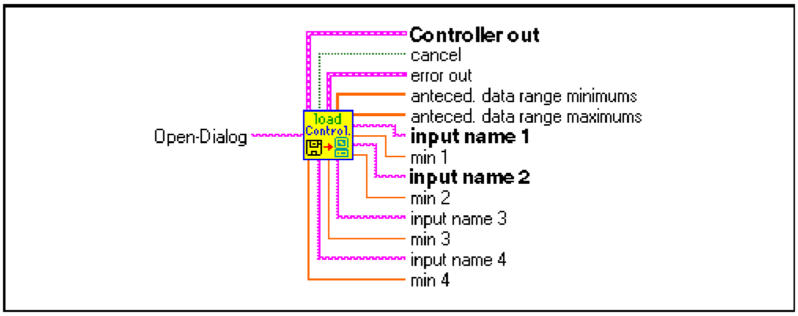


Figure 6-11. Load Fuzzy Controller VI

The result of all the necessary wiring work is shown in Figure 6-12.

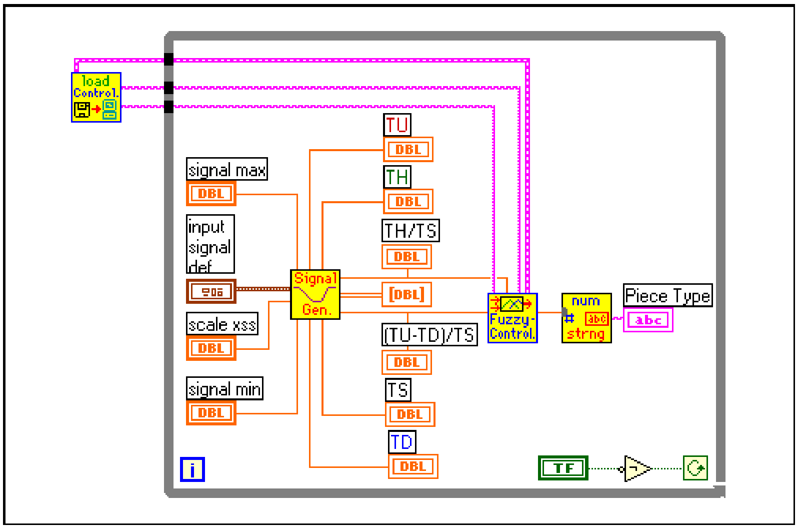
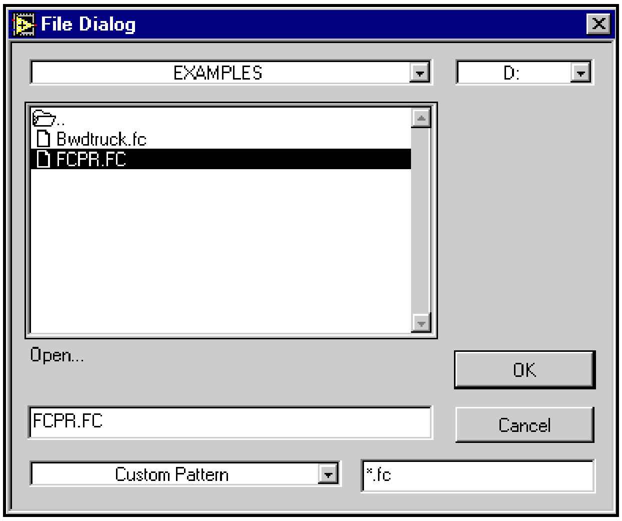


Figure 6-12. Block Diagram of the Pattern Recognition Application

The application example is complete. You can start the pattern recognition application using your fuzzy controller by switching back to the front panel and running the VI.

Immediately after the application begins, a file dialog box prompts you to enter a file containing the appropriate controller data (see Figure 6-13). Open the project file `FCPR.fc`, which represents the fuzzy controller you designed previously.



File Dialog

EXAMPLES

D:

..
Bwdtruck.fc
FCPR.FC

Open...

FCPR.FC

Custom Pattern

*.fc

Figure 6-13. Loading the Fuzzy Controller Data

When the Fuzzy Controller data is loaded, you can try different settings for the pattern recognition process by dragging the sliders. You can see how the pattern recognition process changes with different input signal conditions (see Figure 6-14).

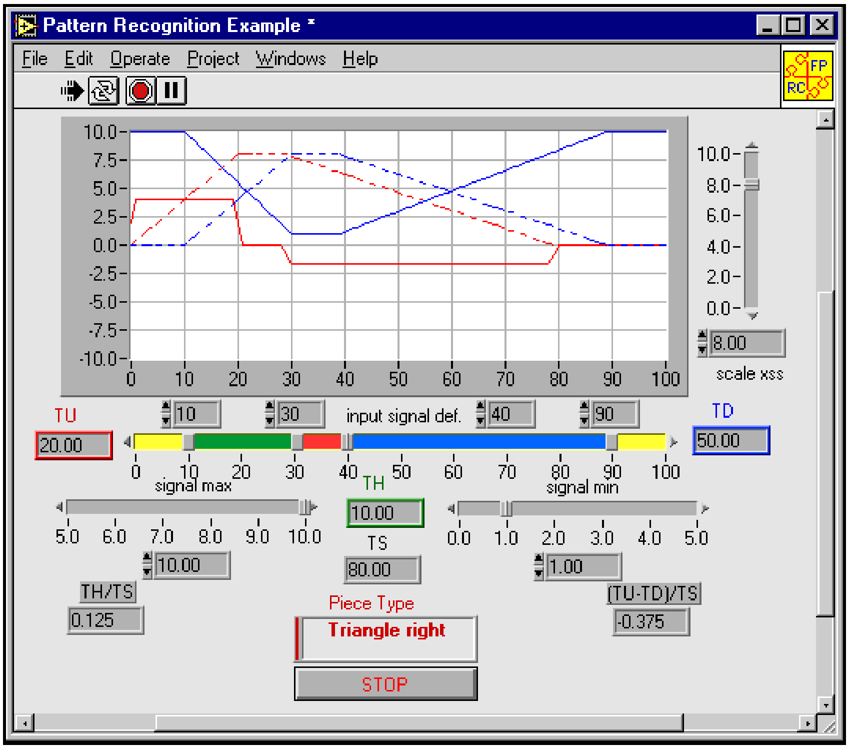


Figure 6-14. Running the Pattern Recognition Application

Look back at Figure 6-13, which shows the file dialog box for loading the fuzzy controller data. Pressing the **Cancel** button instead of selecting the fuzzy controller data file, `FCPR.fc`, executes the default fuzzy controller repeatedly. Without having actual data loaded to the controller, it will use the default data. See the block diagram of the complete pattern recognition application shown in Figure 6-12.

Because of security aspects that may occur when running a controller within a real application environment, the controller should not start if the **Cancel** button is pressed. To improve your controller design, place the While Loop into a Case Structure and connect the **selection terminal** with the **cancel** output of the Load Fuzzy Controller VI (see

Figure 6-11). The result is shown in Figure 6-15. The TRUE case is empty, and the application quits if the **Cancel** button is pressed.

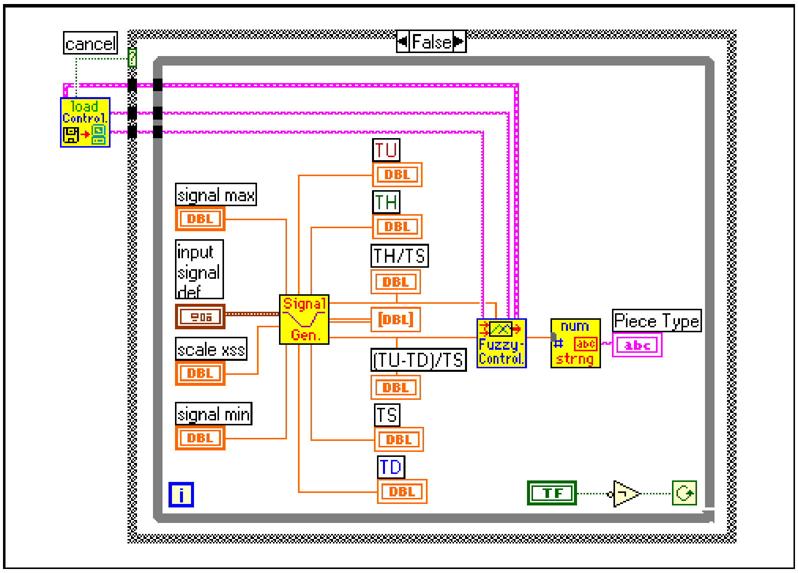


Figure 6-15. Improved Controller Application Block Diagram

The complete pattern recognition application example also is available within the Fuzzy Logic Toolkit package.

Saving Controller Data with the Fuzzy Controller

You might want to use a fuzzy controller like a predefined VI without the need to load its data first. You might wonder how the currently valid controller data file can be made the default for the controller so you can use it as a standalone controller.

A standalone Fuzzy Controller VI can be built for the pattern recognition application example by performing the following steps.

1. Bring the application block diagram to the front and open the Fuzzy Controller VI by double clicking its icon.
2. Bring the application front panel to the front.
3. Start the application so that the input file dialog box requesting a fuzzy controller data file opens.

4. Select the desired fuzzy controller data file, as shown in Figure 6-13.
5. Stop the application.
6. Bring the front panel of the Fuzzy Controller VI to the front.
7. Choose **Operate»make current values default** to make the currently valid controller data the default.
8. Use one of the following options:
   - *Save a copy* of the Fuzzy Controller VI if you want it to be available under a unique name. Select **No** when asked to save the original Fuzzy Controller VI to leave it unchanged.
   - *Save* the original Fuzzy Controller VI, which now has the current controller data as default values. Only the default values of the original Fuzzy Controller VI have been changed. The VI still can be used as a general-purpose Fuzzy Controller VI because the default values are used only when the controller is applied without loading specific data into the VI.
9. Close the application.

Now you can use either the new VI or the modified one as a standalone fuzzy controller as shown in Figure 6-16.

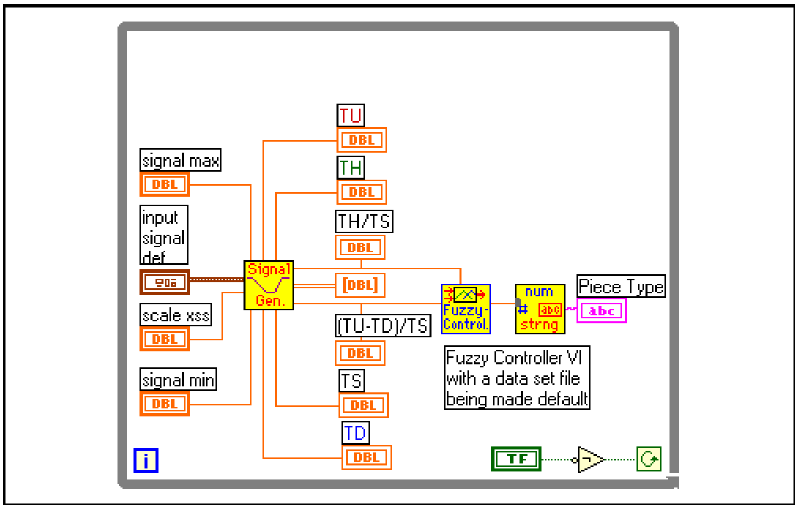


Figure 6-16. Application Block Diagram with Standalone Fuzzy Controller VI

Testing the Fuzzy Controller

There is another predefined VI available in the Fuzzy Logic Toolkit package that you can use to build or test fuzzy control applications. The Test Fuzzy Control VI supplies a fuzzy control test and application environment for as many as four different controller inputs. Input assignment is set automatically according to the data being loaded into the controller. This VI was created to show the proper use of all input and output signals supplied by the Load Fuzzy Controller VI and the Fuzzy Controller VI. The Test Fuzzy Control VI front panel is shown in Figure 6-17.

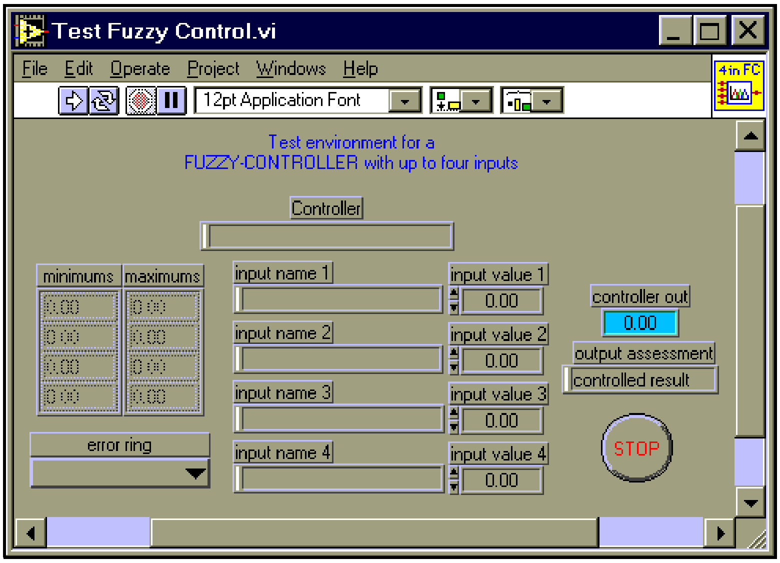
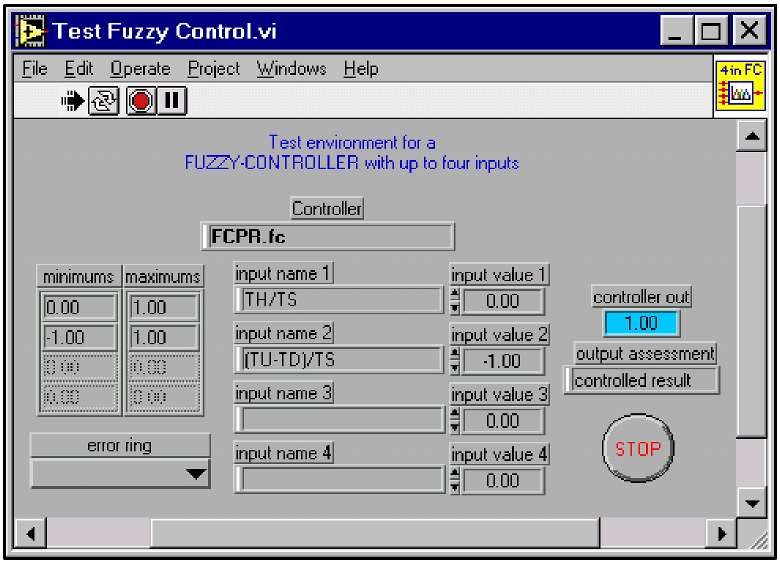


Figure 6-17. Test Fuzzy Control VI Front Panel

The fuzzy controller project identifier is displayed in the **controller** string indicator as soon as the fuzzy controller data file is loaded. The identifiers of all used inputs are displayed in the string **input name** indicator. The currently valid data range for each used input variable is displayed in appropriate **minimums** and **maximums** indicators. You can enter input values to stimulate the controller by using the **input value** control. The output value is displayed in the **controller out** indicator. Each input value is initialized automatically by its lower data range value.

Figure 6-18 shows the application front panel immediately after loading the fuzzy controller data file for the pattern recognition example.



Test environment for a
FUZZY-CONTROLLER with up to four inputs

Controller: FCPR.fc

minimums	maximums
0.00	1.00
-1.00	1.00
0.00	0.00
0.00	0.00

error ring

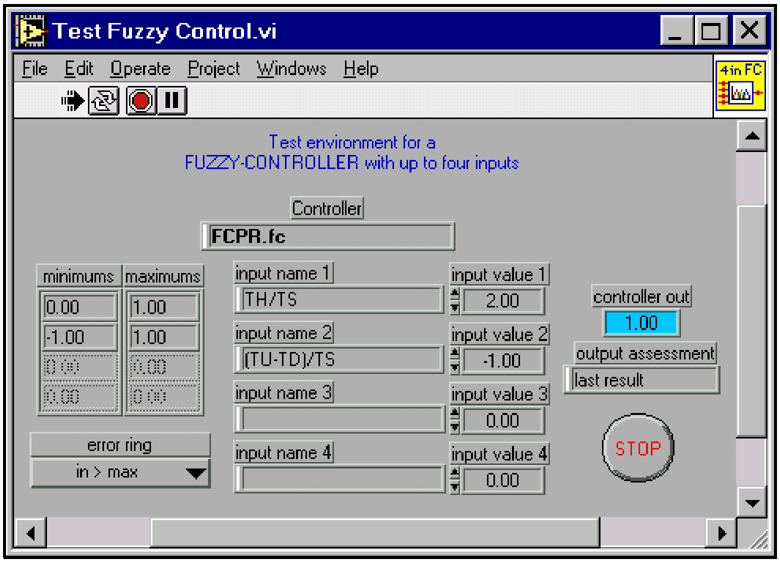
input name 1: TH/TS — input value 1: 0.00
input name 2: (TU-TD)/TS — input value 2: -1.00
input name 3: — input value 3: 0.00
input name 4: — input value 4: 0.00

controller out: 1.00
output assessment: controlled result

Figure 6-18. Test Fuzzy Control VI Front Panel with Controller Data Loaded

Remember that a fuzzy controller uses default values if there is an input situation not covered by active rules. This situation is indicated by a message displayed in the **output assessment** string indicator, as shown in Figure 6-19.

If input values exceed the data range assigned to the related input variable, an error message is displayed in the error ring, and the output value is set to the default output value, also shown in Figure 6-19.



Test environment for a
FUZZY-CONTROLLER with up to four inputs

Controller: FCPR.fc

minimums	maximums
0.00	1.00
-1.00	1.00
0.00	0.00
0.00	0.00

error ring: in > max

input name		input value	
input name 1	TH/TS	input value 1	2.00
input name 2	(TU-TD)/TS	input value 2	-1.00
input name 3		input value 3	0.00
input name 4		input value 4	0.00

controller out: 1.00

output assessment: last result

Figure 6-19. Test Fuzzy Control VI Front Panel with Incorrect Input Value for Input 1

The proper use of all input and output signals supplied by the Load Fuzzy Controller VI and the Fuzzy Controller VI is shown in Figure 6-20. You can use this program structure as a basis for building your own applications using fuzzy logic.

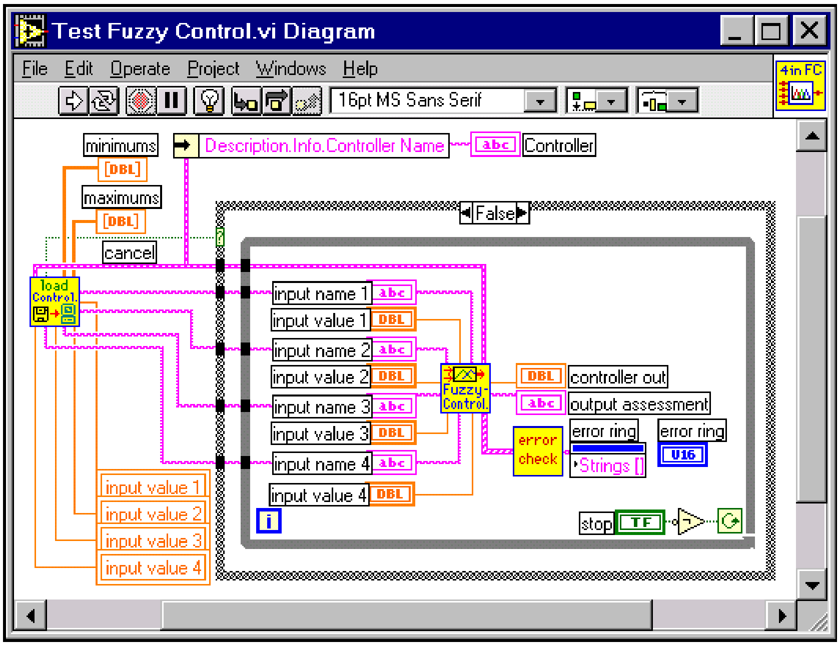


Figure 6-20. Test Fuzzy Control VI Block Diagram Example

Note: ***The inputs and the controller output can be connected directly to the outputs and inputs of the DAQ VIs available in LabVIEW or BridgeVIEW in order to use real process data from sensors instead of the values from the panel controls as shown in Figures 6-18 and 6-19.***

Chapter 7

Fuzzy Logic VI Descriptions

This chapter contains descriptions of the fuzzy logic VIs.

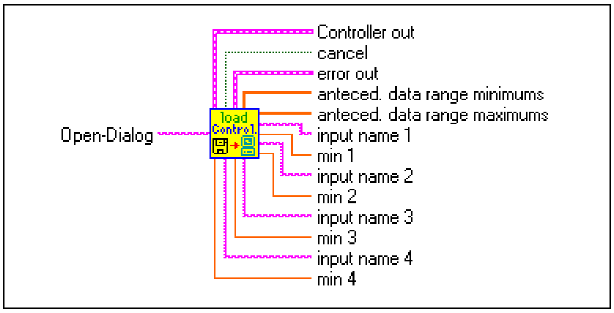
Fuzzy Logic Controller Design VI

This VI provides a graphical user interface for the definition of fuzzy controller membership functions, rule base, and controller parameters.

This VI is run-only and has no inputs or outputs. Typically the VI is run from the **File** menu in LabVIEW or BridgeVIEW by selecting **File»Open**. The VI is set up to run when opened and to close afterwards, therefore operating as a standalone application for the development of the fuzzy logic controller. The VI icon may be placed on the block diagram of your G application if you would like to launch the Fuzzy Logic Controller Design VI programmatically to edit the fuzzy controller.

Load Fuzzy Controller

This VI loads the complete set of fuzzy controller parameters and information defined in the Fuzzy Logic Controller Design VI. The file extension used for the data file is `.fc`.



Open-Dialog is the prompt used by the File Open VI when locating the fuzzy controller data file. The default prompt string is **Open...**.

Controller out is the cluster of all data used to define the fuzzy controller that is read in from the controller data file. The Load Fuzzy Controller VI reads all data from the `.fc` file, parses the data, and creates this cluster to be used by the Fuzzy Controller VI.

cancel is TRUE if you close the dialog box of the File Open VI by pressing the **Cancel** button or if an error occurs during the execution of the dialog box.

error out is a cluster that describes the error status after the Load Fuzzy Controller VI executes. If an error occurred before this VI was called, **error out** is the same as **error in**. Otherwise, **error out** displays the errors, if any, that occurred in this VI. Use the error handler VIs to identify the error codes and display the corresponding error messages. Using **error in** and **error out** clusters is a convenient way to check errors and to specify execution order by wiring the error output from one subVI to the error input of the next.

antecedent data range minimums is a one-dimensional array of the minimum values of the universe of discourse for each of the controller input variables. These values are defined in the Fuzzy Logic Controller Design VI for the inputs.

antecedent data range maximums is a one-dimensional array of the maximum values of the universe of discourse for each of the controller input variables. These values are defined in the Fuzzy Logic Controller Design VI for the inputs.

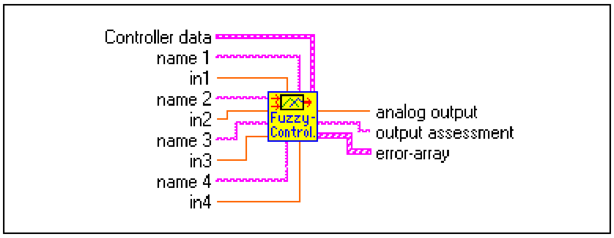
input name 1 through **input name 4** are the defined names of the corresponding controller input as defined in the Fuzzy Logic Controller Design VI. These names can be wired directly to the inputs of the Fuzzy Controller VI.

min 1 through **min 4** are the minimum values of the universe of discourse for the corresponding controller input variable.

Fuzzy Controller VI

The Fuzzy Controller VI is used to implement fuzzy control in your application. A controller data file (the project file generated by the Fuzzy Logic Controller Design VI) must be loaded by the Load Fuzzy Controller VI.

The Controller inputs can be connected to appropriate process variables using the Data Acquisition VIs. The Fuzzy Controller VI allows up to four inputs and one output. Each used name-input must have the name of the assigned input variable. The inputs of unwired name-input are disabled.



Controller data is the cluster of all data used to define the fuzzy controller that is read in from the controller data file. The Load Fuzzy Controller VI reads all data from the `.fc` file, parses the data, and creates this cluster to be used by the Fuzzy Controller VI.

name 1 through **name 4** are the names of a defined controller input for the fuzzy controller loaded by the Load Fuzzy Controller VI.

Note: ***The names must exactly match the names defined for the controller, but the specific inputs 1–4 may be in a different order than the output of the Load Fuzzy Controller VI.***

in 1 through **in 4** are the controller input values (process parameters to be controlled). The input number corresponds to the input number of the **name 1** through **name 4** inputs.

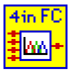
analog output is the controller output value (process control value) determined by the fuzzy logic controller.

output assessment indicates whether or not at least one rule was activated for the given set of input values. During normal fuzzy processing when input values cause at least one rule to fire, the string **controlled value** is returned. In the event that no rules are fired (all inputs have degree of membership 0 to all linguistic fuzzy sets), either **default value** or **last value** is returned, depending on the option specified in the Fuzzy Logic Controller Design VI for the given fuzzy controller.

error array is a cluster that describes the error status after this VI executes. **error array** displays the errors, if any, that occurred in this VI.

Test Fuzzy Control VI

This VI was created as an example for testing a fuzzy controller application.



This VI can be used as a general-purpose Fuzzy Controller VI with as many as four inputs and one output that can be connected to the desired process directly. The fuzzy controller data file containing the fuzzy project data is requested to be loaded from disk after starting up the VI. Input values can be entered using the input controls of the VI front panel when used as a test environment. The controller output is also displayed on the VI front panel.

☞ **Note:** ***To use this VI as a subVI in your control application, you must define the connector terminals. The VI has no defined inputs or outputs, so you must edit the terminals and save the VI with the redefined connector.***

Appendix A

References

This appendix lists the reference material used to produce the VIs in this manual. These references contain more information on the theory and algorithms implemented in the fuzzy logic VIs.

Industrial Applications of Fuzzy Logic and Intelligent System. Edited by John Yen, Reza Langari, and Lotfi Zadeh. Piscataway, NJ: IEEE Press, 1995.

Zimmerman, H.-J. *Fuzzy Set Theory and Its Applications,* Second Revised Edition. Boston, MA: Kluwer Academic Publishers, 1991.

Kahlert, J. and Frank, H. *Fuzzy Control fuer Ingeniere.* Braunschweig, Wiesbaden: Vieweg, 1995.

Kahlert, J. and Frank, H. *Fuzzy Logik und Fuzzy Control.* Braunschweig, Wiesbaden: Vieweg, 1993.

Zimmerman, H.-J. *Fuzzy Sets, Decision Making, and Expert Systems.* Boston, Dordrecht, London: Kluwer Academic Publishers, 1987.

Customer Communication

Appendix B

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve your technical problems and a form you can use to comment on the product documentation. When you contact us, we need the information on the Technical Support Form and the configuration form, if your manual contains one, about your system configuration to answer your questions as quickly as possible.

National Instruments has technical assistance through electronic, fax, and telephone systems to quickly provide the information you need. Our electronic services include a bulletin board service, an FTP site, a fax-on-demand system, and e-mail support. If you have a hardware or software problem, first try the electronic support systems. If the information available on these systems does not answer your questions, we offer fax and telephone support through our technical support centers, which are staffed by applications engineers.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: (512) 794-5422
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 01 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as `anonymous` and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at (512) 418-1111.

E-Mail Support (currently U.S. only)

You can submit technical support questions to the applications engineering team through e-mail at the Internet address listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

`support@natinst.com`

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

	Telephone	Fax
Australia	02 9874 4100	02 9874 4455
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 527 2321	09 502 2930
France	01 48 14 24 24	01 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Israel	03 5734815	03 5734816
Italy	06 5729961	06 57284309
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 520 2635	5 520 3282
Netherlands	31 348 43 34 66	31 348 43 06 73
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name ____

Company ____

Address ____

____

Fax (___)___________________ Phone (___)______________________________

Computer brand ________________ Model ________________ Processor___________________

Operating system (include version number) _____________________________________

Clock speed ______MHz RAM _____MB Display adapter ___________________________

Mouse ___yes ___no Other adapters installed ___________________________________

Hard disk capacity _____MB Brand ____

Instruments used ____

____

National Instruments hardware product model __________ Revision ______________________

Configuration ____

National Instruments software product ____________________________Version ____________

Configuration ____

The problem is: ____

____

____

____

____

List any error messages: ____

____

____

The following steps reproduce the problem: ______________________________________

____

____

____

____

Fuzzy Logic for G Toolkit Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

National Instruments Products

DAQ hardware ______________________________

Interrupt level of hardware ______________________________

DMA channels of hardware ______________________________

Base I/O address of hardware ______________________________

Programming choice ______________________________

LabVIEW or BridgeVIEW version ______________________________

Other boards in system ______________________________

Base I/O address of other boards ______________________________

DMA channels of other boards ______________________________

Interrupt level of other boards ______________________________

Other Products

Computer make and model ______________________________

Microprocessor ______________________________

Clock frequency or speed ______________________________

Type of video board installed ______________________________

Operating system version ______________________________

Operating system mode ______________________________

Programming language ______________________________

Programming language version ______________________________

Other boards in system ______________________________

Base I/O address of other boards ______________________________

DMA channels of other boards ______________________________

Interrupt level of other boards ______________________________

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title:	*Fuzzy Logic for G Toolkit Reference Manual*
Edition Date:	March 1997
Part Number:	321511A-01

Please comment on the completeness, clarity, and organization of the manual.

____
____
____
____
____
____
____

If you find errors in the manual, please record the page numbers and describe the errors.

____
____
____
____
____
____
____

Thank you for your help.

Name ____

Title ____

Company ____

Address ____

____

Phone (___)___________________________ Fax (___) _________________________

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
(512) 794-5678

Glossary

Prefix	Meaning	Value
m-	milli-	10^{-3}
µ-	micro-	10^{-6}
n-	nano-	10^{-9}

Numbers/Symbols

°	degrees
%	percent

B

Boolean set theory	Traditional set theory based on strict membership or nonmembership of elements to a set. Examples are TRUE or FALSE, ON or OFF, 1 or 0, and so on.

C

Center of Area (CoA)	Method of defuzzification in which the crisp output is determined by the geometrical center of the composite output membership function. Also known as Center of Gravity (CoG).
Center of Maximum (CoM)	Method of defuzzification in which the crisp output is determined by a weighted average of the maximum values of each output membership function. This method is equivalent to the Center of Area method using singleton sets.
crisp value	A finite single value such as a measured physical quantity, for example, $x = 5.3$ m.

D

defuzzification	The process of converting the linguistic output of the rulebase evaluation to a crisp controller output value.
degree of membership	A value that represents the degree of partial membership of an element to a fuzzy set. This value may range from 0 to 1 inclusive.

E

expert	A human operator of a system or process that has acquired knowledge related to controlling the process through experience.

F

fuzzification	The process of evaluating crisp controller input values (process parameters) using the defined membership functions to determine linguistic input variables for the rulebase evaluation.
fuzzy inference	The process by which the rules of the rulebase are evaluated to determine output linguistic variables for defuzzification.
fuzzy set	A set that allows for partial membership of elements. Fuzzy sets usually represent linguistic terms and are defined quantitatively by a membership function.
fuzzy set theory	An extension of traditional Boolean set theory, fuzzy set theory is based on the idea that fuzzy sets may be defined such that elements can have partial membership to the set.

L

linguistic term	A word or set of words to describe a quality of a process variable (for example, hot, very low, small positive, and so on). The term is defined quantitatively by the corresponding membership function.
linguistic variable	Defines the state of a process variable by the degree of membership of the parameter to each linguistic term defined (for example, vehicle position {left 0.0; left center 0.0; center 0.8; right center 0.1; right 0.0}).

M

Max-Min inference	Fuzzy inference method using the maximum function for the OR operator and the minimum function for the AND operator. Another common inference method is the Max-Prod, method which uses the product function for the AND operator.
Mean of Maximum (MoM)	Method of defuzzification in which the crisp output is determined by selecting a value corresponding to the maximum degree of membership of the composite output membership function. If there are multiple maximums, the mean of the corresponding values is selected.
membership function	A function that defines degree of membership to the fuzzy set over a defined universe of discourse of the variable parameter.

P

PID control	A common control strategy in which a process variable is measured and compared to a desired set point to determine an error signal. A proportional gain (P) is applied to the error signal, an integral gain (I) is applied to the integral of the error signal, and a derivative gain (D) is applied to the derivative of the error signal. The controller output is a linear combination of the three resulting values.

R

rule	A linguistic definition of a specific control action of the form IF {condition} AND {condition}... THEN {action}. For example, IF *vehicle position* is right center AND *vehicle orientation* is left up THEN *steering angle* is negative medium.
rule base	A complete set of rules defined for control of a given system. Used during fuzzy inference to determine the linguistic controller output.

S

singleton	A normalized membership function with an infinitely small width. A singleton is used to model a crisp value with a fuzzy set.

Index