



Reference Manual

S 40-Fuzzy

06/99 AWB-EM 2700-1339 GB

1st published 1998, edition 07/98

2nd published 1999, edition 06/99

© Moeller GmbH, Bonn

Author: Heribert Einwag

Editor: Ruth Walrafen

Translator: Terence Osborn

IBM is a registered trademark of International Business Machines Corporation.

All other brand and product names are trademarks or registered trademarks of the owner concerned.

All rights reserved, including those of the translation.

No part of this manual may be reproduced in any form (printed, photocopy, microfilm or any other process) or processed, duplicated or distributed by means of electronic systems without written permission of Moeller GmbH, Bonn.

Subject to alterations without notice.

Contents

1 Getting Started	2
Using fuzzyTECH:	
The Container Crane Simulation	4
Start the Fuzzy Logic Control of the Crane	13
Create a Fuzzy System from Scratch:	
The Fuzzy Design Wizard	54
2 Fuzzy Primer	62
Fuzzy Logic	63
Types of Uncertainty	63
Fuzzy Logic Technologies	72
Computing Fuzzy Systems	87
Fuzzification	89
Fuzzy Rule Inference	95
Defuzzification	98
3 Design Steps	102
System Definition	103
Definition of Linguistic Variables	103
Structure Design	116
Formulation of Fuzzy Rules	123
fuzzyTECH Analyzers	145
Index	160

1 Getting Started

This chapter guides you through the basic steps of fuzzy logic and NeuroFuzzy system design.

In the first section, you will use a built-in crane simulation to:

- Get immediate, “hands-on” experience with fuzzy logic by manually controlling the example.
- Evaluate solution performance using the *fuzzyTECH* analyzers.
- Test a given control strategy with the built-in simulation.

The second section introduces the fuzzy design wizard. Here you will:

- Create a fuzzy logic prototype from scratch.
- Become familiar with the design methodology of fuzzy logic systems.

Conventions

In order to ease your understanding of tool descriptions and the instructions, please notice the following conventions:

- All filenames are spAbbelled in caps, such as: “CRANE.FTL”, “SETUP.EXE”, ...
- All user input to *fuzzyTECH* and the process simulations are also spelled in caps.
- *fuzzyTECH* uses both the left and the right mouse buttons. “Clicking” or “double clicking” always refers to the left mouse button. Activities with the right mouse button are referred to as “click right” or “double click right.”
- In general, clicking with the left mouse button activates pointed elements; double clicking with

the left mouse button activates functions for the element. In *fuzzyTECH*, the right mouse button is only used for activating “pop-up” menus in some windows. Click right in one of *fuzzyTECH*'s windows and select the option with the left mouse button to activate a pop-up menu. The pop-up menus contain functions and options that are available only for the associated window. In most windows, these functions and options are also accessible through toolbars. You can enable or disable toolbars in the Preferences dialog, which is activated with “Options\Preferences”. This way of using mouse buttons and clicks is similar to most other MS-Windows software, such as MS-Word or MS-Excel.

- The “main menu” is the upper menu bar in the main window of *fuzzyTECH*. Activation (clicking) of main menu entries opens a pull down menu for the entry. This pull down menu lists the available functions for the entry. These options in the main menu are referred to by path names. For example, “File\Open” refers to the option “Open” of the main menu entry “File”.

Using *fuzzyTECH*: The Container Crane Simulation

This section demonstrates basic usage of the *fuzzyTECH* software with a crane control example. The design of this example is explained in more detail in the section “System Definition”. You can find the basics of fuzzy logic explained in the section “Fuzzy Logic”. If you are not familiar with the concepts of fuzzy logic, please read this section before proceeding. The following description has been structured so that you can follow it in parallel on your PC.

Tool Introduction

The description of the *fuzzyTECH* functionality in this section is very brief and covers only the most important features pertinent to the crane simulations. You may obtain more detailed information by using the online help system integrated with *fuzzyTECH*. You can activate the help system either by pressing the [F1] key or utilizing the “Help” entry in the main menu. A complete reference is contained in the *fuzzyTECH* Reference Manual.

Visible Elements

Depending on the options enabled in *fuzzyTECH*, window elements such as status bars, tool bars, list boxes and others may be visible or not. Hence, the windows depicted as screen shots in following pages may look slightly different from those you see while working with *fuzzyTECH*.

fuzzyTECH Program Group

The installation procedure of your *fuzzyTECH* Edition creates a new program group “*fuzzyTECH*” in the Program Manager of MS-Windows. Figure 1 shows this program group.



Figure 1–1: The setup routine *SETUP.EXE* automatically creates the program group *fuzzyTECH* in the MS-Windows program manager. This program group contains your *fuzzyTECH* edition and the software simulations.

Start the Container Crane Simulation

First, start the container crane simulation, for example, by double clicking the crane symbol “Crane” in the *fuzzyTECH* program group.

Figure 1–2 shows the crane simulation window. The container (yellow) is already picked up from the ship (red). It must be positioned over the truck (green).

Process Values

The box visible in the lower part of the crane simulation window always shows the actual values for Angle, Distance and Power. The values of Angle and Distance are computed by the process simulation, while Power is the control variable either set manually or by the fuzzy logic controller. Enable the manual control mode by clicking the [Manual] button. The Distance is more than 20 yards, and Angle and Power are zero.

Manual Control

Use the [-], [0] and [+] buttons to control the Power. First, click once or twice on the [+] button. This sets the motor power to 0.75 or 1.5 kilowatts, respectively. The jerk caused by this makes the Angle oscillate lightly, but is not sufficient to put the

crane in motion due to the friction in the mechanical system. At least 3 kilowatts motor power are required to set the crane head in motion. Now, further increase the motor power by clicking on the [+] button again. The [0] button resets the motor power setting to zero and the [-] button lets you apply negative motor power for braking. You can also use the [-], [0] and [+] buttons on your keyboard. The [Reset] button sets the crane to its start position.

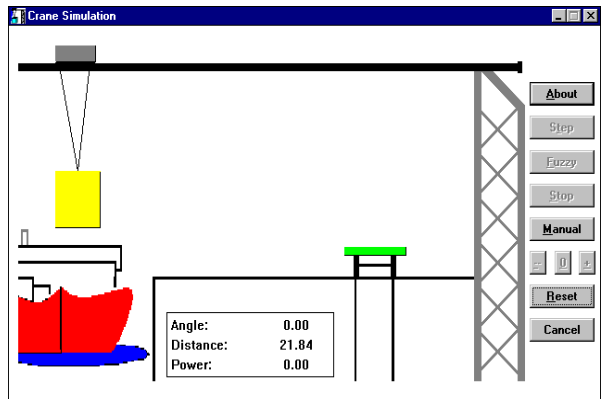


Figure 1–2: The crane simulation window animates the crane operation. The buttons let you either operate the crane by hand or establish a link to fuzzyTECH for automatic operation.

Control Strategy

When you start the crane with very high motor power, you see that, due to the large weight of the container, the “container drives the crane” rather than vice versa. This strong feedback of the load to the drive is typical for container cranes, and it causes one of the major difficulties for the control of such processes.

A Crane Operator

The crane operator must position the container over the truck, so the container can be released. The container must not sway as releasing a swaying container could damage its contents. There exist two simple strategies to position the container without sway over the truck. One is to move the crane head so slow that the container never starts to sway. Since you are safe from wind gusts with the software simulation of the crane, this certainly works. However, it may take a long time to reach the target. The other simple strategy is to start with full power and position the crane head over the target position then wait until the container stops swaying. This works since the sway is of no harm during transportation and there are no wind gusts. However, it also takes far too much time.

Both of these simple strategies are not practical for container cranes. The opportunity costs of a container ship tied up in a harbor go into thousands of dollars each hour. Hence, loading and unloading must be completed within a minimum amount of time. Only an anti-sway control strategy of the container crane operator can achieve this.

Try this yourself: start with medium power. If you apply full power right from the start, the container starts to sway very strongly and takes a long time to stabilize later. If the container sways a little behind the crane, you can further increase the power

because this situation is stable. The increased power gets the container to the target faster. Watch the container as it moves. If it sways ahead of the crane head, increase the power. This reduces the sway and returns the container to a stable position. It also moves the container to the target position faster. If it sways too far back, decrease power to reduce the sway as it is harder to reduce a large sway later. The section “Fuzzy Logic Technologies” contains more details on crane controller engineering.

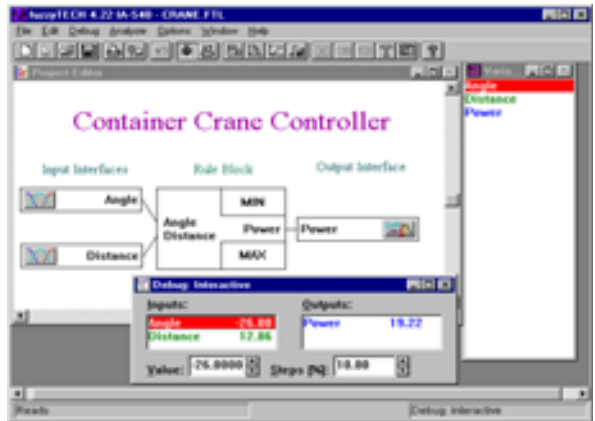




Figure 1–3: Main window of fuzzyTECH editions after start-up.

Start fuzzyTECH

You have to start your *fuzzyTECH* Edition first in order to use the pre-defined fuzzy logic controller for the container crane simulation. Double-click the icon “*fuzzyTECH* Edition” in the Program Manager. Upon start-up, the *fuzzyTECH* main window contains two sub-windows: the “Project Editor” and the “Variables” window (figure 1–3).

These two windows can never be closed in *fuzzyTECH* nor can a second set of these windows be created. However, the  or the  button minimizes these windows. Underneath the main menu, a gray toolbar provides quick access to the most often used *fuzzyTECH* functions.

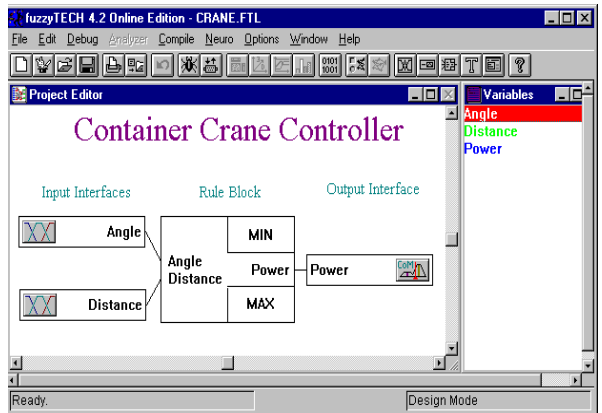


Figure 1–4: After loading the project CRANE.FTL, the Project Editor shows the structure of the controller; all defined linguistic variables are listed in the variables window.

Status Bar


At the lower border, a gray status bar provides quick information about current settings, modes, and operations of *fuzzyTECH*. The status bar consists of three fields. The left field indicates basic operations of *fuzzyTECH*, such as loading, saving, or code generation. When *fuzzyTECH* is idle, this field displays “Ready”. The left field has a second function: when the mouse pointer moves over a button of the toolbar, the left field displays a short description of button’s function.


Design Mode

The right field of the status bar always displays the current debug mode. Upon start of *fuzzyTECH*, no debug mode is active and thus the right field displays “Design Mode.” The middle field displays the progress of complex *fuzzyTECH* operations, such as loading, saving or the current state of a data transmission.

Project Editor

At the end of the loading procedure, *fuzzyTECH*’s Project Editor window shows the structure of the crane controller (figure 1–4). The Variables window lists all linguistic variables of the crane controller. “Angle” and “Distance” are the output variables (measured variables) of the crane simulation and, hence, the input variables of the fuzzy logic controller. Power is the input variable (command variable) of the crane simulation and, hence, the output variable of the fuzzy logic controller. Linking the respective inputs and outputs of the crane simulation and the fuzzy logic controller yields a closed control loop. The Debug:RCU window will always be displayed.

This window always shows the current values of all input and output values of the fuzzy logic controller. You should minimize this window by clicking the .

or  button when you start *fuzzyTECH* to reduce the overview. This also expedites the simulation on slow PCs.

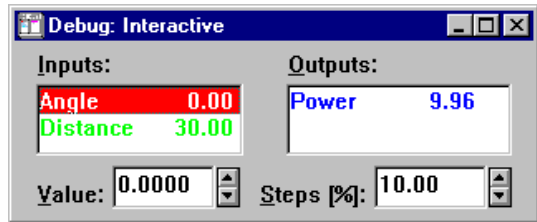


Figure 1–5: The Debug window is active in any Debug Mode and always shows the values of the input and output variables. In interactive Debug Mode, the edit fields in the lower part of the Debug window allow you to set values for the input variables.

Start the Fuzzy Logic Control of the Crane

After you have successfully established the link from the simulation to *fuzzyTECH*, the simulation starts using the fuzzy logic controller in *fuzzyTECH*. In an infinite loop, the crane simulation computes the current values for Angle and Distance and sends them to *fuzzyTECH* where they are the inputs of the fuzzy logic controller. *fuzzyTECH* then computes the value of the output variable Power and sends it back to the crane simulation. The crane simulation uses the value of Power to compute the reaction of the crane a time unit later and displays the new situation in its window. This sequence repeats itself until either the [Stop] or [Reset] button of the simulation is pressed. Depending on the computation and video performance of your PC, you see this loop as a smooth movement of the container crane.

First, the fuzzy logic controller starts with medium power, then speeds up as the container sways a

little behind the crane head. As the container is a very heavy one, it sways further back as a result of this power increase. As a reaction, the fuzzy logic controller reduces the motor power to reduce the sway. Upon reaching the target position, the fuzzy logic controller positions the container directly over the target with one overshoot. The button [Reset] puts the container back in the start position. The button [Stop] lets you halt the simulation at any time. Pressing [Fuzzy] continues the simulation, and the [Step] button lets you single-step through the operation.

Analyze Time Response

The graphical simulation gives you an overview of how the fuzzy logic controller controls the crane. For more in-depth analyses, *fuzzyTECH* provides numerous tools and analyzers. *fuzzyTECH* provides Time Plots to evaluate the time response. In order to create a new Time Plot, select “Analyzer\Time Plot...” to activate the Configuration dialog as shown in Figure 1–6. The three left list boxes display elements that you can select for display. The right list box titled “Plot Items” shows the elements currently selected for this Time Plot. Just double-click on the respective element to move elements from the left list box to the right one. Otherwise, you can highlight the elements you want to move and then use the [<<] and [>>] buttons to move the highlighted elements.

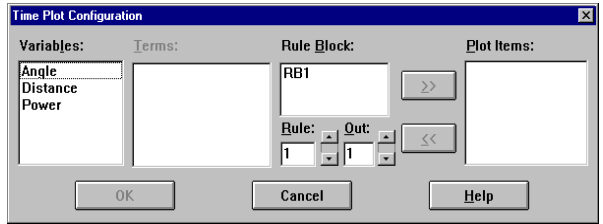


Figure 1–6: The Time Plot configuration dialog lets you specify Linguistic Variables, Terms and Rules for display.

For the crane controller the leftmost list box “Variables:” shows the three input and output variables “Angle”, “Distance”, and “Power” in alphanumeric sequence. Double-click on all three of these variables to move them in the “Plot Items:” list box. The list box “Terms:” is empty since the crane controller does not contain any linguistic variables without membership functions. The list box “Rules:” lets you select individual rules for display. Do not select any rules now. After you have selected the three linguistic variables for display, click on the [OK] button to close this window. Figure 1–7 shows the resulting Time Plot. You may define up to 10 Time Plots in each *fuzzyTECH* session. Each Time Plot is uniquely identified by a number in its title bar.

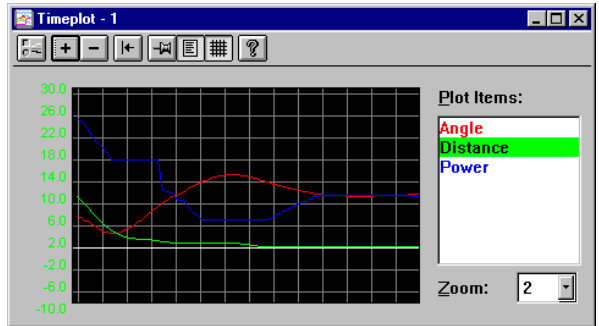



Figure 1–7: The Time Plot lets you analyze input and output variables, as well as rule firing degrees over time.

On the right side of the Time Plot window, a list box shows all selected Plot Items. The left part of the window displays the plot area. Change to the crane simulation window and start the simulation with the [Fuzzy] button. You now see that the Time Plot window plots the value of all three control variables over time. A scale for the element highlighted in the Plot Item list box is displayed on the left border of the plot area. A dotted white horizontal line in the plot area shows the zero line if it is in the displayed range. Each Time Plot also has its own toolbar that can be disabled to save screen space.

Next, reset the simulation and press [Fuzzy] while watching the Time Plot window. The blue line (Power) starts with medium positive Power. After the container sways slightly back (red line declines), the fuzzy logic controller increases the motor power (blue line increases). When the sway becomes too large (red line deep down), the fuzzy logic controller reduces motor power (blue line declines). After reducing the sway (red line comes back up), the fuzzy logic controller increases the motor power again. The constantly declining green line (Distance) shows the progress toward the target.

Customizing Time Plots

You may set the resolution of the time axis in the “Zoom:” drop list to customize the Time Plot. Also, the buttons [+] and [-] in the tool bar let you change the zoom. You may use the Freeze button in the toolbar to freeze the display. You may also use a button from the tool bar to hide the Plot Items to save screen space. Similarly, the grid in the plot area may be turned on and off by the Grid button. The Reset button clears the plot area. Click the Configuration button to add or delete elements from the Time Plot. Move the mouse pointer over the buttons without pressing a mouse button to see a quick description in the left field of the status bar.

You can also access the toolbar functions through the pop-up menu activated by clicking right somewhere in the Time Plot window. You can turn the Time Plot toolbars on and off in the Preferences dialog box activated by selecting “Options\Preferences...”. Double-click on the system icon or when using Windows 95 click  to close a Time Plot. If you leave the debug mode, all Time Plots are automatically closed. If you have enabled “Save Window Position” in the Preferences dialog, all Time Plots still opened upon leaving the debug mode are automatically reopened when you reenter debug mode.

Time Plot of Rule Firing Degrees

You may also plot the firing degrees of individual rules. Open a new Time Plot by selecting “Analyzer\Time Plot...” then double-click on the “RB1” entry in the “Rule Block:” list box. RB1 is the name of the one and only rule block of the container crane controller. The list box “Plot Item:” now displays the entry “RB1.1.1”. The first number after the point identifies the number of the rule in the respective rule block; the second number is the output variable of the rule block to be plotted. Hence, “RB1.1.1” identifies the first rule in rule block RB1 and the first output variable of the rule.


Next, select the second rule for display. Enter “2” in the field “Rule:” and double-click on the entry “RB1” in the “Rule Block:” list box. After pressing [OK] a second Time Plot opens that plots the firing degrees of the first two rules. Start the crane simulation again and you observe that initially the first rule fires, then the second one. When a rule is highlighted in the “Plot Item:” list box of the Time Plot, the respective rule is shown abbreviated under the plot area. You may mix any combination of linguistic variables, terms and rules in a Time Plot.

Analyzing Control Surfaces

In addition to analyzing time response, you may also analyze the control surfaces of the fuzzy logic controller. Close the two Time Plot windows to save video and computing resources if you have a low-performance PC. Select the option “Analyzer\3D Plot...” from the main menu. The 3D Plot as shown in Figure 1–8 displays the control surface of the crane controller. The 3D Plot always shows two input variables on the horizontal axis and one output variable in the vertical axis. Since the crane controller has precisely two input and one output variables, the Configuration dialog of the 3D Plot is omitted in the case of the crane controller. You may open up to 10 3D Plots in one *fuzzyTECH* session.

Customizing the 3D Plot

The 3D Plot also has its own toolbar. Click on the rightmost arrow-down button to change the resolution of the 3D Plot. This displays a list of the possible resolutions. A high resolution can result in slow response and redraw times of the time plot, depending on the video and computing performance of your PC. You can select the output variable using the second rightmost field and the two input variables using the two fields on the left. The four arrow buttons on the left side of the toolbar rotate the plot in all directions. A double click on an arrow button starts a (continuous) rotation; the hand button stops this rotation. The double-arrow buttons flip the 3D Plot for better visibility.

If the output variable selected for the 3D Plot depends on more than the two inputs displayed, the control surface exhibits dynamic dependence on these variables. If the Repaint button is pressed, the 3D Plot is repainted every time one of these other input variables changes to reflect the change in the control surface shape. Note, this only results in a smooth display if you have a PC with very high video and computing performance. Double-click on the system icon or when using Windows 95 click  to close a 3D Plot. If you leave the debug mode, all 3D Plots are automatically closed. If you have enabled "Save Window Position" in the Preferences dialog, all 3D Plots still opened upon leaving the debug mode are automatically reopened when you reenter debug mode.

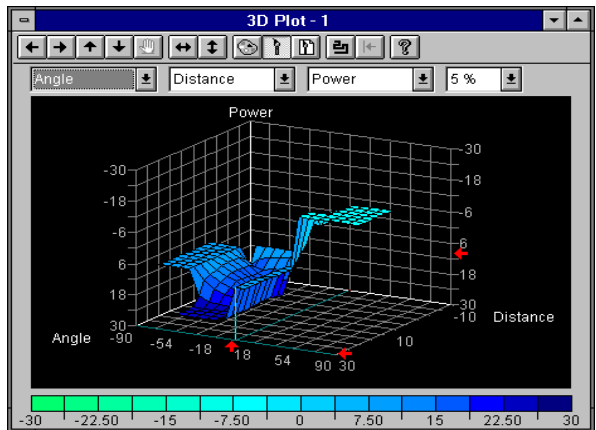


Figure 1–8: The 3D Plot draws the transfer surface for two input variables and one output variable.

Display of Operation Points

The red arrows at the three axes and the cyan plot lines always show the current value of the input and output variables. The Tracing button lets you also trace the operating point over time (green trace). Enable Tracing then reset and start the crane simulation. The “green trace” starts at the initial position of the crane, Angle=0 and Distance=22, and moves toward the target position Angle=0 and Distance=0. The height of the control surface and its color indicate the reaction of the fuzzy logic controller to the combination of the input variables.

With the Time Plot and 3D Plot, you analyze the “outside” performance of the fuzzy logic controller by using input and output variables; in the next steps, you analyze the internal structure. The RCU debug mode is closed remotely by closing the crane simulation.

Linguistic Variable Editors

The Variables window contains a list of all linguistic variables in the system. Simply double-click on the a variable name in the Variables window to activate an editor for the respective variable. You may open an independent editor window for each linguistic variable. Figure 1–9 shows the editor window for the linguistic variable “Angle”. Click on the “New Linguistic Variable” button in the main toolbar or use the pop-up menu in the Variables window to create a new linguistic variable.

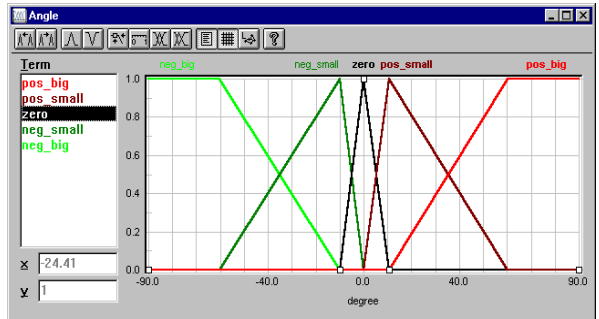


Figure 1–9: Variable Editor window for “Angle”.

Customizing the Variable Editor

The variable editor shows a plot area of all membership functions defined for the linguistic variable. Membership functions are always defined by definition points that can be freely positioned and dragged. Above the plot area, the term names are displayed. Each name is assigned to the “peak area” of the defined membership function. In addition, the left side of the variable editor can be extended with a list box showing the terms for the linguistic variable. A term can be selected by clicking the term name either above the plot area or in the list box. The definition points of the membership function of the selected term appear as small squares in the plot area. You may highlight individual definition points with mouse click and then drag them. A highlighted definition point is indicated by a filled square.

Point Definition of Membership Functions


In order to erase a definition point, select it by highlighting the point with a mouse click and then press the [Del] key. The confirmation dialogs that pop up any time you erase a system component can be avoided by disabling the “Enable Confirm Dialogs” option in the Preferences dialog box. The

Preferences dialog box is activated by selecting “Options\Preferences...”. Most experienced users prefer to disable confirmation messages since *fuzzyTECH* provides a multi-stage UNDO function. You can undo most actions by pressing [Alt]-[↓] (Alt-Backspace) or selecting “Edit\Undo” from the main menu.

Operate on Multiple Points

When performing operations on points you may select more than one point at the same time. If you want to select individual, non-connecting points, select the first point and then press the [Ctrl] key while you click on the other points you want to select. The selected points can then be dragged or deleted at the same time. If you want to select a group of connecting points, click on the leftmost point of the group then keep the [Shift] key (⇧) pressed while you select the rightmost point of the group. Double-click on an empty space in the plot area to create a new definition point for the selected term.

Using the Grid Function

The variable editor provides a grid function to ease the placement of definition points. Click on the Grid button  in the toolbar to change the grid. Figure 11 shows the Grid dialog box. The grid resolutions for the base variable (horizontal axis variable) and membership degree (vertical axis) can be chosen independently. Enter the value “5” for “Base Variable”. This sets the grid resolution in intervals of 5 yards, resulting in 8 possible values for Distance (40 yards total distance / 5 yards resolution = 8 intervals).

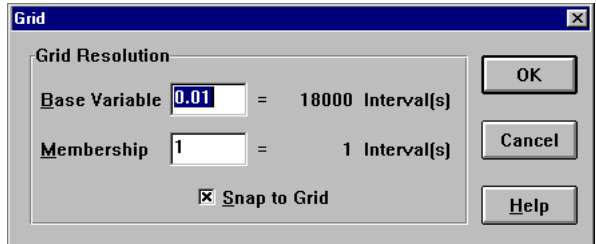



Figure 1–10: The Grid dialog lets you specify different resolutions for base variable values and membership degrees.

Enter the value “0.1” for “Membership”. This allows the membership degree of definition points to be set at values of 0, 0.1, 0.2, ... Enable the check box “Snap to Grid” and close the Grid dialog box by clicking [OK]. If you now drag definition points, only grid positions are considered.

The fields “x” and “y” show the coordinates of the nearest grid position to the current mouse cursor position while no definition point is highlighted. If one definition point is highlighted, the “x” and “y” fields show its position. You can now also enter new coordinates for this point in the fields. If more than one definition point is highlighted, the fields show the coordinates of the definition point which the mouse cursor is over. If you do not want to use a mouse, you can also highlight definition points by the [Tab] and [Space] keys. Then use the cursor keys to move the definition point and the [Return] key to place it.

Base Variables

In order to modify the properties of the base variable for which the linguistic variable is defined, click on the Base Variable button  in the toolbar of the variable editor. This activates the Base Variable dialog as shown in Figure 12. The fields “Min:” and “Max:” let you specify the universe of the base variable. Two representations exist in *fuzzyTECH*: Shell Values and Code Values. Shell Values represent the universe of the values that are used for display in all editors and analyzers of *fuzzyTECH*. If you have selected “Double” as “Base Variable Data Type” in the Global Options dialog box (select “Options\Global Options...” to activate this dialog box), the same values are used in the generated code.

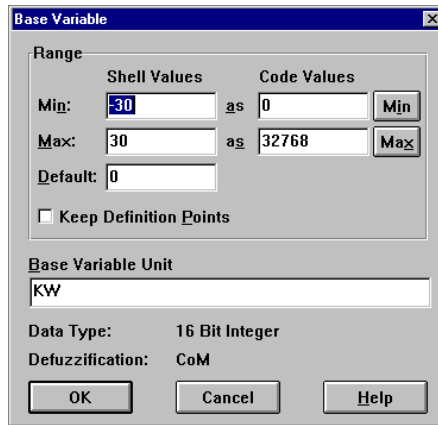




Figure 1–11: The Base Variable dialog lets you specify the range and code value representation of a linguistic variable.

However, the use of double resolution float variables is of prohibitive computational effort for most microcontrollers and other real-time process controllers. Also, many compilers used with these

target hardware platforms cannot provide libraries for this computation. Hence, the column “Code Values” lets you specify integer values for the minimum and the maximum of the variable in the generated code. The possible range for these integer values depends on the base variable data type. Assume, you want to implement the crane controller on a 16-bit microcontroller. Your distance sensor delivers values from 822 to 3222 for a distance of -10 and 30 yards respectively. If you specify 822 and 3222 as code values, *fuzzyTECH* automatically converts these values into the shell values used for display in all editors and analyzers.

The reason for this automatic range conversion is that the code generators of *fuzzyTECH* can tie in the range conversion with the fuzzification code. This saves both on conversion code, as well as computing steps, and makes the code produced more efficient. If you do not need this automatic range conversion, click the [Min] and [Max] buttons to achieve maximum integer resolution. If no rule for this variable fires and it is used in an output interface, then the “Default” field lets you specify a default value that is output by the fuzzy logic system. The field “Base Variable Unit” lets you specify a unit string for the base variable. This string is used for display purposes only. Leave the Base Variable dialog by clicking [Cancel] to return to the variable editor.

In the variable editor you can also hide the list box “Term” using the Listbox button  of the toolbar. This is useful to save screen space once the membership functions of the variable are defined. You may also hide the toolbar by disabling the “Variable Editor” option in the Preference dialog. All toolbar functions are accessible by the pop-up menu of the variable editor. Click right somewhere in

the variable editor window to activate this pop-up menu. Double-click the system icon or when using Windows 95 click  of the variable editor window to close a variable editor.

Editing Linguistic Terms

Double-click on the term name in the list box “Term” of the variable editor to change the properties of an existing term. Figure 1–12 shows the Term dialog box. The term name is edited in the “Term Name” field.



Term names may only contain alphanumeric characters and underscores because the names are used in the generated code. The [Color...] button lets you change the colors that *fuzzyTECH* automatically associates to terms.

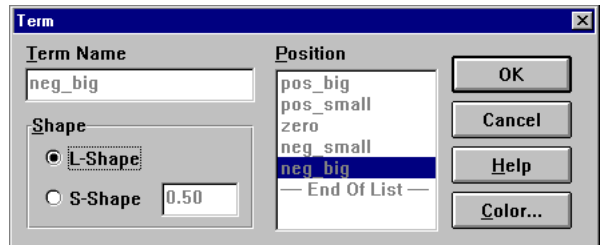





Figure 1–12: The Term dialog lets you specify a term’s name, sequence and membership function shape.

As mentioned before, *fuzzyTECH* uses point definition of membership functions. Two alternatives exist in *fuzzyTECH* to connect these definition points: linear and S-shape. You can select between these two types using the respective radio buttons. For S-shape membership function definition, you also can specify the asymmetry factor in the range from 0 to 1.

In order to create a new term, click on the New Term button  or the Inverse Term button  of the variable editor's toolbar. The Inverse Term tool creates a new term with a membership function defined as $\mu_{\text{new}}=(1-\mu_{\text{highlighted}})$.

For Standard MBFs, an expedited design alternative exists. Define membership functions where the maximum point is at the appropriate position then press the Standard MBF button  and all other definition points are automatically set. You may press the Standard MBF button at any time to convert any membership function definition into a Standard MBF function.



The multi-stage UNDO function of *fuzzyTECH* lets you take back even complex “experiments.”

Editing Systems Structure in the Project Editor

You define the actual structure of the fuzzy logic controller in the Project Editor window. Three types of objects exist in the Project Editor: Interfaces, Rule Blocks and Remarks. Remarks are pure text objects that have no impact on the actual information being processed. Interfaces can also contain fuzzification and defuzzification. Rule Blocks contain the fuzzy logic rules of a system design. These elements suffice to design even complex and hierarchical fuzzy logic systems.

All objects in the Project Editor window can be placed arbitrarily by mouse drag and drop. Keep the [Shift]/[L] key pressed to fine position the objects. The mouse movement is then limited to the horizontal or vertical direction, depending on in which direction the move started. If you do not want to use the mouse, you can highlight any object using the [Tab] key, moving it with the cursor keys, and placing it with the [Return] key. Edit the properties of an object by double-clicking on the object.

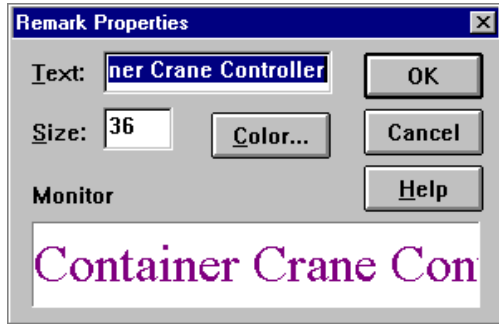


Figure 1–13: The Remark Properties dialog lets you specify text objects.

Double-click on the purple text “Container Crane Controller” in the Project Editor window. This activates the Remark Properties dialog as shown in Figure 1–13. The text to be displayed is entered in the field “Text:”. You can also specify font size (field “Size:”) and text color (by pressing the [Color...] button). Leave the dialog by pressing the [Cancel] button. Next, double-click on the interface with the variable name “Angle”. This activates the Interface Options dialog as shown in Figure 1–14. The group “Type” lets you define the fuzzification or defuzzification method. In the Project Editor window, each method is represented by a different icon. The fuzzification and defuzzification selection also determines whether the interface shall act as an input or output interface. In the Project Editor window, you can differentiate input and output interfaces by the location of the method icon. With input interfaces, the icon is on the left side of the interface box; with output interfaces it is on the right.

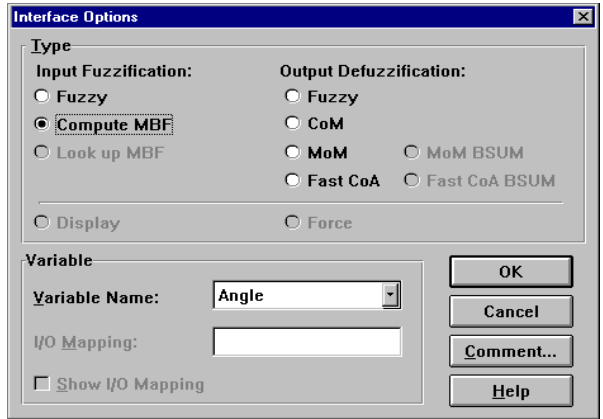


Figure 1–14: The Interface Options dialog lets you define the properties of an interface by fuzzification/defuzzification type.

Input Interfaces

For input interfaces, three different fuzzification methods are supported. “Fuzzy Input” indicates that the variable is passed into the fuzzy logic system as a “fuzzy” variable, which is as a vector of membership degrees. This is primarily used in decision support applications where the input stems from a linguistic source rather than a technical sensor. The fuzzification method, “Compute MBF”, is the standard fuzzification method used in almost all applications. This method only stores the definition points of the membership functions in the generated code and computes the fuzzification at runtime. Some *fuzzyTECH* MCU Editions (dedicated assembly code generation for microcontrollers) also support the “Look up MBF” method. This method computes the membership functions completely at compile time and stores all values in a table in the generated code. This method results in extensive code sizes and is only provided where it delivers expedited computation.

Output Interfaces

For output interfaces, different defuzzification methods exist as well. Analogous to a “Fuzzy Input”, “Fuzzy Output” outputs a vector of the membership degrees of the inference result. The most often used method is Center-of-Maximum (“CoM”), which delivers the best compromise of the firing rules. In contrast, Mean-of-Maximum method (“MoM”) delivers the most plausible result. Center-of-Area (“CoA”) is similar to the CoM method. *fuzzyTECH* uses a modified CoA algorithm, “Fast CoA”, which circumvents the numerical integration required by the original CoA method. The BSUM variants of CoA and MoM defuzzification use the bounded sum rather than the maximum operator to aggregate the individual area pieces.

Assign Variables

Use the drop list “Variable Name” to select a variable for the interface. Since all variables are already used in interfaces, this drop list only contains “Angle”. The “I/O Mapping” lets you dynamically link *fuzzyTECH* variables to external variables. This is used in process control systems or PLC implementations of *fuzzyTECH* to interface the fuzzy logic system to variables and periphery. In other *fuzzyTECH* Editions, the dynamic link of variables is not supported and this group is disabled. Close the dialog box by clicking [Cancel].

Definition of Fuzzy Logic Rules

The large block in the middle of the Project Editor window is the Rule Block. This block contains the rules of the system describing the control strategy. Double-click on the Rule Block to activate the Spreadsheet Rule Editor as shown in Figure 16. Each row corresponds to a single fuzzy logic rule. The leftmost column assigns a number to each rule (gray fields). Clicking on this field highlights the rule;

clicking again de-highlights it. The next two columns comprise the “if-part” of the rule. Above the columns are two buttons [Angle] and [Distance], above them the button [IF]. The two rightmost columns under the [THEN] button describe the “then-part” of the rules. The right one under the [Power] button denotes the result term; the left one under the [DoS] button denotes the individual weight of the rule. The weight (Degree-of-Support) lies in the range from zero to one.

	IF		THEN	
	Angle	Distance	DoS	Power
1	pos_small	zero	<input type="text" value="1.00"/>	neg_medi
2	zero	zero	<input type="text" value="1.00"/>	zero
3	pos_small	close	<input type="text" value="1.00"/>	neg_medi
4	zero	close	<input type="text" value="1.00"/>	neg_medi
5	neg_small	close	<input type="text" value="1.00"/>	pos_medi
6	neg_small	medium	<input type="text" value="1.00"/>	pos_high
7	neg_big	medium	<input type="text" value="1.00"/>	pos_medi
8	zero	far	<input type="text" value="1.00"/>	pos_medi
9	neg_small	far	<input type="text" value="1.00"/>	pos_high
10				

Figure 1–15: The Spreadsheet Rule Editor supports the definition of rule sets.

Add and Modify Rules

Simply click in the respective field to change a rule. A list box shows all possible values for the field. Use the last row of the table to add a rule. Once you have entered a new rule, another empty row appears at the end of the table. Incomplete rules, that is, rules that either have no “if-part”, no “then-part”, or no “DoS”, are automatically deleted when

the Spreadsheet Rule Editor is closed. Empty fields in the “if-parts” are considered as “don’t care” conditions. A rule that contains an empty field is not influenced by the respective variable. Rules containing “don’t care” conditions are not incomplete. In the list box that appears when you click on a field, the “don’t care” condition is listed under the terms as “[...]”.

Aggregation

In order to delete a rule, first highlight the rule by clicking the gray left field and then press the [Del] key. If you want to temporarily disable a rule, you can also set the DoS to zero. Such a rule has no effect on the system but is not considered incomplete.

Click on the [IF] button to activate the “Aggregation” dialog as shown in Figure 1–16. In this dialog, you specify the aggregation operator. The group “Operator” lets you select the operator family, the “Parameter” scroll bar and edit field lets you tune the characteristic. The closer the parameter value is to 0, the more the operator performs like a perfect AND; the closer the value is to 1, the more the operator performs like a perfect OR. Close the dialog by clicking [Cancel].

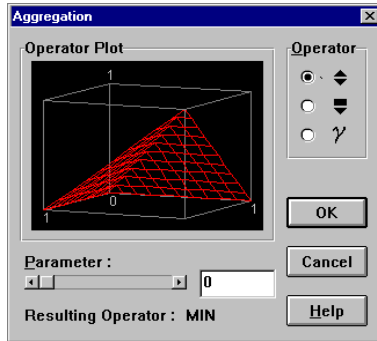


Figure 1–16: The Aggregation dialog lets you specify the fuzzy logic operator used for computing the “if-part” of a rule.

Result Aggregation

Click on the [THEN] button to change the result aggregation. *fuzzyTECH* supports two methods for result aggregation, the maximum method and the BSUM method. If more than one rule has the same result, the first method takes the maximum of the two as the final result and the second one takes the bounded sum.



BSUM result aggregation is different from BSUM MoM and BSUM CoA. The bounds are zero and one. For more details on computation of aggregation and result aggregation methods, refer to the section “Computing Fuzzy Systems”. The usage of the operators is depicted in the section “Design Steps”.

Sort Rules

Just click on the respective button, such as [Angle], [Distance] or [Power], to sort rules in the sequence of the terms of a certain variable. You may also sort rules in the sequence of their firing degree by pressing the [DoS] button. The sequence of rules shown in Figure 1–15 results from pressing the buttons [Angle] first and [Distance] second. Sort the rules in this way, since the rules are later referenced by their number.

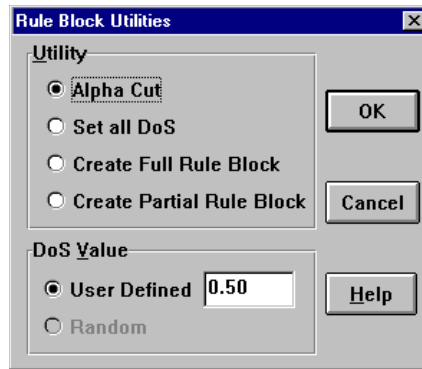



Figure 1–17: The Rule Block Utilities dialog contains a variety of functions to manipulate an entire rule block.

Rule Block Utilities


Click on the button  to access the Rule Block Utilities dialog box shown in Figure 1–17. In the group “Utility”, select the desired function:

- “Alpha Cut” deletes all rules with a DoS lower than the value specified in the group “DoS Value”.
- “Set all DoS” forces the rule weights to the value specified in the group “DoS Value”.
- “Create Full Rule Block” deletes all existing rules and creates a rule for each combination of variable terms. Thus, for each possible

combination of rule block input variable terms, a rule is generated for all output variable terms. The weights of the generated rules are specified in the group “DoS Value”.

- “Create Partial Rule Block” only creates a rule for each combination of input variable terms. You have to specify the output variable terms for each rule manually. The rules for which you do not specify an output term are incomplete and are erased when you close the Spreadsheet Rule Editor. The weights of the generated rules are specified in the group “DoS Value”.
- Close the dialog box by pressing the [Cancel] button.

Matrix Rule Editor

Click the  button to activate the Matrix Rule Editor as shown in Figure 1–18. This editor is only available for rule blocks that do not contain rules with “don’t care” conditions. Many experienced fuzzy logic designers prefer the Matrix Rule Editor over the Spreadsheet Rule Editor when designing complex systems.

The lower part of the window has a list box for each input and output variable of the rule block. Each list box shows all terms that are defined for the respective variable. By highlighting a term in each list box, a single rule is addressed. The weight of this rule is shown on the scroll bar and in the edit field in the group “Degree-of-Support”. Rules that are non-defined are identical to zero-weighted rules in the Matrix Rule Editor. In order to add a rule in the Matrix Editor, address the rule by highlighting the respective terms in the list boxes and set the DoS to a non-zero value. In order to erase a rule in the Matrix Editor, address it and set its DoS to zero.

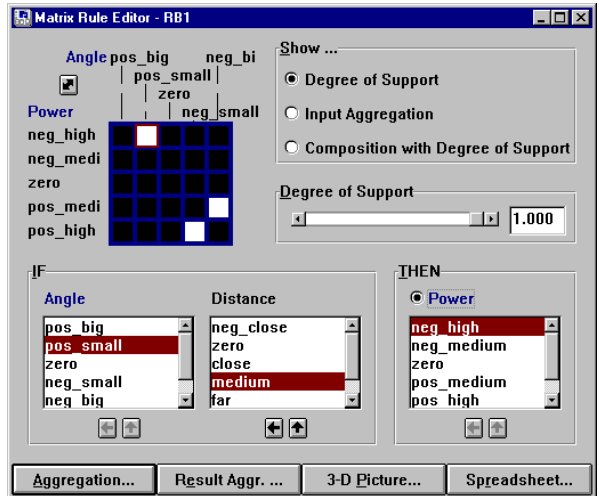



Figure 1–18: The Matrix Rule Editor displays a slice through the multi-dimensional rule space as a matrix.

You may select two of the variables for matrix display in the upper left part of the window to visualize the rule base. Use the two arrow buttons below each list box to select variables for the matrix. The arrow buttons of the two variables currently displayed in the matrix are disabled. Use the left arrow button to select the variable as the vertical matrix variable and the up arrow button to select the variable as the horizontal matrix variable. Use the double-arrow button in the upper left corner to flip the positions of the two variables displayed in the matrix.

Next, select the linguistic variable “Power” as the horizontal variable by pressing the up arrow beneath the “Power” list box. Then, highlight different terms in the list box “Distance”. For each of the selected distances, the matrix shows the relation between the angle input and the power output. In the matrix, rules with a DoS of 1 are shown as white squares and rules with a DoS of 0, as well as non-existing rules, as black squares. Rules with a DoS between 0 and 1 are shown with a respective shade of gray. The matrix field that corresponds to the addressed rule is identified by a red square. You may also address rules by clicking the respective matrix field. You may double-click the respective matrix field to toggle the DoS of a rule between 0 and 1. This is faster than using the scroll bar.

You may also visualize the rule matrix three-dimensionally by clicking the [3-D Picture] button. The group “Show...” lets you select what the matrix should display in any Debug Mode:

- Degree of Support shows the weight of the rules.
- Input Aggregation shows to what degree the condition for the rules is true.
- Composition with Degree of Support shows to what degree the output of the rule fires.

The [Spreadsheet] button lets you go back to the Spreadsheet Rule Editor. Next, close the Matrix Rule Editor with a double-click on the system icon or click on  when using Windows 95.

If you have more than one rule block in your project, you may simultaneously open a rule block editor for each block. In order to differentiate between different rule block editors, the name of the rule block is shown in the title bar.

Test and Verification – Using Debug Modes

Now that you have evaluated the various editors used to design a fuzzy logic system, the next development steps are optimization and testing. *fuzzyTECH* features numerous debug modes to support this development step. All debug modes cooperate with the editors and analyzers you have learned to use to expedite system verification.

When you switch from Design Mode to any of the debug modes, the entire fuzzy logic system developed is simulated by *fuzzyTECH*. All editors become dynamic, that is, they graphically display information flow, fuzzification, defuzzification, and rule inference. In addition, most system components may still be edited during debugging. First, a quick overview of all debug modes:

Overview on Debug Modes

Interactive

The Interactive Debug Mode shows the reaction of the system to the input values that you set. All editors and analyzers of *fuzzyTECH* graphically show the information flow. When you modify any elements of the fuzzy logic system, you can see the effect of the changes instantly. This allows quick “if-then” analysis. The Transfer Plot and the 3D Plot let you test the completeness and ambiguity of the rule base.

RCU, DDE, fT-Link

These debug modes allow you to dynamically link *fuzzyTECH* to application software or to software simulations of the process to be controlled. All simulations contained with the software use these interfaces. RCU and DDE can be used to integrate *fuzzyTECH* into standard windows application software. For details on how to program RCU or DDE, refer also to the *fuzzyTECH* Reference Manual.

Serial Link

The Serial Link Debug Mode allows *fuzzyTECH* to operate over a serial port. The settings for communications must be set in the Serial Port dialog that you access via “Options\Serial Port..”. In Serial Debug Mode, *fuzzyTECH* acts as a slave to the process or simulation connected at the serial port of your PC. After activation of Serial Debug Mode, *fuzzyTECH* waits for the process or simulation to write all input values of the fuzzy logic system to the serial port. When all input values are received, *fuzzyTECH* computes the output values and writes them back to the serial port. Then, *fuzzyTECH* waits for the next input. The communication format is ASCII: ASC(31) is used as

delimiter between values; ASC(0) is used to indicate end of transmission.



The Serial Debug Mode can only be used with slow systems because the response time for *fuzzyTECH* sending back the output values cannot be guaranteed (due to the MS-Windows operating system). For real-time applications, the more sophisticated Online Debug Mode should be used.

File Recorder

The File Recorder lets you use input data stored in files as input to the fuzzy logic system. These input data files may stem from process monitoring, the *fuzzyTECH* Pattern Generator, or real-time traces.

Connection, Monitor, Online/RTRCD

The *fuzzyTECH* Online Edition and MCU Editions equipped with the RTRCD Module support the generation of C code that can be modified remotely from the development PC while running on the target hardware. The development PC is connected to the target hardware (process controller, PLC, microcontroller board) by a serial cable, a field bus, a common file system or a similar communication medium. Initiating a “Connection” with *fuzzyTECH* establishes communication with the target hardware. “Monitor” lets you visualize the entire information flow of the fuzzy logic system in real time. “Online” lets you modify the running system in addition to allowing visualization of information flow. For microcontroller implementations, the *fuzzyTECH* RTRCD Modules deliver a reduced functionality version of the *fuzzyTECH* Online Edition. The *fuzzyTECH* Demo Edition does not support any of these debug modes.

System Test Using the Interactive Debug Mode

Enable the Interactive Debug Mode either by clicking the respective button on the main toolbar or selecting “Debug\Interactive” from the main menu. Minimize the “Debug: Interactive” window and open a Variable Editor window for each of the three linguistic variables of the crane controller. Open the Spreadsheet Rule Editor window for the rule block. Minimize Project Editor and Variables windows. The main window of *fuzzyTECH* should then look similar to Figure 1–19.

If you drag the small red arrows below the plot area of the membership functions in the Variable Editors for “Angle” and “Distance”, a thin vertical line shows the fuzzification. The firing degree of each rule is shown by small black bars on both sides of the DoS values in the Spreadsheet Rule Editor. Click on the bar to see the exact values. The defuzzification process is shown in the Variable Editor for “Power”.

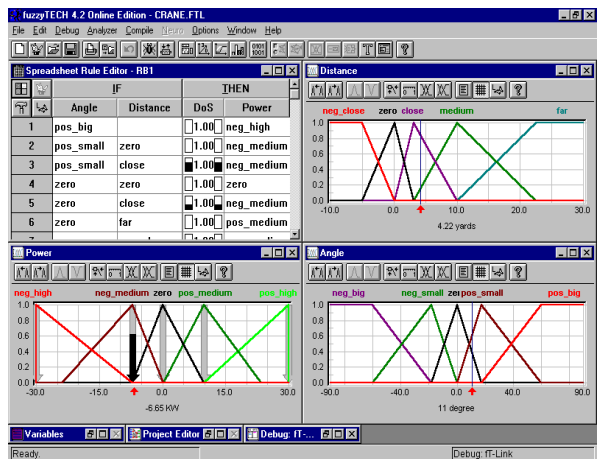


Figure 1–19: In Interactive Debug Mode, all editors graphically visualize the inference.

Set the input variables to the starting position of the crane: “Angle” to 0 and “Distance” to 22. In the Spreadsheet Rule Editor, you see that only the eighth rule in Figure 1–15 fires. This rule states that in the starting position, the crane should start with medium power. The defuzzification shown in the Variable Editor for “Power” is unequivocal, and a crane connected to this controller would now start with 10 kilowatts power.

The situation changes after a short time. As the crane starts to move, the load of the crane sways behind and the distance declines. Set “Angle” to -10 and “Distance” to 20. This is the stable situation when the crane motor should be powered up. This situation is represented by the ninth rule that now fires completely. In a similar fashion, you can now test the fuzzy logic system performance for arbitrary situations. Change the “then-part” of the ninth rule from “pos_high” to “neg_high”, and you instantly see the effects of this in the Variable Editor for “Power”. Similarly, any change of a membership function or a rule weight is instantly reflected in all editors.

Using Analyzers in Interactive Debug Mode

Open the 3D Plot by selecting “Analyzer\3D Plot...” and arrange windows similar to Figure 1–20. By dragging the red arrows in the 3D Plot, you may select any controller operating point. All Editors are updated to show fuzzification, rule inference, and defuzzification for exactly this operating point. Likewise, any change of input variable values in the Variable Editors for “Angle” and “Distance” is reflected instantly in the 3D Plot. By enabling the Tracing option in the 3D Plot window, any operating points tested are shown in a trace in the plot.

Any modification to the system is instantly visible. Change the “then-part” of the ninth rule from “neg_high” back to “pos_high” and back again. The 3D Plot instantly shows the effect on the control surface (Figure 1–20).

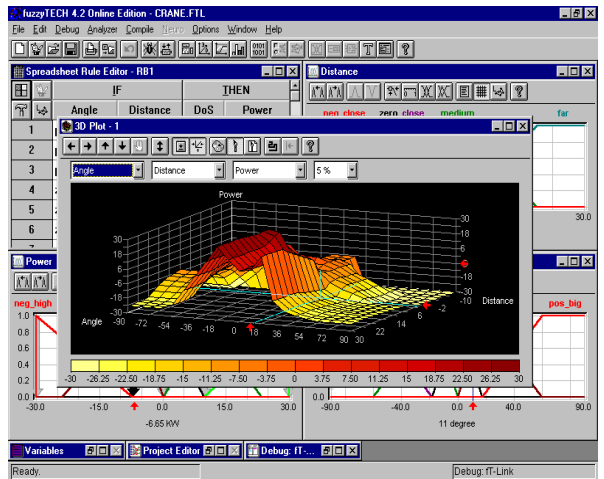



Figure 1–20: Modifications in the system become instantly visible in all editors and analyzers. If you change the “then-part” of rule 2 from “pos_high” to “neg_high”, the front left part of the transfer surface flips down.

In-depth Analysis of the Crane Controller

Start the container crane simulation again by double-clicking on the crane icon in the *fuzzyTECH* program group of the Program Manager. Start the crane by pressing the [Fuzzy] button. All editors and analyzers of *fuzzyTECH* show fuzzification, rule inference, and defuzzification while the crane is running. Does your crane operation look strange now? Then you may have left the “then-part” of the second rule at “neg_high”. Just change it back to “pos_high” while the crane operates.

Statistics Analyzer

You may enable the Statistics Analyzer by selecting “Analyzer\Statistics”, or by clicking  to further examine the rule base. In contrast to Time Plot, Transfer Plot, and 3D Plot, the Statistics Analyzer does not occupy its own windows. Rather, it adds a new column in each Spreadsheet Editor. You can turn Statistics on and off by selecting “Analyzer\Statistics” or by clicking the tool bar button.

The Statistics column in the Spreadsheet Rule Editor is located at the right side of the window. You may have to expand the window to the right to make the statistics column visible. For each rule, two bars show the minimum and maximum firing degrees. The number between the bars counts how often the rule fired with a strength greater than zero. Pressing the [Statistics] button in the Spreadsheet Rule Editor activates a configuration dialog. In this dialog, you can also switch the counter to relative, plus the number of rule firings is displayed in percentages. Pressing the [min-#-max] button resets the statistics.

Transfer Plot Analyzer

An analyzer similar to the 3D Plot is the Transfer Plot (Figure 1–21). A Transfer Plot is created similar to a 3D Plot by selecting “Analyzer\Transfer Plot”. The Transfer Plot shows a two-dimensional cut of the control surface and optionally the two cross-sections at the current operating point. The Transfer Plot also has a Trace function similar to the 3D Plot. See the Reference Manual and the online help system of *fuzzyTECH* for more details.

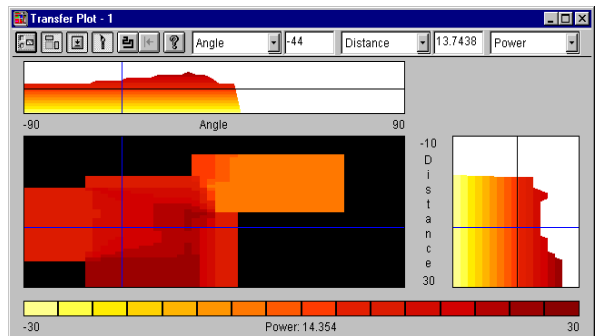


Figure 1–21: The Transfer Plot features a two-dimensional top view of the transfer surface. The two vertical cross section views for the current operation point can also be optionally displayed.

Trace Analyzer

The Trace function provides a real-time trace on the target hardware. Because the trace buffer is kept on the target hardware, the trace is not limited by the bandwidth of the serial communication. Also, by tracing slow processes, the Trace may be used as a time accelerator. The size of the trace buffer depends on the resources on the target hardware and is configured at compilation time. The sample frequency may be changed any time from the PC. You may upload the contents of the trace buffer to the PC for further analysis at any time. *fuzzyTECH*

automatically converts the trace buffer into ASCII format readable by the File Recorder. The Trace function is only supported by the Online Edition or the RTRCD Modules.



Not all modifications to the fuzzy logic system are possible in all Debug modes. For example, you cannot modify the system structure in the Project Editor and you cannot create new terms. However, you may modify any existing membership functions and create new definition points. Also, rules within an existing Rule Block can be added, modified or deleted.

Creating Sample Patterns

fuzzyTECH features a Pattern Generator to create sample patterns that cover the control space. Select “File\Pattern Generator” to activate the Pattern Generator dialog, as shown in Figure 1–22. The list box in the upper part of the window shows the minimum and maximum values for the pattern to be generated, as well as the step width for each input variable. To change any of these values, first select the variable in the list box and then change the values in the fields “From:”, “To:”, and “Step:”. The option, “Margins: On”, ensures that the minimum and maximum values are part of the pattern in all instances. *fuzzyTECH* computes and displays the resulting number of patterns as “Number of Records” and the disk space required to save these patterns as “Disk Space”. Click on the [Generate..] button to create the pattern. *fuzzyTECH* proposes that you accept the file name. The suffix *.PTN is appended to indicate a pattern generated with *fuzzyTECH*. Close the Pattern Generator by pressing the [Close] button. You can view the generated pattern using the “File\View File...” option.

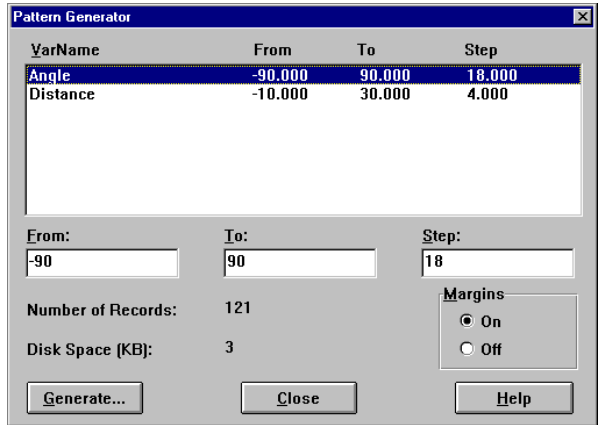


Figure 1–22: The Pattern Generator creates test input data sets for the current fuzzy logic system.

Enable the File Debug Mode in *fuzzyTECH* by selecting “Debug\File Recorder”. This opens the “Read File Control Information From...” dialog. Specify “Pattern Data File (*.PTN)” as the “Filetype” and open the file “CRANE.PTN” that you just created with the Pattern Generator. Clicking the [OK] button opens the File Control dialog (Figure 1–23).

The File Control dialog lets you navigate through the pattern data. The File Control dialog is always at the top of all open windows to allow constant access to its controls. The group “File Information”, at the top of this dialog, shows information about the sample data file and the current record. Open a 3D Plot, activate Tracing in the 3D Plot, and open a Time Plot containing all input and output variables. Press the [>>] button in the File Control dialog. *fuzzyTECH* now processes all patterns in the files from the first record to the last. If you change the direction in which you browse through the records in the file, a vertical red line is plotted in the Time Plot. If you

change anything in the fuzzy logic system, such as membership functions or rules, a vertical yellow line indicates the change in the Time Plot.



Figure 1–23: The File Debug Mode lets you navigate through sample data files.

The [$>$] and [$<$] buttons step one pattern forward or back. The [ν] button is the stop button. The outer buttons move the File Control to the first and last record of the file respectively. You may use the scroll bar to directly address a record. The current record name is shown as “Record:” in cases where a record name is specified in the file. For example, real-time traces that are uploaded from a target hardware always store a time stamp as the record name. The [Read File...] button lets you select a different sample data file.

If you want to get the results of the fuzzy logic inference on file as well, you may use the Batch Mode. The Batch Mode does not display the fuzzy logic inference in all editors, but generates an output file that contains the computed output values in addition to the input values. Activate the Batch Mode by “Debug\Batch”, specify the input data file “CRANE.PTN” and click [OK]. *fuzzyTECH* suggests “CRANE.OUT” as the name of the output file. Accept this name by clicking [OK]. You can view the result file “CRANE.OUT” by “File\View File...”.

The FTL Format

fuzzyTECH uses the hardware-independent FTL format (Fuzzy Technology Language) to store fuzzy logic systems. The Reference Manual contains a detailed description of FTL. If you manually edit an FTL file that was generated by *fuzzyTECH*, make sure to erase the line containing “SHELL = DEMO;”. This forces a complete consistency check when opening the FTL file.

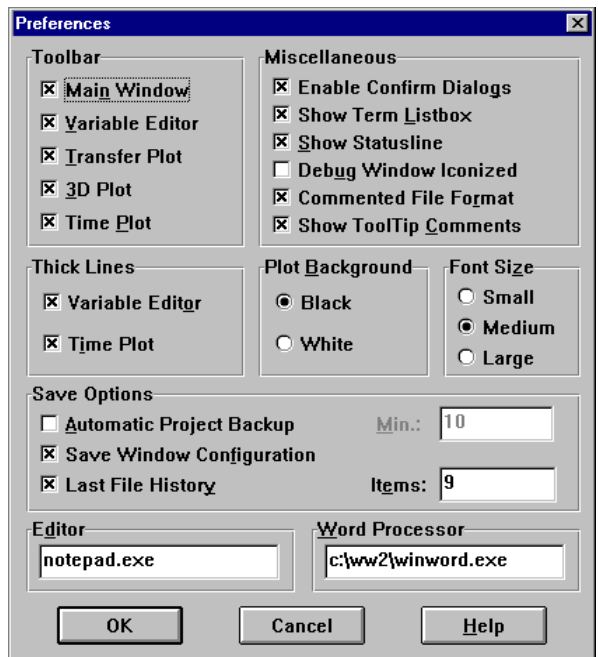


Figure 1–24: The Preferences Dialog lets you customize the user interface of *fuzzyTECH*.

Customization of the User Interface

The user interface of *fuzzyTECH* can be customized in many ways. You may, for instance, turn off individual toolbars or the status bar to save screen space on low resolution monitors. Open the “Preferences” dialog box (Figure 1–24) by selecting “Options\Preferences...” from the main menu.

Toolbar

For each window type, you can turn the toolbar on or off.

Miscellaneous:

Enable Confirm Dialogs

When enabled, this option ensures that a confirmation dialog is shown any time you delete a component of the fuzzy logic system. Experienced designers should disable this option because the UNDO function of *fuzzyTECH* reverses accidental deletions.

Show Statusline

Lets you turn the status bar on and off.

Debug Window Minimized

The Debug window is active in any debug mode. It always displays the crisp values of all input and output variables of the system. Enabling this option always minimizes the Debug window any time you start the Debug mode. If you have low video resolution, you can use this option to save screen space and computing time.

Commented File Format

Uses the extended file format containing a three row header and identification strings in each data record.

Show ToolTip

Comments

Shows the written comments of every object in the form of a tool-tip, when the mouse is placed directly on the object.

Thick Lines

Use this option for high resolution monitors to increase the line thickness in Variable Editors and the Time Plot.

Plot Background

Use this option for screen shots of the Time Plot and the 3D Plot to turn the black background color to white.

Font Size

Depending on the screen size, *fuzzyTECH* has pre-set the font size for many windows. You can change this using the Font Size option.

Save Options:

Automatic Project Backup

fuzzyTECH saves the current project in a separate BAK directory at regular intervals as specified by the time period .

Save Window Configuration

By enabling this option, *fuzzyTECH* saves all window positions and all analyzer configurations with the project. A detailed description follows.

Last File History

By activating this option, the number entered here is the number of the most recently used files that are shown at the end of the “Files” entry of the main menu.

Editor

Lets you specify a preferred editor to view data and project files.

Word Processor

Lets you specify the word processor for documentation.

Saving of Window and Analyzer Configuration

If the “Save Window Configuration” option in the Preferences dialog is enabled, *fuzzyTECH* saves the positions and sizes of all windows, as well as the configuration of all open analyzers, in a *.CFG file whenever you save the *.FTL file. If you open an FTL file, *fuzzyTECH* searches for a *.CFG file with the same project name (the project name consists of the first eight characters of the file name). If this *.CFG file exists, it is opened and all windows and analyzers (once you activate a debug mode) are restored in size and configuration. You may open the *.CFG file with an ASCII editor. If no *.CFG file with the same project name is found in the same directory, all windows open in the default size and you have to configure analyzers manually.

The functions “Save As...” and “Open...” also let you save and open *.CFG files under a different project name. This allows you to have multiple configurations for the same FTL project. In order to reset the configuration to *fuzzyTECH* defaults, just erase the *.CFG with the same project name as the *.FTL file.

Create a Fuzzy System from Scratch: The Fuzzy Design Wizard

The Fuzzy Design Wizard (FDW) supports quick generation of a system prototype. For inexperienced designers, the FDW provides a step-by-step guide through all fundamental design steps. Experienced designers can design a system prototype in just minutes.

Dialog Sequence

The FDW activates a sequence of dialogs that guide you through the creation of a complete fuzzy logic prototype. The FDW can also be used to create a new component for an existing fuzzy project. The sequence of dialogs is as follows:

- Welcome Dialog

- Project Size Dialog

- Linguistic Variable Dialogs for each defined variable

- Defuzzification Dialogs for each defined output variable

- Rule Configuration Dialog

Each window contains a few of the several required design steps. For each step, reasonable defaults are displayed when entering the dialog. The defaults can be accepted or changed.

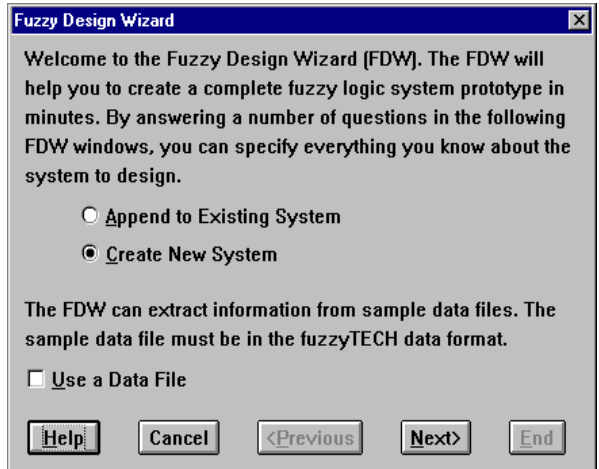



Figure 1–25: In the first Fuzzy Design Wizard window, you specify whether the new components are appended to an existing system or define a new system. By specifying a sample data file, the FDW automatically extracts design data.

With these features, the FDW is well suited to create the “empty” fuzzy logic system that NeuroFuzzy training requires. Call the FDW by either selecting “File\Fuzzy Design Wizard...” in the main menu or by clicking the respective button  in the main toolbar. In the first FDW window (Figure 1–25), specify:

- Whether the fuzzy logic system to be created shall be added to the currently opened fuzzy logic system or whether you want to create a new system.
- Whether a sample data file exists that can be analyzed by the FDW. The FDW uses information such as variable names, as well as the interval and the distribution of the values, to propose default variables in later FDW windows.

Control Buttons

The [Help], [Cancel], [Previous], [Next] and [End] buttons are located at the bottom of each FDW window. The [Next] button always takes you to the next window. The [Previous] button always brings you back to the previous window. By stepping back and forth in the FDW windows, you can reverse any design decision you make.

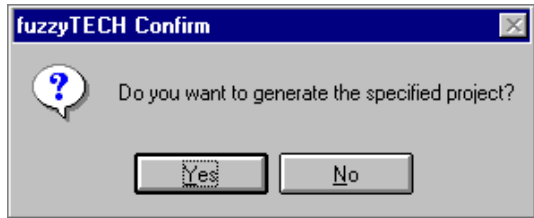


Figure 1–26: Before the Fuzzy Design Wizard generates the system, this confirmation dialog is displayed.

The [End] button forces the FDW to generate the fuzzy logic system with the current specifications. Since the FDW proposes default values for every design decision, the [End] button can be pressed in any FDW window. The FDW also ends when the [Next] button is pressed in the last FDW dialog. Then, a confirmation dialog (Figure 1–26) appears before the actual system is generated.

The basic idea of the FDW is to propose default values for all design decisions. This minimizes the input required for generating a fuzzy logic system prototype. If you specify a sample data file, the FDW automatically determines default parameters for many design decisions from the data.



You can overwrite all of the default parameters if desired.

The rest of this section walks you through the generation of a fuzzy XOR system from a sample data file to illustrate the use of the FDW. In the section which follows, the system generated here is trained with the NeuroFuzzy Module.

Enable the check box, "Use a Data File", so that the FDW analyzes the file "XOR.EXP" that contains the training data for the XOR. When you click on the [Next] button or press the [Return] key, the FDW prompts you for a sample data file. Select "XOR.EXP" in the sub-directory "..\SAMPLES\XOR". Figure 1-27 shows the next FDW window where you specify number of input, output, and intermediate variables as well as their number of terms. You may change the number of terms for individual linguistic variables later.

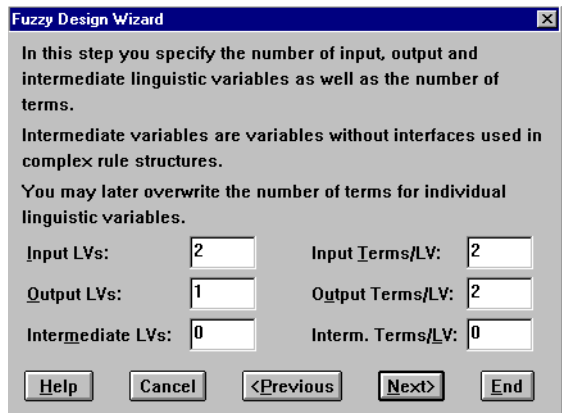


Figure 1-27: In the second FDW window, you specify the number of input, output and intermediate variables and their term numbers.

By analyzing the file “XOR.EXP”, the FDW discovered that the system to be created requires two input and one output variables. Due to the simple structure of the training data, the FDW does not propose intermediate variables. For terms, the FDW proposes three (3) terms for each input variable and five (5) terms for the output variable. This is due to the fact that even simple fuzzy logic systems normally do not have less than three (3) terms for each input variable and five (5) terms for the output variable. However, training a digital function such as an XOR is a non-standard task for a fuzzy logic system. Hence, you should manually overwrite the term numbers to 2 for true and false. Move to the next FDW window by clicking the [Next] button or pressing the [Return] key.

The following three FDW windows specify the three linguistic variables of the system. The first of these FDW windows (Figure 1–28) defines the first input variable, “Input A”. By analyzing the sample data set, the FDW has set the range for the variable from 0 to 1. The FDW has also determined that only 0 and 1 values are specified for all variables and proposes the term names “false” and “true”. You may choose other term names from the drop list box or overwrite the term number. For now, accept the default settings proposed by the FDW and move to the next FDW window.



Figure 1–28: The FDW presents a similar window for each variable to be defined.

The next FDW window defines the next input variable, “Input_B”. Accept the proposed defaults and step to the next FDW window that defines the output variable. For the output variable, the FDW proposes the range -1 to 2. This results from considering the possibility that a later shift of the membership functions out of the range 0 to 1 could be necessary. Accept the default values proposed by the FDW and step to the next FDW window.

This FDW window defines the defuzzification method for the output variable. Accept the proposed default by clicking the [Next] button or pressing the [Return] key. The next FDW window defines the rule blocks (Figure 1–29). Due to the simplicity of the training data, the FDW proposes just one rule block. Enabling the check box, “Create Rule Base”, lets the FDW generate a complete rule set for each rule block. For each combination of terms, a rule is generated. As Degree-of-Support

(DoS), either random values ("Random Value") or a constant value ("User Defined Value") is assigned.

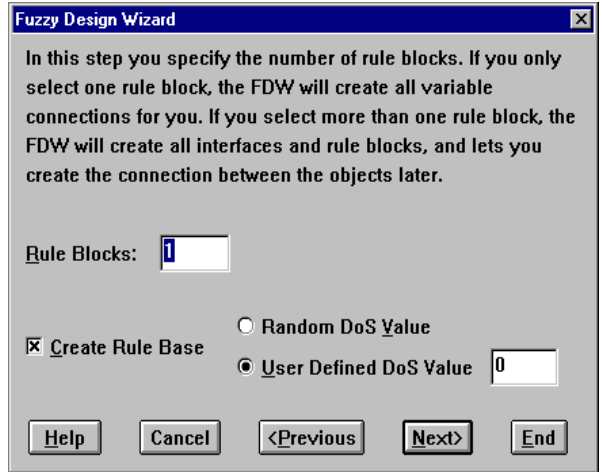


Figure 1–29: The last FDW window specifies whether and how the FDW generates rule blocks.

The FDW proposes the generation of a rule set where all rules have a DoS of 0. Since the DoS is the degree of membership a rule has in the set of totally true rules, this is equivalent to a set of completely false rules. Hence, the generated rule set contains no information. The reason for creating a complete but totally false rule set is that the NeuroFuzzy training can only start with an existing rule set. Accept the proposed default values of the FDW and either click the [Next] button or press the [Return] key. Since this is the final window of the FDW, this is equivalent to pressing the [End] button. After pressing [Next], the Confirmation dialog (Figure 1–26) appears. Confirming the dialog or pressing [End] generates the system as shown in Figure 1–30.

Create a Fuzzy System from Scratch: The Fuzzy Design Wizard

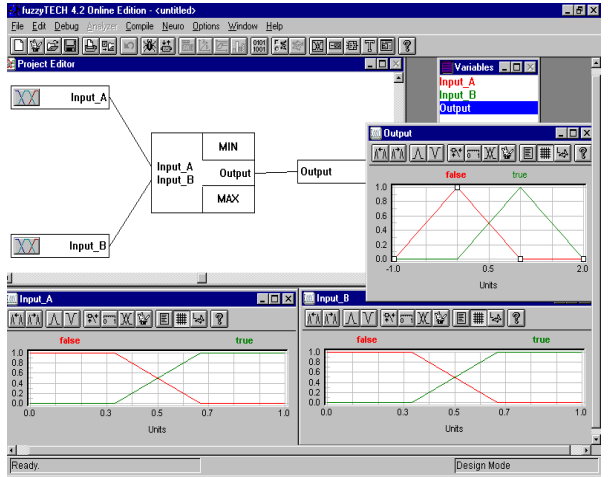


Figure 1–30: Structure of the system generated by the FDW for the file XOR.EXPT

2 Fuzzy Primer

Fuzzy logic is an innovative technology that allows a description of the desired system behavior by using everyday spoken language. Applications range from consumer electronic goods and household appliances to auto electronics, process engineering, and industrial automation solutions. Many successful applications are achieved not by conventional mathematical modeling, but with fuzzy logic and its everyday language. Fuzzy logic “fills” ambiguous descriptors such as *permanent*, *light*, or *above average* with crisp mathematical models so they can be understood by computers.

Boolean Logic

Logic is known as the most precise of all sciences and theoretical disciplines. Most of modern science and mathematics – not to mention the foundation of modern computers – is based upon its principles of precision. Despite the advantages of its accuracy, classical Boolean Logic has a major drawback: it cannot reproduce human thought patterns.

A Natural Logic

Fuzzy logic is a continuous logic patterned after the approximate reasoning of human beings. As a theoretical mathematical discipline, fuzzy logic is designed to react to continuously changing variables and challenge traditional logic by not being restricted to the conventional binary computer values of 0 and 1. Instead, it allows for *partial* and *multi-valued* truths.

This discipline is especially advantageous for problems that cannot be easily represented by mathematical modeling because data is either unavailable, incomplete, or the process is too complex. The real-world language used in fuzzy control allows programmers to incorporate the ambiguous, approximate nature of human logic into computers. The use of linguistic modeling – instead of mathematical modeling – greatly enhances system transparency and modification potential. It leads to quick development cycles, easy programming and accurate control.

Fuzzy Logic

Types of Uncertainty

This section introduces the basic principles of fuzzy logic. Reading this section is essential to understanding how fuzzy logic systems work. Advanced fuzzy logic technologies are treated in the section “Design steps” 3.

Mathematical Principles of Uncertainty

Many mathematical disciplines deal with the description of uncertainties, such as probability theory, information theory, and fuzzy set theory. It is most convenient to classify these by the type of uncertainty they treat. In this section, we consider only two types of uncertainty, stochastic and lexical uncertainty.

Stochastic Uncertainty

Stochastic uncertainty deals with the uncertainty toward the occurrence of a certain event. Consider Statement 1:

Statement 1

The probability of hitting the target is 0.8

The event itself – hitting the target – is well defined. Close shave, no cigar. The uncertainty in this statement is whether the target is hit or not. This uncertainty is quantified by a degree of probability. In the case of Statement 1, the probability is 0.8. Statements like this can be processed and combined with other statements using stochastic methods, such as the Bayesian calculus of conditional probability.

Lexical Uncertainty

A different type of uncertainty lies in human languages, the so-called lexical uncertainty. This type of uncertainty deals with the imprecision that is inherent in most words humans use to evaluate concepts and derive conclusions. Consider words such as “tall men”, “hot days”, or “stable currencies”, for which there are no exact definitions. Whether a man is considered “tall” hinges on many factors. A child has a different concept of a “tall” man than an adult. Also, the context and the background of a person making an evaluation plays a role. Even for one single person, an exact definition on whether a man is considered “tall” does not exist. No law in existence determines the threshold above which a man is conceived as “tall.” This would not make sense anyhow, because a law that defines all men taller than 6' 4" to be “tall” would imply that a man of 6' 3" is not tall at all.

The science that deals with the way humans evaluate concepts and derive decisions is psycholinguistics. It has been proven that humans use words as “subjective categories” to classify qualities such as “height” or “temperature.” Using these subjective categories, elements of the real world are evaluated by the degree to which they satisfy the criteria.

Even though most concepts used are not precisely defined, humans can use them for quite complex evaluations and decisions that are based on many different factors. By using abstraction and by thinking in analogies, a few sentences can describe complex contexts that would be very hard to model with mathematical precision. Consider Statement 2:

Statement 2

We will probably have a successful financial year

At first glance, Statement 2 is very similar to Statement 1. However, there are significant differences. First, the event itself is not clearly defined. For some companies, a successful financial year means that they deferred bankruptcy, for others it means to have surpassed last years profit. For one company, there may be no fixed threshold that exists to define whether a fiscal year is considered to be successful or not. Hence, the concept of a “successful fiscal year” is a subjective category.

Another difference lies in the definition of expressing probability. While in Statement 1, the probability is expressed in a mathematical sense, Statement 2 does not quantify a probability. If someone expresses that a certain type of airplane probably has problems, the actual probability can well be lower than 10 %, still justifying this judgment. If someone expresses that the food in a certain

expensive restaurant is probably good, the actual probability can well be higher than 90 %. Hence, the expression of probability in Statement 2 is a perceived probability rather than a mathematically defined probability as in Statement 1. In Statement 2, the expression of probability is also a subjective category much like “tall men.”

Modeling Linguistic Uncertainty

Statements using subjective categories, such as Statement 2, play a major role in the decision making process of humans. Even though these statements do not have quantitative contents, humans can use them successfully for complex evaluations. In many cases, the uncertainty that lies in the definition of the words we use adds a certain flexibility. Consider for illustration the annual wage increase negotiations between unions and industry. Both want to achieve the same goal: an appropriate wage increase. The problem starts when they have to express in percentage, what they mean by “appropriate”.

The flexibility that lies in the words and statements we employ is used widely in our society. In most western societies, the legal system consists of a certain number of laws that each describe a different situation. For example, one law could express that theft of a car should be punished with 2 years of prison. Another law could define diminished responsibility. For instance, the judge has to decide in a courtcase the exact number of days in prison for a thief. The thief stole a car while under the influence of a 0.1 % blood alcohol level, plus he had a bad childhood and his wife left him the day before. Since for each “real” case a specific law does not exist, the judge must combine all applying laws to derive a fair decision. This is only possible due to the

flexibility in the definition of the words and statements used in each law.

Fuzzy Logic as Human Logic

The basic idea is simple: in reality, you cannot define a rule for each possible case. Exact rules (or laws) that cover a case perfectly can only be defined for a few distinct cases. These rules are discrete points in the continuum of possible cases and humans approximate between them. Hence, for a given situation, humans combine rules that describe similar situations. This approximation is possible due to the flexibility in the definition of the words that constitute the rules. Likewise, abstraction and thinking in analogies are only rendered possible by the flexibility of “human logic”.

A mathematical model is required to implement this human logic in engineering solutions. Fuzzy logic has been developed as such a mathematical model. It allows representation of human decision and evaluation processes in algorithmic form. There are limits to what fuzzy logic can do. The full scope of human thinking, fantasy, and creativity can not be mimicked with fuzzy logic. However, fuzzy logic can derive a solution for a given case out of rules that have been defined for similar cases. So, if you can describe the desired performance of a technical system for certain distinctive cases by rules, fuzzy logic can effectively put this knowledge into a solution.

Fuzzy Logic vs. Probability Theory

Practitioners, especially those working extensively with probability theory, have denied the usefulness of fuzzy logic in applications. They claim that all types of uncertainty can be expressed with probability theory. Rather than embarking on a discussion of whether this is true, consider Example 1. If you find such a statement in a medical textbook and want to implement it in a system, it looks very easy at first glance. Suppose you have a patient that suffers from strong fever, has no yellowish colored skin, but suffers from nausea. You can compute the probability for a hepatitis infection using Bayesian calculus.

Example 1:

“Patients suffering from hepatitis show in 60 % of all cases strong fever, in 45 % of all cases a yellowish colored skin, and in 30 % of all cases nausea.”

Although this looks very easy, the problem starts when you have to define what a “strong fever” is. If you read medical books or ask doctors, you do not get an undisputed, single answer. Even if most doctors agree that if the threshold is at about 102° F (39° C), this does not mean that a patient with temperature of 101.9° F does not have a strong fever at all while another patient with a temperature of 102° F has a strong fever.

If a threshold for “strong fever” does exist, the reverse must also exist. This implies that a very precisely measured body temperature results in a very precise diagnosis. If this was true, you could measure your body temperature up to the fifth significant figure and expect a doctor to tell you, based on this very precise information, the disease from which you suffer. In reality, a doctor does not get a competent diagnosis from the precision of a single parameter, but rather from the evaluation

many symptom parameters. Here, the precision of each parameter does not, for the most part, infer the quality of the result. If the doctor asks you whether you sweat at night, he is not interested in the precise amount but rather a tendency.

As Example 1 illustrates, stochastic uncertainty and linguistic uncertainty are of a different nature. Stochastic uncertainty deals with the uncertainty of whether a certain event will take place and probability theory lets you model this. In contrast, lexical uncertainty deals with the uncertainty of the definition of the event itself. Probability theory cannot be used to model this since the combination of subjective categories in human decision making processes does not follow its axioms.

A “fuzzy” Set

How can you model linguistic uncertainty adequately? If a doctor does not have a precise threshold in mind when evaluating whether a patient suffers from “strong fever,” how does it work? Psycholinguistic research has shown that a doctor would compare the patient with two “prototypes.” On one side exists the “perfect” strong fever patient: pale, sweating, and with chills. On the other side exists the “perfect” patient without a fever, who does not show any signs of fever at all. Comparing with these two extremes, a doctor evaluates where in-between the two his patient ranks.

How can this be modeled mathematically? Consider set theory, where you would first define the set of all patients with strong fever. Then you define a mathematical function that indicates for each patient whether he is a member of this set or not. In conventional math, this indicator function has to uniquely identify each patient as member or non-member of the set. Figure 1 gives an example of the set of “patients with strong fever” (black area), where the indicator function defines “strong fever” as a temperature of higher than 102° F.

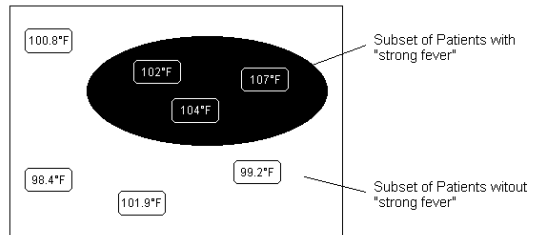


Figure 2–1: In conventional set theory, the set of “patient with strong fever” is defined exactly by $\geq 102^{\circ}$ F.

As pointed out before, a doctor instead evaluates the degree to which his patient matches the prototype of a strong fever patient. Figure 2–2 gives an example of a set where certain elements can also be “more-or-less” members. The “shade of gray” indicates the degree to which the body temperature belongs to the set of strong fever. This “shade of gray” that makes the black area in Figure 2–1 looks as if it is “fuzzy” led to the name “fuzzy logic.”

In Figure 2–2, each body temperature is associated with a certain degree to which it matches the prototype for “strong fever.” This degree is called the “degree of membership” $\mu_{SF}(x)$ of the element $x \in X$ to the set “strong fever” (SF). The body temperature is called a “base variable”, x , with the universe X . The range of μ is from 0 to 1, representing absolutely no membership to the set and complete membership respectively.

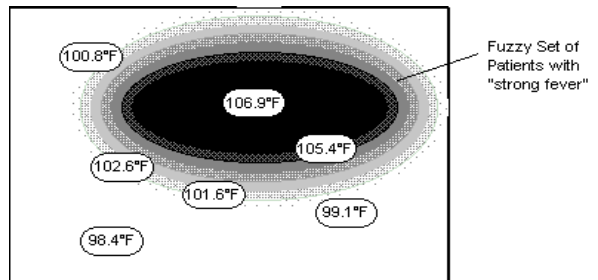


Figure 2–2: The “fuzzy” set of “patient with strong fever” also allows for elements that are “more-or-less” members of the set.

As a temperature of 94° F would have no membership at all, a temperature of 110° F would have complete membership. Temperatures between are members of the set only to a certain degree (Example 2).

Example 2:

$\mu_{SF}(94^\circ F) = 0$	$\mu_{SF}(100^\circ F) = 0.1$	$\mu_{SF}(106^\circ F) = 0.9$
$\mu_{SF}(96^\circ F) = 0$	$\mu_{SF}(102^\circ F) = 0.35$	$\mu_{SF}(108^\circ F) = 1$
$\mu_{SF}(98^\circ F) = 0$	$\mu_{SF}(104^\circ F) = 0.65$	$\mu_{SF}(110^\circ F) = 1$

The degree of membership can also be represented by a continuous function. Figure 3 plots such a membership function.



A temperature of 102°F and a temperature of 101.9°F are evaluated differently, but just slightly and not relative to a crisp threshold. How to define membership functions for a certain application is handled in the section “Design Steps”.

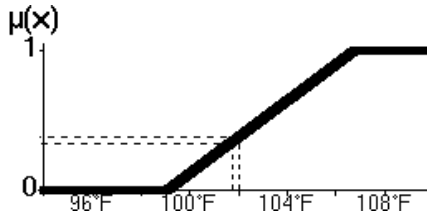


Figure 2–3: The degree, $\mu_{SF}(x)$, to which a temperature x is considered to belong to the set of patients with “strong fever” can be expressed as a continuous function.



Fuzzy sets are a true generalization of conventional sets. The cases $\mu=0$ and $\mu=1$ of the conventional indicator function are just a special cases of the fuzzy set. The use of fuzzy sets defined by membership functions in logical expressions is called “fuzzy logic.” Here, the degree of membership in a set becomes the degree of truth of a statement. For example, the expression, “the patient has a strong fever,” would be true to the degree of 0.65 for a temperature of 104° F.

The primary building block of any fuzzy logic system is the “linguistic variable.” Here, multiple subjective categories describing the same context are combined. In the case of fever, not only strong fever but also raised temperature, normal temperature, and low temperature exist. These are called “linguistic terms” and represent the possible values of a linguistic variable. Figure 2–4 plots the membership functions of all terms of the linguistic variable “fever” into the same graph.

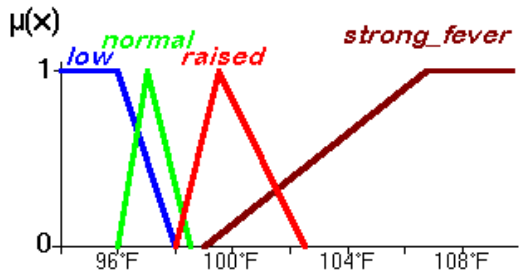


Figure 2–4: A Linguistic Variable translates real values into linguistic values.

This linguistic variable now allows for the translation of a measured body temperature, given in Fahrenheit, into its linguistic description. For example, a body temperature of 100° F would be evaluated as “pretty much raised temperature, and just slightly high fever.” How to use this technology in engineering system design is handled in the next section.

Fuzzy Logic Technologies

In the past 30 years, a great number of methods using fuzzy sets have been developed. This book is restricted to the so-called “rule based” fuzzy logic technologies. Nearly all recent fuzzy logic applications are based on this methodology. This section gives a brief introduction to the basic technology of rule based fuzzy logic systems using a case study in container crane control. A detailed description of the fuzzy logic design methodology follows in the section “Design Steps”.

Case Study: Container Crane Control

Container cranes are used to load and unload containers onto and from ships in most harbors. They pick up single containers with flexible cables that are mounted to the crane head. The crane head moves on a horizontal track. When a container is picked up and the crane head starts to move, the container begins to sway (Figure 2–5). While sway is no problem during transportation, a swaying container cannot be released.

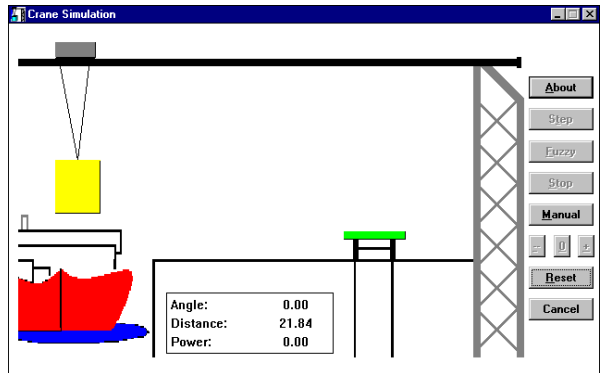


Figure 2–5: Since the container is linked to the crane head with a flexible cable, the container starts to sway when the crane moves. The fuzzy Logic controller compensates for this sway using the human operator’s experience (screen shot of the software simulation).

Two trivial ways to solve this problem exist. One is to position the crane head exactly over the target position, then wait until the sway dampens to an acceptable level. On a non-windy day, this will eventually happen, but it takes far too much time. A container ship needs to be loaded and unloaded in a minimum amount of time for cost reasons. The other trivial option is to pick up the container and move it so slow that no sway ever occurs. Again, this may work on a non-windy day, but it takes far too much time. An alternative is to build container cranes with additional cables to fix the position of the container during operation. Very few cranes make use of this technique due to the much higher cost of the solution.

For these reasons, most container cranes use a continuous speed control for the crane motor under the direction of a human operator. The operator has to simultaneously compensate for the sway while ensuring the target position is reached in a minimum amount of time. This task is not easy, but a skilled operator is capable of achieving acceptable results.

Control Model Alternatives

Many engineers have tried in the past to automate this control task. They have attempted:

- Linear PID control,
- Model-based control, and
- Fuzzy logic control.

Conventional PID (Proportional-Integral-Differential) control was not successful because the control task is non-linear. Sway minimization is important only when the container is close to the target. Other engineers have tried to derive a mathematical model of the crane to use in a model-based controller. This results in a fifth-degree differential equation to describe the mechanical behavior. Although in theory this should work, it does not. The reasons for this are:

- The crane motor behavior is by far not as linear as assumed in the model,
- The crane head moves with friction, and
- Disturbances such as wind gusts cannot be included in the model.

Linguistic Control Strategy

On the other hand, a human operator is capable of controlling a crane without differential equations. The operator does not even use a cable length sensor that any model-based solution would require. Once he has picked up the container, the operator starts the crane with medium motor power to see how the container sways. Depending on the reaction, he adjusts motor power to get the container a little behind the crane head. In this position, maximum speed can be reached with minimum sway. Getting closer to the target position, the operator reduces motor power or even applies negative power to brake. As the crane gets very close and power is further reduced or reversed, the container gets a little ahead of the crane head until the container has almost reached target position. Finally, the motor power is increased so that the crane head is over the target position and sway is zero. No differential equations are required for this operation, and disturbances and non-linearities are compensated by the operator's observance of the container's position.

The analysis of the operator's actions reveals that he uses some "rules of thumb" to describe his control strategy:

Start with medium power.

If you have started and you are still far away from the target, adjust the motor power so that the container gets a little behind the crane head.

If you are closer to the target, reduce speed so the container gets a little ahead of the crane head.

When the container is very close to target position, power up the motor.

When the container is over the target and the sway is zero, stop the motor.

Implementing a Linguistic Control Strategy

Sensors for the crane head position ("Distance") and the angle of the container sway ("Angle") are employed to automate the control of this crane. Using these inputs to describe the current condition of the crane, the five rules of thumb can be translated to an "if-then" format:

IF Distance = far AND Angle = zero THEN Power = pos_medium

IF Distance = far AND Angle = neg_small THEN Power = pos_big

IF Distance = far AND Angle = neg_big THEN Power = pos_medium

IF Distance = medium AND Angle = neg_small THEN Power = pos_medium

IF Distance = close AND Angle = pos_small THEN Power = pos_medium

IF Distance = zero AND Angle = zero THEN Power = zero



Rule 2 has been translated into two rules to fit the if-then format.

If-then rules always describe the reaction to a certain situation as:



IF <situation> THEN <action>

In the case of the container crane, each situation is identified by two conditions. The first condition describes the value of Distance, the second the value of Angle. The conditions are combined by AND, representing the fact that both conditions have to be valid for the respective situation.

Fuzzy Logic

Once you have set up a set of rules describing the desired behavior of a system, the question becomes: how can you implement these rules? First, consider using a programming language to code the “if-then” rules. The problem with this method is that you have to define the words that the conditions of the rules use. However, exact definitions for these words do not exist. This is the same as with the definition of “strong fever” discussed in the previous section. This is the reason you can use fuzzy logic to implement a linguistic control strategy. The following shows you step by step, how to design a controller using fuzzy logic techniques.

Structure of a Fuzzy Logic Crane Controller

Figure 2–6 shows the complete structure of a fuzzy logic controller. First, all sensor signals must be translated into linguistic variables. For example, a measured distance of 12 yards may be translated into the linguistic value “still medium, just slightly far.” This step is called “fuzzification,” as it uses fuzzy sets for translating real variable values into linguistic variable values.

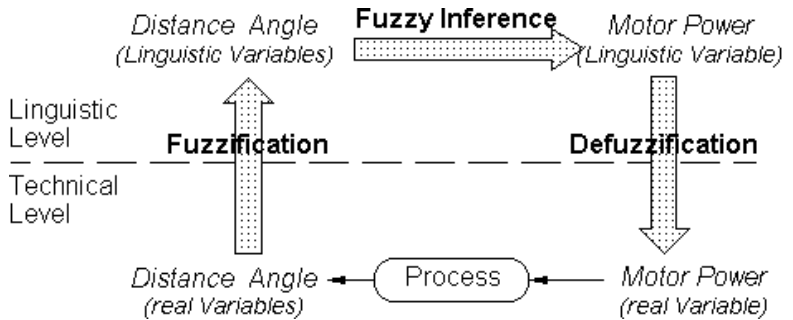


Figure 2–6: Structure of a fuzzy logic controller for the container crane. The fuzzy logic system consists of three steps: Fuzzification, Fuzzy Inference, and Defuzzification.

Once all input variable values are translated into the respective linguistic variable values, the so-called “fuzzy inference” step evaluates the set of if-then rules that define system behavior. The result of this is again a linguistic value for the output linguistic variable. For example, the linguistic result for Power could be “a little less than medium.” The so-called “defuzzification” step translates this linguistic result into a real value that represents the power setting of the motor in kilowatts.

Fuzzification using Linguistic Variables

Linguistic variables have to be defined for all variables used in the if-then rules. As described in the section “Computing Fuzzy Systems”, possible values of a linguistic variable are called terms or labels. For the crane controller, the terms are:

Example 3:

Linguistic Variable	Possible Values (Terms)
Distance	{far, medium, close, zero, too_far}
Angle	{pos_big, pos_small, zero, neg_small, neg_big}
Power	{pos_big, pos_medium, zero, neg_medium, neg_big}

For every linguistic variable, each term is defined by its membership function. Figures 2-7 and 2-8 show the definitions for the two input variables.

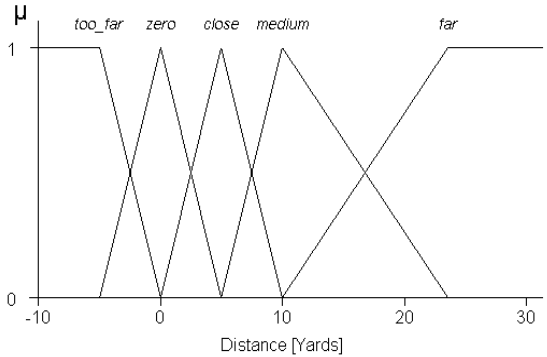


Figure 2-7: Linguistic Variable: “Distance” between crane head and target position

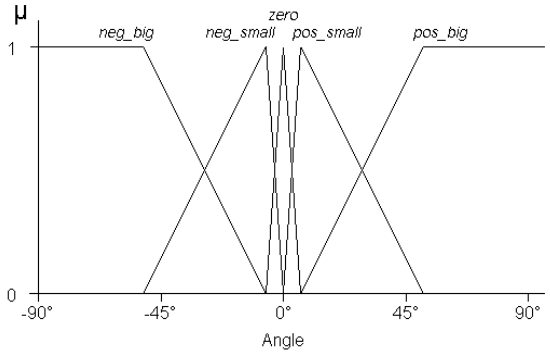


Figure 2–8: Linguistic Variable: “Angle” of the container to the crane head

Consider a possible situation for the crane where the Distance of the crane head to the target position is 12 yards and the Angle of the container is +4°. Example 4 shows how the fuzzification is computed for this case.

Example 4:

A Distance of 12 yards is a member of the fuzzy sets for the terms:

- far to the degree of 0.1
- medium to the degree of 0.9
- close to the degree of 0
- zero to the degree of 0
- too_far to the degree of 0

An Angle of $+4^\circ$ is member of the fuzzy sets for the terms:

neg_big	to the degree of 0
neg_small	to the degree of 0
zero	to the degree of 0.2
pos_small	to the degree of 0.8
pos_big	to the degree of 0

The Distance of 12 yards is translated into the linguistic variable value of {0.1, 0.9, 0, 0, 0} which can be interpreted as “still medium, just slightly far.” The Angle of $+4^\circ$ is translated into the linguistic value of {0, 0, 0.2, 0.8, 0} which can be interpreted as “positive small, somewhat zero.” How to define terms and membership functions is described in detail in the section “Fuzzification”.

Fuzzy-Inference using If-Then Rules

Now that all input variables have been converted to linguistic variable values, the fuzzy inference step can identify the rules that apply to the current situation and compute the value of the output linguistic variable. Example 5 shows a subset of three rules for illustration:

Example 5:

- Rule 1: IF Distance = medium AND Angle = pos_small
THEN Power = pos_medium
- Rule 2: IF Distance = medium AND Angle = zero
THEN Power = zero
- Rule 3: IF Distance = far AND Angle = zero
THEN Power = pos_medium

The computation of the fuzzy inference consists of three components:

Aggregation: computation of the IF part of the rules. This step computes the validity of the rule relative to the conditions.

Composition: computation of the THEN part of the rules. This step computes the degree of truth for the rule.

Result Aggregation: After the degrees of truth for the rules are computed, this step determines which rules will contribute to the defuzzified result

Aggregation

The IF part of Rule 1 combines the two conditions “Distance = medium” and “Angle = pos_small.” The IF part defines whether the rule is valid in the current situation or not. In conventional logic, the combination of the two conditions can be computed by the Boolean AND as shown in the following table:

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

In the case of fuzzy logic, the Boolean AND cannot be used as it cannot cope with conditions that are more-or-less true. Hence, new operators had to be defined for fuzzy logic to represent logical connectives such as AND, OR and NOT. The first set of operators that has been proposed is given in Example 6. These three operators are used in the majority of today’s fuzzy logic applications.

Example 6:

$$\text{AND: } \mu(A \wedge B) = \min\{ \mu(A), \mu(B) \}$$

$$\text{OR } \mu(A \vee B) = \max\{ \mu(A), \mu(B) \}$$

$$\text{NOT: } \mu(\neg A) = 1 - \mu(A)$$

If you use the min operator to represent the logical AND, the IF parts of the rules of Example 5 using values from Example 4 can be computed as shown in Example 7. The results are the degrees of truth of the IF parts and thus indicate how adequate each rule is for the current situation.

Example 7:

$$\text{Rule 1: } \min\{ 0.9; 0.8 \} = 0.8$$

$$\text{Rule 2: } \min\{ 0.9; 0.2 \} = 0.2$$

$$\text{Rule 3: } \min\{ 0.1; 0.2 \} = 0.1$$

Composition

Each rule defines an action to be taken in the THEN part. The degree to which the action is valid is given by the adequacy of the rule to the current situation. This adequacy is computed by the aggregation as the degree of truth of the IF part. Therefore, Rule 1 results dictate the action “Power = pos_medium” to the degree 0.8; Rule 2 dictates the action “Power = zero” to the degree 0.2; and Rule 3 dictates the action “Power = pos_medium” to the degree 0.1.

In many cases the degree of truth of the IF part is considered the degree of truth of the rule, however, in some cases it is advantageous to let the rules themselves be fuzzy. In the composition step, the degree of truth of the IF part of the rule is multiplied by a weighting factor. This factor represents the weight of the rule in relation to the other rules in the system. The use of weights is the most common, simple and transparent implementation of more

general concepts such as Fuzzy Associative Memories or the Compositional Rule of Inference. In this example fuzzy logic inference, a weight of 1.0 is multiplied with the aggregation result in the composition step. In *fuzzyTECH* this weight factor is referred to as the DoS of the rule and may take values in the interval [0, 1].

Result Aggregation

As both Rules 1 and 3 result in the same action but with a different degree of truth, these results must be combined before the defuzzification step. In a fuzzy logic rule base, rules are defined alternatively: either Rule 1 is true, OR Rule 2 is true, OR Rule 3 is true, OR ... Using the fuzzy logic operators as listed in the example, the OR can be represented mathematically by the max operator. The final result of the fuzzy logic inference for the linguistic variable Power is shown in Example 8:

Example 8:

For the linguistic variable Power, the fuzzy inference result is:

pos_big	to the degree of 0.0	
pos_medium	to the degree of 0.8	(= max { 0.8; 0.1 })
zero	to the degree of 0.2	
neg_medium	to the degree of 0.0	
neg_big	to the degree of 0.0	

This fuzzy inference method is sometimes called MAX/MIN or MAX/PROD inference. Advanced inference methods and fuzzy logic operators are discussed in the following section. Experience with the optimization of fuzzy logic systems has shown that it can be necessary to associate weights to each rule. In the section "Design Steps" you find

how if-then rules and weights can be defined for a given application.

Defuzzification using Linguistic Variables

At the end of the fuzzy inference, the result for Power is given as the value of a linguistic variable. In order to use it to set the motor power, it has to be translated into a real value. This step is called defuzzification. The relation between linguistic values and corresponding real values is always given by the membership function definitions. Figure 2-9 plots the membership functions for the linguistic variable “Power”.

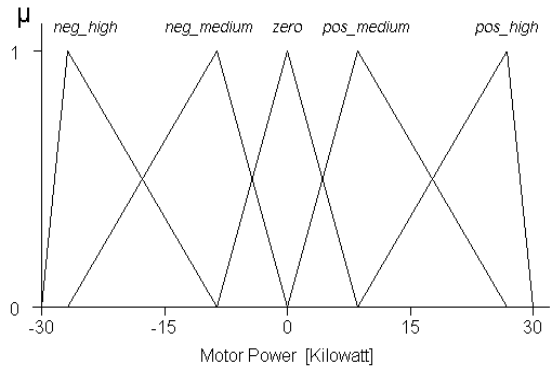


Figure 2-9: Linguistic Variable “Power”.

The result of the fuzzy inference given in Example 8 is both fuzzy and ambiguous since two different actions have non-zero truth degrees. How can two conflicting actions that are defined in fuzzy sets be combined to a “crisp” real-valued output for motor power? Consider how humans solve the problem of combining two fuzzy and conflicting actions in Example 9.

Example 9:

Consider yourself in an apartment house at 11 p.m. You would like to listen to some music such as Mozart or Nirvana, music that requires some volume to be fun. On the other hand, your neighbors have already suffered quite a bit from your recent music sessions. When you set the volume on your stereo, you must combine these two conflicting and fuzzy goals into a crisp value, as only such a value can be set with the volume knob of your stereo.

In order to find a volume that compromises the two goals, you could turn on the music and tune the volume until you balanced out the two goals.

As fuzzy logic mimics the human decision and evaluation process, a good defuzzification method should also approximate this approach. Most defuzzification methods use a two step approach. In the first step, a “typical” value is computed for each term in the linguistic variable. In the second step, the “best compromise” is determined by “balancing” out the results.

“Typical” Values

The most common approach to compute the typical values of each term is to find the maximum of the respective membership function. If the membership function has a maximizing interval, the median of the maximizing set is chosen. For the linguistic variable Power as shown in Figure 2–9, the computation of the typical values is illustrated in Figure 2–10. Here, the gray arrows point to the horizontal position of the typical values.

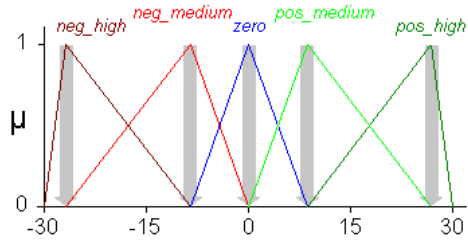


Figure 2-10: In the first step of Defuzzification, the typical value for each term is computed as the maximum of the respective membership function.

Best Compromise

In the second step, the best compromising crisp value for the linguistic result is computed. Figure 2-11 illustrates this step. A “weight” proportional to the degree to which the action is true is placed at the horizontal position of the typical values. The weights are shown as the heights of the black arrows over the gray arrows. The compromise crisp value is then determined by balancing the weights “on a pen tip.” In the example, the position that balances the fuzzy inference result is at the position of 6.4 kilowatts. This value is considered the best compromise and is outputted to the motor.

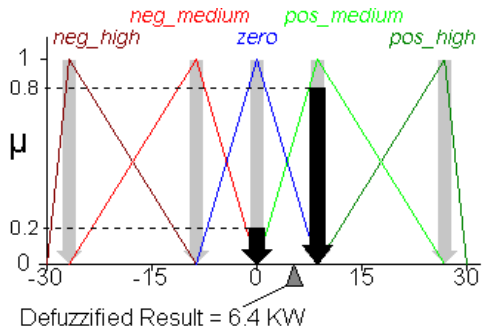


Figure 2-11: By balancing out the conflicting results, a crisp result is found.

This method of defuzzification is called “Center-of-Maximum” and is identical to the “Center-of-Gravity” method using singleton membership functions. These defuzzification methods are used in most fuzzy logic implementations. Other defuzzification methods are introduced and compared in the section “Defuzzification”.

Computing Fuzzy Systems

Fuzzy logic recognizes the advantages of approximate logic. In most real-world situations, a precise answer does not necessarily provide the optimal solution. For example, for the expression $10/3$, the answer *about 3* often suffices. Computing a more precise answer, 3.33, may in some cases be both superfluous and time-consuming. Fuzzy logic expands the strict true/false classifications of traditional logic and incorporates values such as *mostly true* and *totally wrong*. It accommodates the anomalies of human thought patterns by allowing a situation to be *more or less true* in one case and not in another.

Linguistic Concepts

Fuzzy logic systems are also rule-based but they use different concepts to represent the linguistic elements of the rules. Technical quantities, represented by linguistic variables, allow expressions to be *more or less true* in one or even multiple sets. Linguistic rules are formed using operators that represent a linguistic AND and OR. Finally, a computation of the applicability of the rules themselves – represented by a linguistic IF...THEN expression – is performed. This step is called fuzzy inference. In control applications, specialized fuzzification and defuzzification methods link the entire system to the process.

Fuzzy Control

In control applications, fuzzy logic is used to devise a control strategy using every day spoken language. The goal of any control strategy is to obtain a desired output, like crane motor power, from given inputs such as crane position or load angle. Because cranes cannot interpret linguistic concepts, two-way translations between crisp values and linguistic concepts are necessary. Thus, a fuzzy logic process controller is created in three steps:

Fuzzification

Crisp input values are translated into linguistic concepts, which are represented by fuzzy sets. These concepts are called linguistic variables. Degrees of membership for all input values are assigned.

Example:

To what extent is 60 Yards considered far?

Fuzzy Rule Inference

IF...THEN rules that define the relationship between the linguistic variables are defined. These rules determine the course of action the controller must follow.

Example:

If "Distance" is *far* THEN "Power" is *neg_big*.

Defuzzification

The result of the fuzzy inference is retranslated from a linguistic concept to a crisp output value.

Example:

"Power" *pos_big* equals a physical value of 8 W.

The different options to compute these 3 steps are explained in the following sections.

Fuzzification

Linguistic Variable

In fuzzy logic, the different values for a given linguistic variable represent concepts, not numbers. Linguistic values (also called *terms* or *labels*) associated with a linguistic variable “Distance” could be *far*, *medium*, *close*, *zero* or *neg_close*. Each linguistic term is represented by a specific fuzzy set; even approximate descriptions like *extremely close* and *very far* are possible.

Linguistic Terms

A technical quantity like “Distance” is measured as a crisp value, like 20 yards. The degree to which the crisp value belongs to a set is represented by a value between 0 and 1. This value is called the degree of membership. Degree of membership equal to zero means that a value definitely does not belong to a set, while a degree of membership equal to 1 reflects absolute membership. Degree of membership values between 0 and 1 represent partial membership, thus allowing for computations that are *partly true* and *nearly false* – a task which is impossible using conventional logic. In fuzzy logic, a degree of membership is usually normalized in the interval $[0; 1]$ and denoted with the symbol μ .

Membership Function

The degree to which crisp values belong to a given fuzzy set is represented by a function known as a membership function (MBF). Typically, technical quantities are represented on the horizontal axis of the function and membership degrees on the vertical. The technical quantity is called the base variable and represents the universe of discourse.

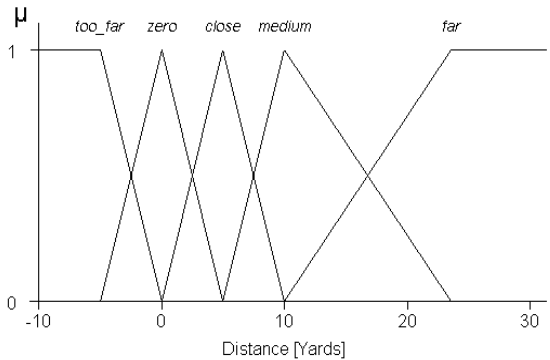


Figure 2-12: Membership functions for “Distance”.

Standard MBFs

Although scientific publications have suggested many different types of membership functions for fuzzy logic, standard membership functions are used in most practical applications. As illustrated in Figure 2-13, standard membership function types are Z, Lambda, Pi and S.

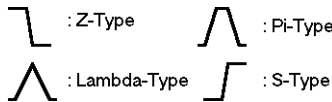


Figure 2-13: Standard membership function types.

All of these can be mathematically represented as piecewise linear functions. The membership degree of a base variable value as given by its piecewise linear membership function is computed as follows:

The membership function is defined as a set of points $P(x_i, \mu_i)$ with:
 $i = 1 \dots n$, $x_1 \leq x_i \leq x_n$ and $\mu_i \in [0, 1]$.
 $[x_1 \dots x_n]$ is the base variable range.

The current operational value is $X \in [x_1 \dots x_n]$
 Evaluate the valid base variable interval with:

$$x_k \leq X \leq x_{k+1}$$

Compute the membership degree by:

$$\mu = \mu_i + (X - x_i) * (\mu_{i+1} - \mu_i) / (x_{i+1} - x_i)$$

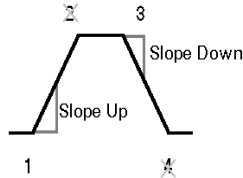


Figure 2-14: Standard membership function definition with two points and two slope values.

The method uses only two definition points and two fixed integer slope values to reduce code size and computing effort (Figure 2-14). The method computes a membership degree by:

The membership function is defined with the points:

Point 1 at $\mu = 0$: x_1 ,

Slope Up: s_1 ,

Point 3 at $\mu = 1$: x_2 ,

Slope Down: s_2 ,

with: x_i, s_i : integer.

The current operation value is X , integer.

Membership degrees are noted as integer in $0 \dots I_{max}$.

Evaluate the valid case with:

case1: $X \leq x_1$

case2: $x_1 \leq X \leq x_2$

case3: $x_2 \leq X$

Compute the membership degree for:

case 1: $\mu = 0$

case 2: $\mu = \min(l_{\max}, (X - x_1) * s_1)$

case 3: $\mu = \max(0, l_{\max} - (X - x_2) * s_2)$



If Fast Computation of MBF is enabled, the Linguistic Variable editor in *fuzzyTECH* automatically modifies the MBF to a feasible definition by moving any defined membership function point to the closest appropriate value. Definition points 1 and 3 can be positioned as desired – restricted only by the resolution of the base variable. However, definition points 2 and 4 can only be positioned so that the resulting slope can be represented by integer numbers.

S-Shaped MBFs

Scientific studies based on psycholinguistic research about the human classification of continuous variables propose an alternative to these four standard MBFs. From these studies, four axioms relating to membership functions were derived:

$\mu(x)$ is continuous over X:

No infinitesimally small change of the base variable can result in a step in the membership degree.

$d(\mu(x)) / dx$ is continuous over X:

No infinitesimally small change of the base variable can result in a step in the derivative of the membership degree.

$d^2(\mu(x)) / dx^2$ is continuous over X:

No infinitesimally small change of the base variable can result in a step in the second derivative of the membership degree.

$\mu(x)$: $\min_{\mu} (\max_x (d^2(\mu(x)) / dx^2))$:

The maximum of the second derivative over the universe of the base variable has to be minimal for the definition of $\mu(x)$.

μ is the degree of membership; $\mu(x)$ the membership function; X the universe of the base variable, and x is an element of X .

Research has proven that only one family of functions satisfies all four axioms: the interpolative cubic spline function, or S-shaped MBF. In this case, only the intervals of the base variable with $\mu(x)=0$ and $\mu(x)=1$ need to be defined. S-shaped MBFs should be designed similarly to standard MBFs only the interpolation is S-shaped, rather than linear. In *fuzzyTECH*, S-shape membership functions permit the specification of an asymmetry parameter, the “shape” parameter. This allows for context-sensitive adaptation. Figure \15 shows an example of a symmetrical S-shape membership function in *fuzzyTECH*. Note, S-shape membership functions require more run-time code and computation time when they are not stored as look-up tables. S-shape membership functions are not supported by all *fuzzyTECH* Editions.

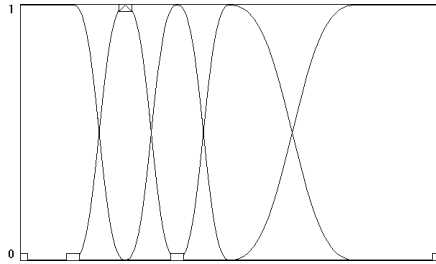


Figure 2-15: S-shaped membership functions (symmetrical).

Arbitrary MBFs

While the majority of applications use standard MBFs (not S-shaped), some membership function derivation methods and adaptation techniques do require more general functions. For the most simple design methodology of these MBF types and for the most efficient run-time code representation, arbitrary membership functions should be approximated as piecewise linear functions.

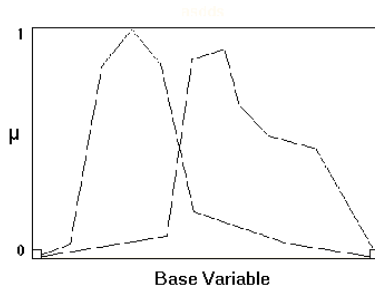


Figure 2-16: Definition of arbitrary MBFs

Handling in fuzzyTECH

All standard MBF types can be represented with up to four (4) definition points. The fuzzification algorithm of the *fuzzyTECH* MCU-Editions uses these four point definitions for all types of Standard MBFs to optimize code size and speed. The *fuzzyTECH* Precompiler and Online Edition also allow for MBF definitions using linear shapes with more than four definition points rather than using different interpolation procedures. In these Editions, the optimizing code generator always picks the most effective algorithm. For example, if you are only using Standard MBFs, the Precompiler and Online Editions use the same algorithm as the MCU Editions.



The Linguistic Variable editor in *fuzzyTECH* enforces the use of feasible shape definitions.

Once the membership degrees of all terms have been evaluated, the controller strategy can be computed within the fuzzy logic inference.

Fuzzy Rule Inference

Most fuzzy logic based systems use production rules to represent the relationships among the linguistic variables and to derive actions from sensor inputs. Production rules consist of a precondition (IF-part and a consequence THEN-part). The IF-part can consist of more than one precondition linked together by linguistic conjunctions like AND and OR.

Fuzzy Rule Inference

The computation of fuzzy rules is called fuzzy rule inference. The inference is a calculus consisting of the steps: aggregation, composition and, if necessary, result aggregation. The first step of the

fuzzy inference, aggregation, determines the degree to which the complete IF-part of the rule is fulfilled. Special fuzzy operators are used to aggregate the degrees of validity of the various preconditions.

Computation of the IF part of a fuzzy rule

- Aggregation of i terms (i condition parts) of linguistic variables.
- Use minimum operator for AND aggregation.
- Use maximum operator for OR aggregation.

$$\text{AND: } \mu_{IF} = \min_i (\mu_i)$$

$$\text{OR: } \mu_{IF} = \max_i (\mu_i)$$

Editions: Precompiler, Online, Business

Compensatory Operators

Using the conjunction AND for the minimum and OR for the maximum is often appropriate in small control applications. Sometimes other kinds of operators are needed to signify the relationship of the different parts of the conditions. While all *fuzzyTECH* Editions support MIN and MAX operators, some editions also support compensatory operator families:

Computation of the IF part of a fuzzy rule

- Aggregation of i terms (i condition parts) of linguistic variables.
- Use one of the three families of operators.
- Use the compensation parameter λ or γ .

$$\text{Min-Max: } \mu_{IF} = \lambda * \min_i (\mu_i) + (1-\lambda) * \max_i (\mu_i)$$

$$\text{Min-Avg: } \mu_{IF} = \lambda * \min_i (\mu_i) + (1-\lambda) * \Sigma (\mu_i) / i$$

$$\text{GAMMA: } \mu_{IF} = \prod_i (\mu_i)^{1-\gamma} * (1-\prod_i (1-\mu_i))^\gamma$$

MAX-MIN Inference

The second calculation step of each production rule (composition) uses the validity of the precondition to calculate the validity of the consequence. In standard MAX-MIN or MAX-PROD (sometimes called MAX-DOT) inference methods, the consequence of a rule is considered equally as true as the condition.

FAM Inference

Using standard MAX-MIN/MAX-PROD methods, rule base optimization often consists of arbitrary rule addition and deletion. This method can result in a clumsy trial-and-error approach, since the individual importance of a rule can be expressed only as a 0 or 1. For this reason, most *fuzzyTECH* Editions support an advanced inference method, the Fuzzy Associative Map inference, or FAM. With FAM, each rule is assigned a Degree of Support representing the individual importance of the rule. Rules themselves can be “fuzzy” – meaning, with a validity between 0 and 1.

The validity of a consequence is calculated by linking the validity of the entire condition with the Degree of Support by a composition operator. Usually, the product operator is used as the composition operator because then the Degree of Support reflects rule “significance.”

Computation of the THEN part of a fuzzy rule:

- Combination of the IF part with the Degree of Support.
- Use the product operator to represent rule significance.

$$\mu_{\text{THEN}} = \mu_{\text{IF}} * \text{DoS}$$

Result Aggregation

Finally, if more than one rule produces the same consequence, an operation must aggregate the results of these rules. A result aggregation step determines the maximum degree of validity for each consequences which is used for all further processing.

BSUM vs. MAX

For the Result Aggregation, you may either select the MAX operator or the BSUM (Bounded SUM) operator. The computation is defined as:

Result aggregation of rules having the same consequence:

- Aggregation of the results of the rules.
- Use the maximum or BSUM operator.

MAX result aggregation:

$$\mu_{\text{RESULT}} = \max_i (\mu_{\text{THEN,RLUE } i})$$

BSUM result aggregation:

$$\mu_{\text{RESULT}} = \min (1, \Sigma (\mu_{\text{THEN,RLUE } i}))$$

Defuzzification

The result produced from the evaluation of fuzzy rules is, of course, fuzzy. In the previous example, the result could be linguistically expressed as “mostly *pos_small* but also “slightly *zero*.” Naturally, a crane motor cannot interpret such a linguistic command. Membership functions are used to retranslate the fuzzy output into a crisp value. This translation is known as defuzzification and can be performed using several methods.

CoM Defuzzification

Because more than one output term can be evaluated as valid, the defuzzification method must compromise between the different results. The Center-of-Maximum Method (CoM) does this by computing a crisp output as a weighted average of the term membership maxima, weighted by the inference results. Figure 2–17 illustrates CoM defuzzification for the linguistic variable “Power.” The locations of the individual term membership maxima are indicated by the gray arrows and the inference result is shown by the height of the black bar in the arrows.

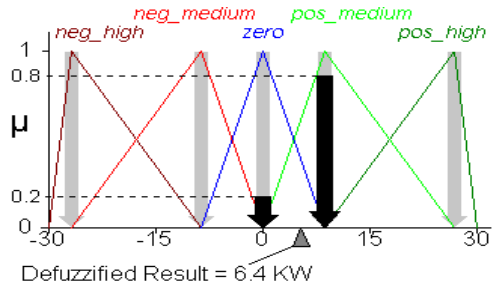


Figure 2–17: Defuzzification with Center-of-Maximum

CoM-Defuzzification:

- Calculates the crisp output value Y .
- Uses Center of Maximum (CoM) method.
- Compromises between the aggregated results of the different terms J of a linguistic output variable
- Based on the Maximum Y_J of the each term J .

$$Y = \frac{\sum_j (\mu_{\text{RESULT, TERM } i} * Y_j)}{\sum_j \mu_{\text{RESULT, TERM } j}} = 6.1 \text{ KW.}$$

MoM Defuzzification

fuzzyTECH supports other defuzzification methods as well. The Mean-of-Maximum Method (MoM), for example, computes a system output only for the term with the highest resulting degree of validity. If the maximum is not unique (like in a Pi-shaped MBF), the mean of the maximizing interval is computed. Figure 2–18 illustrates the MoM defuzzification procedure.

MoM-Defuzzification:

- Calculates the crisp output value Y .
- Uses Mean of Maximum (MoM) method.
- Evaluates the most significant of the different terms J of a linguistic output variable
- Based on the Maximum Y_J of the each term J .
 $Y = Y_J (\mu_{\text{RESULT,TERM,MAX}})$

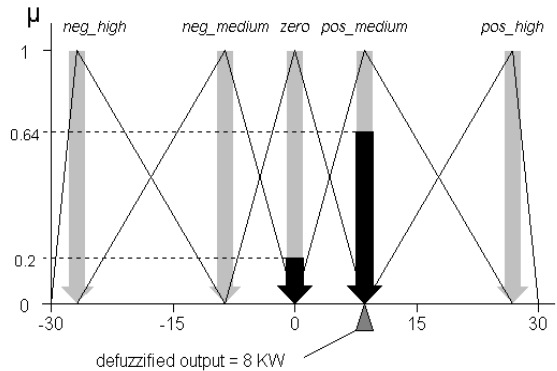


Figure 2–18: Mean-of-Maximum Defuzzification

CoA Defuzzification

Center-of-Area (CoA), sometimes also called Center-of-Gravity (CoG), is the most frequently used defuzzification method in fuzzy systems. With singleton membership functions, CoA and CoM defuzzification are identical. Most CoA implementations are only approximations since they neglect overlapping and can be represented with the CoM defuzzification method.

The CoA defuzzification supported by *fuzzyTECH* is not an approximation as it uses numerical integration for the computation of the areas. Although specification of the number of iterations used for numerical integration is possible, the real CoA defuzzification is much slower than an approximated CoA defuzzification, i.e., CoM, because it is computed during runtime. Figure 2–19 illustrates the CoA defuzzification process.

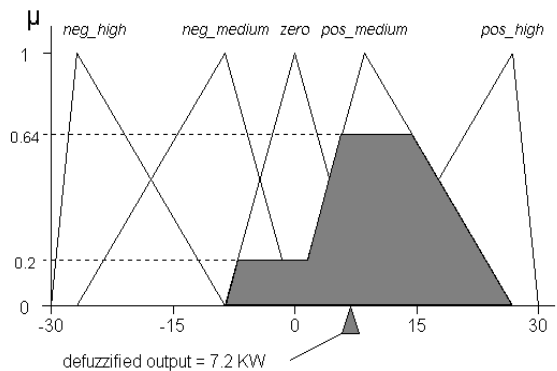


Figure 2–19: Defuzzification with Center-of-Area

Comparisons

While the CoM and CoA defuzzification methods result in the “best compromise solution,” MoM results in the “most plausible solution.” In control applications, CoM is most commonly used because the output value represents the best compromise of all inferred results. MoM is often used in pattern recognition and classification applications as a plausible solution is most appropriate. Scientific literature has suggested many other defuzzification strategies that have rarely been used in industrial applications.

3 Design Steps

This chapter describes the major steps involved in the design of fuzzy logic based systems. First, an overview:

System Definition

The initial system structure is defined by the rule “vocabulary”, which includes the input and output variables with their linguistic terms. In addition to the linguistic variables, a preliminary rule set must also be created for a first system prototype. For complex systems, a structure of rule blocks, interfaces, and intermediate variables can be designed.

Off-line Optimization

Once an initial prototype is set up, the system structure is further refined using the numerous analyzer features for off-line optimization. Interactive debugging, which is used to analyze the system’s reaction to specific situations, can be used to verify system behavior in abnormal situations. Depending on the type of application, either pre-recorded process data or pattern-generator created data sets may be used for optimization. Within off-line optimization, *fuzzyTECH* offers a number of analyzers for system verification. Also, a mathematical simulation of the process or window’s application software can be linked to *fuzzyTECH*.

Online Optimization

Online optimization allows a system to be visualized and modified in real-time on the running process. The control processor (e.g., an embedded controller) is connected to the development PC via a link (e.g., a serial cable). This allows the visualization of the entire inference flow, like in the interactive mode, except that it is done in real-time. Most

system parameters can also be modified in real-time, thus allowing “on-the-fly” system optimization without halting the process. In most closed-loop applications, online optimization features speed up the development cycle by magnitudes. Within online optimization, the analyzers described in the section about off-line optimization can be applied.

Implementation

After the system has been optimized and verified, it can be implemented on various hardware platforms. An overview of the technical alternatives for the use of fuzzy computation routines designed with *fuzzyTECH* is given in the section “Fuzzy Logic Technologie”. For technical details refer to the *fuzzyTECH* Reference Manual.

System Definition

The section “Fuzzy Logic Technologie” presents the basic steps involved in the definition of an initial fuzzy logic system prototype using *fuzzyTECH*: definition of the linguistic variables, set up of the system’s structure, and the formulation of the fuzzy logic rule base.



Read this chapter carefully and follow the “▶” signs in the order of their appearance. This guides you through a fuzzy logic system design from scratch.

Definition of Linguistic Variables

- ▶ Begin the design of your fuzzy logic system from scratch by defining the linguistic variables. (Create at least two (2) linguistic variables to allow one input and one output for your system).

Define a linguistic variable by clicking the right mouse button on the Variables list. This activates

the pop-up menu of this window (Figure 3–1). Otherwise, use [Ctrl][V].

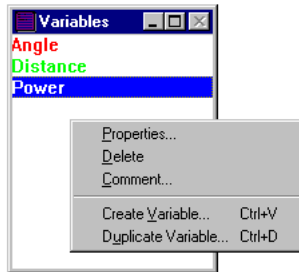


Figure 3–1: Variable window with pop-up menu

Create a new variable with the [Create Variable...] or [Duplicate Variable...] option, or edit an existing variable with the option [Properties...]. In the last case, the appropriate window appears (Figure 3–2).

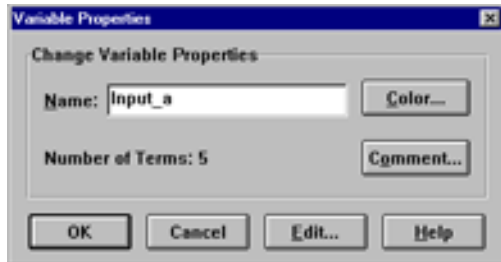


Figure 3–2: Variable properties dialog

Press [Edit...] to open the variable editor window. You may also double-click over the variable's name in the Variables window to open a variable editor window (Figure 3–1).

Define Linguistic Variable

Clicking [Create Variable...] opens the Linguistic Variables Wizard dialog.

Enter a variable name and select the required number of terms; *fuzzyTECH* uses a predefined variable definition to prototype the variable. Pressing [Next>] opens the dialog used to define the membership functions of the variable.

MBF Definition

This dialog lets you visually define the membership functions of the linguistic variable from a holistic view. Use the high shoulder option to generate standard variables that are used in input interfaces, and use the low shoulder option for variables used in output interfaces. A symmetrical or non-symmetrical approach can be appropriately selected according to the respective application.

Press [End] to return to the Variable Editor window from the Linguistic Variables Wizard.

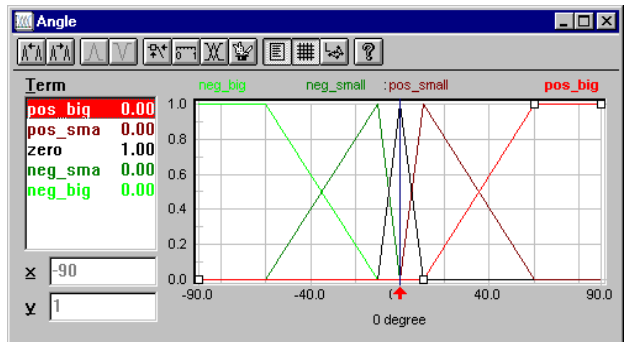



Figure 3–3: Variable Editor for a new variable with popup menu.

The variable editor (Figure 3–4) allows for the graphic definition of each term's membership functions. Clicking the Listbox button  displays a

list of all terms defined for the associated linguistic variable at the left side of the Variable Editor window. On the right side, the linguistic variable's membership functions are plotted. You can activate the variable editor pop-up menu by clicking the right mouse button on the main part of the editor window.

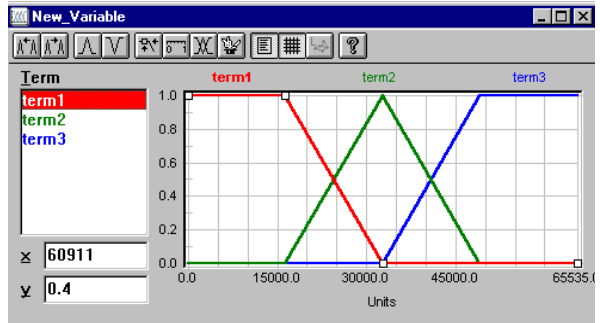



Figure 3-4: Variable Editor for a new variable with toolbar.

- ▶ Define the ranges of discourse for each of your linguistic variables.

Base Variable

Click on the  button in the toolbar to specify the base variable (the crisp variable corresponding to the linguistic variable). This activates the Base Variable dialog box (Figure 3-5). Use this dialog to specify the range and name of the base variable. The name of the base variable is used for display purposes only; you may also enter the units of the base variable here. Also displayed are the data types selected in the Global Options dialog box and the fuzzification / defuzzification method selected in the Interface Options dialog box. For the variable range, two different representations exist: the Shell representation and the Code representation. The Shell representation is used for all editors and debugging tools during development, while the

Code representation is used in the generated code (assembly, C, ...). In all *fuzzyTECH* Editions that support floating point resolution for fuzzy logic inference, the Shell representation may alternatively be chosen for the generated code.

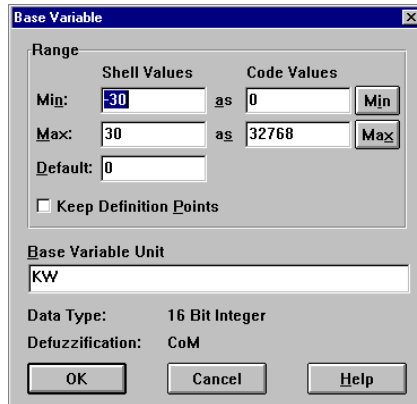


Figure 3-5: Base variable dialog box

Shell Values vs. Code Values

Take the input variable “Angle” of the crane controller for example. The range of this linguistic variable is the interval from -90 to $+90$ degrees. However, in hardware implementation, the angle of the load is measured by an incremental sensor with an output in the interval from 240 to 1400. The necessary conversion can be automatically made through *fuzzyTECH* by entering the range $[-90^\circ, +90^\circ]$ for Shell values and the range $[240, 1400]$ for Code values. The Shell values are used for all editors and analyzers within *fuzzyTECH*, as well as for the ft-Link, File, and Batch debug modes. The Code values are only used to generate the C or assembly code. If you want the Shell and the Code representation to be equal, just enter the same

range. Use the respective buttons to set the Code Values to their minimum or maximum values.

Code Range

The minimum and maximum of the code value interval depend on the data type selected in the Global Options dialog box:


Data typ	Minimum	Maximum
8 Bit Integer	0	255
16 Bit Integer	0	65 535
Double Precision	-1.7E+308	1.7E+308

Default Value

If for any output variable, no rule fires as a result of the fuzzy inference, the default value is used for output. For input variables, this value has no meaning. The default value must be within the range of the Min and Max Shell values.

- Create the terms and design the membership functions for your linguistic variables (create at least one term for each linguistic variable).

Definition of Terms

Begin the term definition by selecting [Create Term...] from the Variable Editor's pop-up menu or click . This calls the term dialog box (Figure 3–6). Here, a term name is specified in the [Term Name] field. The term's position in the sequence of terms is set in the [Position] list box by dragging the current term name to the desired location. Double-click over the term's name in the list box of the variable editor window to activate the Term dialog box after the term is created.

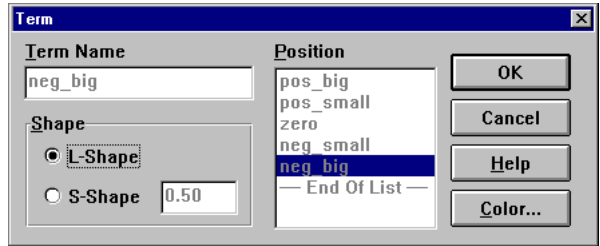


Figure 3–6: Term dialog box

MBF Types

Specify the term’s membership function type with the Shape option. (Refer to the section “Fuzzy Logic” for a comparison between different membership function types.) MCU Editions support only L-Shaped Linear Shape) membership functions of Standard MBF type.

In order to help you visually differentiate among the terms, assign them different colors using the [Color...] option shown in Figure 3–7. Use the scroll bars to adjust the color for the term’s membership plot. *fuzzyTECH* supports a maximum color resolution of 8-bit.



Not all colors can be displayed on certain video adapters.

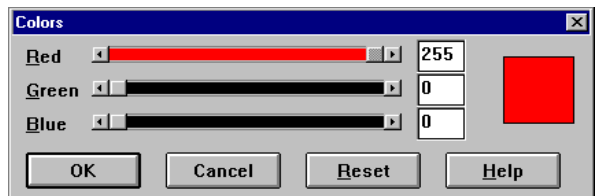


Figure 3–7: Color dialog box

A Standard MBF is easily defined: (See Figure 3–8 for the definition of the term zero)

Step 1

For each term of a given linguistic variable, find the numerical value or range of values that best characterizes it. Since this is a “prototype” value, it should have a membership degree of 1 in the fuzzy set describing the term.

Step 2

Once the values having a membership degree of 1 have been defined, do the same for values having a membership degree of 0. These values are found as the “prototype” of the next higher term and the “prototype” of the next lower term that have been determined in Step 1.

Step 3

After all values with a degree of membership of either 1 or 0 are defined, define the intermediate values. With standard membership functions, use lines to connect the values. Except for the left and right-most linguistic terms, this results in standard Pi- and Lambda-type membership functions.

Step 4

Any value that is less than the lowest-defined value for the leftmost term, or any value that is greater than the highest-defined value for the rightmost term, is considered an absolute member of the respective term’s fuzzy set ($\mu=1$). These are represented by Z- and S-type membership functions.

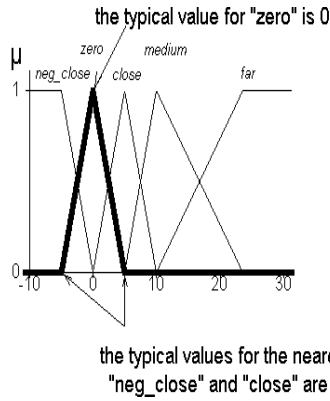


Figure 3–8: Defining a standard MBF for the term “zero”.

In some applications, the value that best characterizes a term is an interval, not a number. For example, any distance within ± 2 yards from the target point could be considered zero and would be best represented as a Pi-shaped MBF. Figure 3–9 shows an example of this.

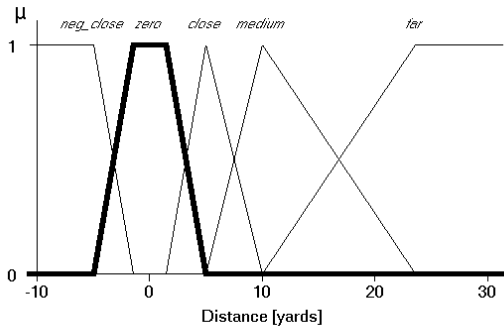


Figure 3–9: Definition of a Pi-type MBF for the term “zero”.

MBF Definition

Return to the Linguistic Variable editor to define a membership function. For each new term, a default membership function (Lambda Type) is created. Standard MBFs are defined by specifying their definition points. Create a definition point for the selected term by double-clicking on an empty area in the function region. A term's definition points are drawn as small rectangles: a highlighted rectangle indicates a selected point, and an empty rectangle shows that the point has not been selected. A selected point is deleted with the [Del] key. In order to move a point, click on it and drag it while keeping the mouse button pressed. An activated point's coordinates are shown in the lower left part of the window. Point coordinates can also be directly entered by entering the value followed by pressing the [=] key. For a variable with multiple terms, the membership functions of all terms are displayed. In order to select a term, click on the term name in the list box or above the plot area (Figure 3–4). The name of the selected term is also displayed in bold characters over the membership function diagram and its definition points are indicated by squares. In order to change term names, colors, sequential position or function types, double-click on the term name in the list box or above the plot area to activate the term window. For editions, that only support standard MBF types, a maximum of four (4) points can be defined. Each point can have a degree of membership of either 0 or 1 (refer to the section "Fuzzy Logic" for details on Standard MBF's).

Point Edit

Assistance with MBF definition tuning is provided by allowing multiple point editing. Highlight the most left point and depress the [Ctrl] key. While depressing the key, *fuzzyTECH* lets you highlight additional points. As long as you hold the key, you can move the complete set of selected points. Using the [Shift] key selects a complete range of points. With this feature, sections of the MBF defined by multiple definition points can be dragged.



The left and right most points can only be moved vertically. If the most left or right point is contained in a highlighted set of points, the whole set can only be moved only vertically. This ensures the definition of a term over the complete universe of discourse.

Inverse Terms

If there is a need to define rules with linguistic statements like IF “Distance” = *not far*, use the [Create Inverse Term...] option from the Linguistic Variable editor’s pop-up menu. A new term, μ' , with the inverse properties of the term’s original membership function, μ , is created. The inverse membership function of a term is computed with: $\mu'(x) = 1 - \mu(x)$. *fuzzyTECH* uses a separate term for the negation, rather than computing the negation during runtime, to allow for a higher degree of flexibility during optimization. This is due to the fact that it is usually necessary to optimize the term’s membership functions so that the relation $\mu'(x) = 1 - \mu(x)$ is not valid in the optimized system.

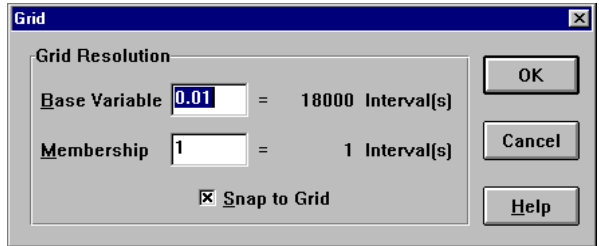



Figure 3–10: Grid dialog box

Grid

From the Variable Editor's pop-up menu, select [Grid...] or click  to define a grid for the membership function definition (Figure 3–10). In order to support standard MBFs, the vertical grid may be set to 1 and [Snap to Grid] to active. Use the base variable grid to divide the base variable range into intervals that allow for an easy positioning of definition points. In the variable editor, the grid can be visualized as a raster.



Grid and raster differ when:

- the Linguistic Variable editor window is too small,
- the grid can not be represented by the raster or
- the internal computing resolution does not match the grid.

Converting to Standard MBFs

In order to ease the design of Standard membership function types, the Linguistic Variable editor also can convert any membership function to a Standard MBF with a fixed overlap of 2 (for every given base variable value, exactly two membership functions are not equal to zero). The conversion to a fixed overlap of 2 may be used to expedite the design of Standard MBFs. Simply define all terms for the linguistic variable for which just the typical values for the term (Section “Fuzzy Logic”) are entered, such as:

- the maximum point of all Lambda-type default membership functions (the inner MBFs of input, all MBF of output variables),
- the right upper point for a Z-type default membership function (the furthest left MBF of the input variables) and
- the upper-left point for an S-type default membership function (the furthest right MBF of input variables).

The Convert to Standard MBFs button in the Variable Editor converts the term definitions to Standard MBFs, as shown in Figure 3–11. You may use the converter function at any time during your design. Figure 3–11 shows an example of a linguistic variable with the terms “NL”, “NS”, “Z”, “PS” and “PL” and the typical values of {-10, -3, 0, 3, 10}.

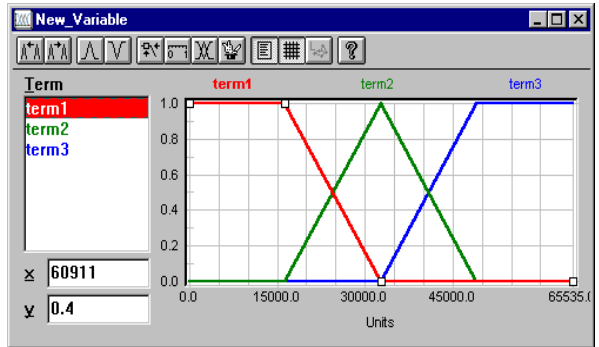


Figure 3–11: Definition of a new variable. The maxima of the membership functions are set to the typical value for each term.

Structure Design

The system structure is displayed graphically in the Project Editor window using three object types:

- Remarks

A remark is a text object used to comment on and describe the system structure. Remarks have no computational influence and can be inserted at any place in the worksheet in different colors and font sizes.
- Interfaces

An interface must be defined for every linguistic variable. The interfaces for input variables of the fuzzy logic system contain the fuzzification procedure, while the interfaces for the output variables contain the defuzzification procedure.
- Rule Blocks

Each rule block contains the rules for a set of variables. The rule blocks of a system contain the entire fuzzy inference step.

Figure 3–12 shows the crane controller example project as it appears in the Project Editor. Information flow between the linguistic variables is indicated by the lines connecting the different objects.

Moving Objects with the Keyboard

Move an object in the Project editor with the keyboard by using the [Tab] key to select the object and the cursor keys to move it. The return key releases the object at the desired position. Activate an object with the [Space] bar. Activate the Project Editor's pop-up menu with [Shift][F10] key.

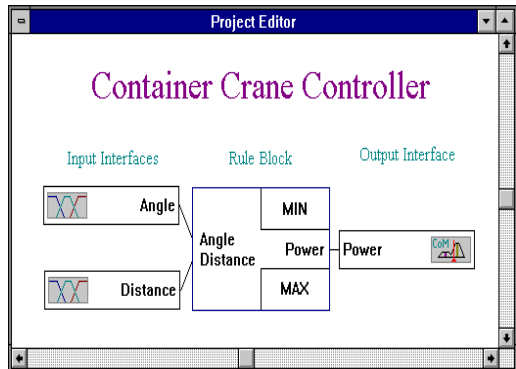


Figure 3–12: Project Editor window with project example

Moving Objects with the Mouse

Objects can also be moved using the mouse. Simply drag and drop objects using the left mouse button. Holding the [Shift] key before dragging the object provides positioning assistance: the direction in which the object was first dragged is fixed. Different cursor shapes indicate the movement mode. Clicking on any object with the left mouse button highlights the respective object. Double-clicking on an object opens an editor window for the object.

Project Editor Pop-up Menu

Clicking the right mouse button anywhere in the Project Editor window activates the following pop-up menu:

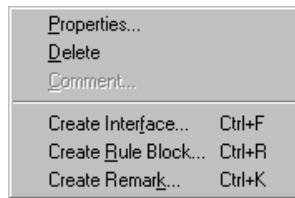


Figure 3–13: Project Editor pop-up menu

This pop-up menu lets you change the Properties of the highlighted object or create new objects.



For expedited system design, the hotkeys [Ctrl][F], [Ctrl][R] and [Ctrl][K] may be used. Double-clicking with the left mouse button activates the Spreadsheet Rule Editor for a rule block, while clicking somewhere in the Project Editor window with the right mouse button activates the pop-up menu shown below if a rule block was highlighted.

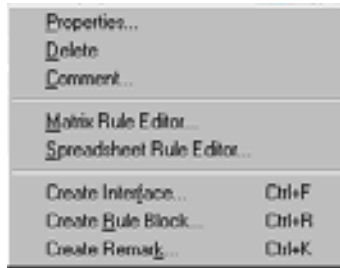


Figure 3-14: Rule Block and Project Editor pop-up menu

Selecting [Properties...] lets you define input and output variables for the respective rule block. Selecting [Spreadsheet Rule Editor...] or [Matrix Rule Editor...] activates the respective editor.

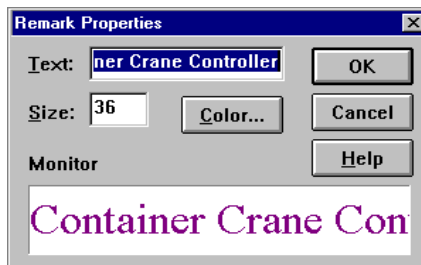


Figure 3-15: Remark Properties dialog box



Create remarks to document your fuzzy logic system.

Remarks

Create a remark by clicking on the main part of the window with the right mouse button and selecting [Create Remark...] from the Project Editor's pop-up menu. Enter the text to appear in the "Text:" field,

plus specify font size and color in the Remark Properties dialog box (Figure 3–15).



Create the interfaces for your linguistic variables (create at least one input and one output interface).

Interfaces

Select [Create Interface...] from the Project Editor's pop-up menu (also see the above paragraph: Project Editor pop-up menu) to create an interface. This activates the Interface Options dialog box (Figure 3–16).

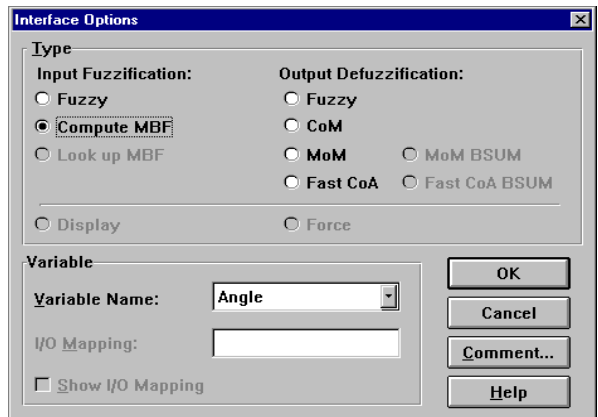


Figure 3–16: Interface Options dialog box

Specify the type of interface in the upper part of the Interface Options dialog box by selecting the fuzzification defuzzification method (a fuzzification method is chosen for a system input and a defuzzification method for a system output). Using Compute MBF for fuzzification and CoM for defuzzification is the standard approach. Compute MBF efficiently computes the membership degrees of the terms of the input variables from crisp input values. CoM is a high performance defuzzification

method, that computes a crisp set value by compromising between multiple firing outputs. Select the variable for the interface in the lower part of the Interface Options dialog box. Each interface can only contain one linguistic variable. For Fuzzy Input Variables and Fuzzy Output Variables, refer to the section “Fuzzy Primer” and the Reference Manual.

Input Interfaces

Input interfaces are displayed in the Project Editor as boxes. In each box, the linguistic variable is printed on the right side and the fuzzification method is shown as an icon on the left side.

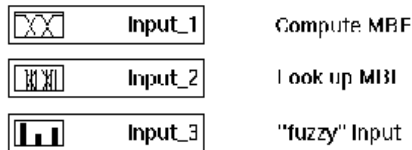


Figure 3–17: Different fuzzification methods

In Figure 3–17, the first interface box uses the “Compute MBF” fuzzification method and the second interface box uses the “Look up MBF” fuzzification method. The later computes fuzzification at compilation time and stores the MBFs as look-up tables in the run-time code. This consumes significantly more memory than the former but yields a faster performance, especially when non-Standard MBFs are used. When no fuzzification is used, a bar graph icon appears. This allows for a “fuzzy” input or customized fuzzification methods.

Output Interfaces

Output interfaces are displayed as boxes, in which the linguistic variable name is printed on the left side and the defuzzification method is shown as an icon on the right side. The first interface box in Figure 3–18 shows the Center-of-Maximum (CoM method, the second shows the Mean-of-Maximum (MoM) method and the third shows the Center-of-Area (CoA) method. The Center-of-Area method is sometimes called Center-of-Gravity CoG). Defuzzification can also be omitted to allow for a “fuzzy” output or to use customized defuzzification methods.





Output_1		Center-of-Maximum
Output_2		Mean-of-Maximum
Output_3		Center-of-Area
Output_4		“fuzzy” output

Figure 3–18: Different defuzzification methods

Rule Blocks

To create a rule block, select the [Create Rule Block...] option of the project editor's popup menu. This invokes the Rule Block Wizard. To configurate an existing rule block, select [Properties...] in the popup menu (Figure 3–20). Specify the variables for the rule block from the [Linguistic Variables] list box using the [>Input>], [>Output>] and [<Remove>] buttons. You may specify more than one variable at the same time to speed up this definition step. If you leave this dialog box with [OK], a rule block icon is created which can be positioned anywhere in the window:



Figure 3–19: Rule Block

The inference method is chosen using the [Aggregation and [Result Aggr. ... buttons. The default inference methods are MIN for aggregation and MAX for result aggregation. Refer to the sections “Computing Fuzzy Systems” and “Formulation of Fuzzy Rules” for details.

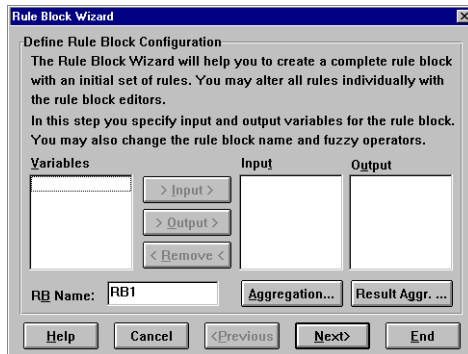


Figure 3–20: Rule Block configuration dialog

Formulation of Fuzzy Rules

Rule Design

When first starting with fuzzy technology, use rules with a Degree of Support of only 0 or 1. If there is a need for individual weighting of rules during optimization, use degrees of support between 0 and 1. At the end of the total inference process, all system output variables are associated with a fuzzy value. The following degrees of validity are assigned to the propelling power to demonstrate this:

Example

The linguistic result for "Power":

<i>neg_big</i>	degree of validity = 0.0
<i>neg_medium</i>	degree of validity = 0.0
<i>zero</i>	degree of validity = 0.2
<i>pos_medium</i>	degree of validity = 0.64
<i>pos_big</i>	degree of validity = 0.0

Rule Definition

Actual system behavior is defined in the individual rules of the fuzzy system. Prototype an appropriate set of rules by first creating rules that represent unambiguous controller strategies at specific operating points. From there, construct your rule set by working backwards step-by-step until the most detailed rule is defined.

Example

The operation point: "If the crane has reached the target point and the load is not oscillating, no power should be applied to the motor"

can be formulated as a fuzzy logic production rule:

IF "Distance" = *zero* AND "Angle" = *zero*
THEN "Power" = *zero*.

Once one unequivocal rule set has been established, more detailed crane behavior can easily be defined. For example, the following rules could be defined if the load is oscillating slightly at the target point:

Example

IF "Distance" = *zero* AND "Angle" = *neg_small*

THEN "Power" = *pos_med*

IF "Distance" = *zero* AND "Angle" = *pos_small*

THEN "Power" = *neg_med*

Utilizing the Matrix rule editor, establish the rules of your fuzzy system with the following steps:

Step 1:

Select the first output variable on the upper horizontal axis of the rule matrix. Select the input variable on the left vertical axis of the rule matrix with the most influence on the system.

Step 2:

Each combination of terms for input variables, which are not selected on the left axis, should be selected in the list boxes. For an input variable on the left axis, find the term that best suits the output variable. Define only those rules with a degree of truth of 0 or 1.

Step 3:

Repeat Step 2 for all output variables on the horizontal axis.

Step 4:

For some combinations of terms for input variables, there is no one exact term that expresses the desired output value. If this is the case, do not change the membership function definitions. Instead, use FAM to express the ambiguities.

FAM Rules

If a unique consequence for a given combination of input variables cannot be found, FAM rules can be used to express ambiguities.

Example:

Consider the following rule to be represented in a fuzzy logic system:

IF “Distance” = *close* AND “Angle” = *zero*

THEN “Power” = mostly *zero* but somewhat *pos_medium*

One approach is to define a new term as “mostly *zero* but somewhat *pos_medium*.” This approach, however, would result in an excessive amount of terms and membership functions. Moreover, system structure would become unnecessarily complex and difficult to survey.

Using FAM, this ambiguity can be expressed in rules. The following figure shows a possible representation of the example rule.

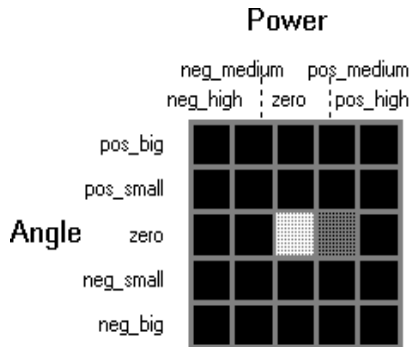


Figure 3–21: Expressing ambiguities using FAM.

fuzzyTECH supports different rule editors: the Spreadsheet Rule Editor, the Matrix Rule Editor, and textual input in FTL syntax from any ASCII editor such as MS-Windows Notepad.




Enter your production rules in either the Spreadsheet Rule Editor or the Matrix Editor (create at least one rule).

Activating a Rule Editor


First highlight a rule block in the Project Editor and press the right mouse button to activate the pop up menus for the rule block. Choose either the Matrix Rule Editor or the Spreadsheet Rule Editor. Another way to activate the Spreadsheet Editor is to double click the left mouse button on the rule block. Each editor also contains a button to toggle to the other editors.

Spreadsheet Rule Editor Window

Each row in the Spreadsheet Editor window (Figure 16) represents a single fuzzy logic rule, with its inputs (conditions) on the left side and its outputs (consequences) on the right side. The input columns of all rules are titled with the [IF] button, the output columns and the DoS columns are headed by the

[THEN] button. Located in the first button row, left of the [IF] and [THEN] buttons, the  button lets you switch to the Matrix Rule Editor window. In the second button row, a button for each column in the spreadsheet exists, showing the respective variable name. For each field in the spreadsheet, the row indicates the rule and the column the variable. The fields themselves contain the term names. Left of each output variable column, a small DoS column indicates the Degree of Support for the output variable. Input and output variable columns are separated by a thicker line.

Utilities

In order to facilitate the design of rule bases, utility functions can be executed from the Rule Block Utilities dialog, which is opened by clicking on the  button located in the upper left corner of the Spreadsheet Rule Editor.

FAM Rules (DoS)

An individual weight may be applied to each rule as shown in the column with the [DoS] button title (Degree of Support). A rule with a Degree of Support of 0 is functionally equal to a non-existent rule. For some design steps, the definition of a rule with a DoS of 0 is useful. For example, the NeuroFuzzy Module can tune rules with an initial DoS of 0 to different values as required.

Edit Rules

Click once with the left mouse button over any field to be changed in order to edit a rule. This activates pop up menus listing the possible values (term names) for the current field. If a linguistic variable in the input section should not have any influence in a rule, select the [...] menu item to leave the field blank.

Sort Rules

The Spreadsheet Rule Editor features various sort options to organize large rule bases. Automatic sorting is provided by clicking on the respective variable name button, causing the rules to be sorted according to the term names of this variable. The buttons on the left side of the spreadsheet indicating the rule number may be used to highlight a rule. Clicking on the [DoS] button causes rules to be sorted in descending order with respect to their Degree of Support.

Create and Delete Rules

New rules may be defined by filling the last empty row, after which a new last empty row is automatically created. When closing the rule editor, all invalid rules are automatically erased. A rule is invalid if neither the input condition nor output consequence is defined. Rules with a Degree of Support of 0 are considered as non-active rules and are not automatically erased when the rule editor is closed. To delete rules, either set all inputs or all outputs to blank [...] to make a rule invalid and close the Rule Editor, or highlight the rules and press [Del]. Highlight a rule by clicking on the gray leftmost field of a rule that displays the rule number.

[IF] and [THEN] Buttons

The inference process of any rule consists of aggregation, composition, and result aggregation. Click the [IF] button to select a fuzzy operator for aggregation. The fuzzy operator for composition is the PROD operator and may not be changed. The result aggregation operator can be switched between MAX and BSUM by using the [THEN] button.




Dynamic Inference

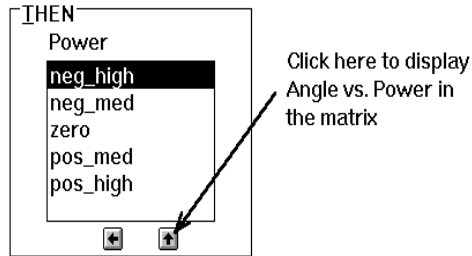
If a debug mode is enabled, the Spreadsheet Editor lets you dynamically trace rules. The degree to which any of the rules fire is displayed graphically to allow visualization of the information flow in the system. In any debug mode, the Spreadsheet Rule Editor displays two small boxes for every rule and output variable located left and right of the Degree of Support value. The height of the black bar in the boxes indicates the aggregation (left bar) and composition (right bar) results.

Rule Matrixes

In *fuzzyTECH*, rule blocks can be displayed and edited as matrixes, rendering even large rule blocks transparent. The rule matrix is displayed in the upper left part of the rule editor window showing the relation between two linguistic variables. Each individual rule maps to a field in the matrix. The selected rule in the matrix is indicated by a red frame and is also shown in the list boxes in the lower part of the window. The selected rule in the figure below reads:

IF "Angle" = *zero* AND "Distance" = *far*
THEN "Power" = *pos_med*.

Any two variables of a rule block are displayed in the matrix with a click on the  and  buttons under each list box. For example, if the display is currently "Angle vs. Distance", click the  button under the "Power" list box to display "Angle vs. Power" in the matrix:



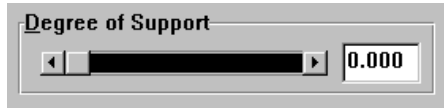
Depending on which button is used, the appropriate linguistic variable is displayed on the horizontal/vertical side of the matrix, and the variable name previously listed is removed. When a variable is displayed in the matrix, its respective buttons are inactive. The button flips the positioning of the two variables.

Rule Definition

The figure displays a matrix for the variables “Angle” and “Distance”. Each white field in the matrix represents a single rule and each black field represents a non-defined rule. Individual rules can be selected by clicking on a matrix field or by selecting them from the list boxes. Define a new rule by double-clicking on the appropriate black matrix field and delete a rule by double-clicking on the corresponding white matrix field.

FAM Rules

Within the matrix, a white field represents a completely plausible rule (100% true) and a black field represents a totally implausible rule (100% false). Gray shades indicate partial plausibility. The degree to which a given rule is plausible is represented as the Degree of Support for this rule. Set the Degree of Support for a rule by clicking on the matrix field for the rule and adjusting the scroll bar:



or use the edit field directly next to the scroll bar.

Adding an individual weight to each rule extends the concept of “on/off” rules to a representation in which rules themselves are “fuzzy.” This allows for the rule base to be fine-tuned during optimization. Since the mapping of the input of the rules to the output is now itself fuzzy, this concept is often referred to as a “Fuzzy Associative Map” (FAM). Figure 3–22 shows the Matrix Rule Editor with FAM rules.

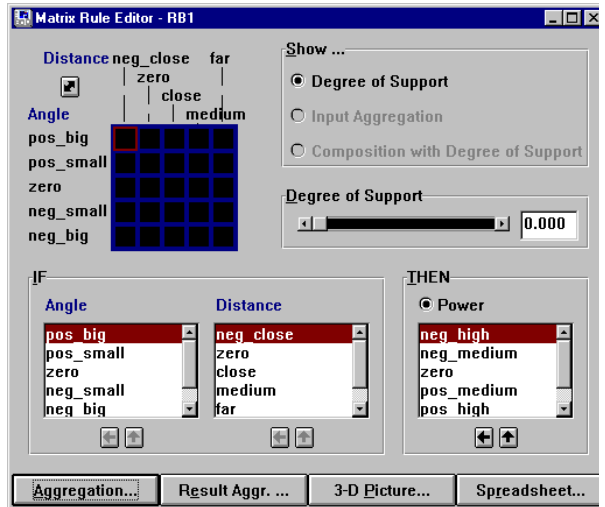
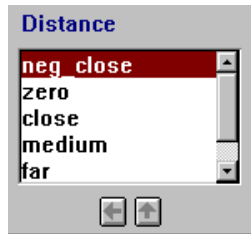


Figure 3–22: Rule Editor window with FAM Rules

Browsing the Rules

Browse through an entire rule base by selecting the terms of the linguistic variables not displayed in the matrix. In the above figure, the variable “Power” is not in the matrix; selecting a term for this variable lets you browse through the rule base:



Show

The Show options are used to select the variable to be shown in the matrix. If a debug mode is active, the Matrix Rule Editor will turn into an analyzer tool.

Degree of Support (Dos)

This display option is automatically selected in Design mode. This option allows the degrees of support to be shown in the matrix. The slide bar and the entry box set the exact degree of support for the selected rule and can be adjusted for the the rules concerned.

Input Aggregation

This display can be selected in any debug mode. The Input Aggregation option provides an intermediate inference result after the inputs have been aggregated. This allows you to visualize the degree of firing for the rule conditions at the actual point of operation, irrespective of the degree of support weighting. Only the rules used in Design mode are shown. All the other rules with degree of support zero are hidden and shown in the background colour.

Composition with Degree of Support

This display shows another intermediate inference result. The composition result shows the degree of fulfillment of the output variables after combining the aggregated conditions with the degree of support for the rules, irrespective of the selected method for the last inference step, the result aggregation. Only the rules used in Design mode are shown here. All the other rules with degree of support zero are hidden and shown in the background colour.



Figure 3–23: Display selection with the Show Group

Off-line Optimization

System optimization steps are usually required to achieve the desired system performance. Because the optimization strategy varies with each type of application, the *fuzzyTECH* development system offers different debugging modes:

Mode	Usage
Interactive:	Lets you check systems response to manually entered input data. (→ section “Interactive Debugging”)
File Recorder:	Lets you process data from a file and display the inference process graphically. (→ section “Performance evaluation using process or generated data”)
Batch:	Lets you process data from an input file to an output file without displaying the inference process (→ section “Performance evaluation using process or generated data”)
Connection, Monitor, Online, RTRCD:	Lets you connect online to a controller running on a separate target system. Online monitoring and modifications are possible; traces can be configured, started, and uploaded.

Of these debug modes, only Connection, Monitor, and Online/RTRCD modes work online, i.e., in real-time. These debug modes are only supported with *fuzzyTECH*'s Online Edition or the RTRCD Modules. The other debug modes are supported by all *fuzzyTECH* Editions. All debug modes are mutually exclusive. The [Debug] menu bar item is only active if a project has been defined or loaded. Within the [Debug] menu, the active debug mode is indicated by a check mark (✓).



Figure 3–24: The different debug modes are mutually exclusive.

Off-line vs. Online

The appropriate optimization tools for your fuzzy system depend upon whether they are to be used on a running process in real-time on target hardware (online development), or whether they run on the development workstation/PC (off-line development). The online optimization technique is a unique *fuzzyTECH* feature in which development, debugging, and visualization features are available *while* the process is under the control of the fuzzy system.

Design Review

Off-line debug modes use the internal computation kernel in *fuzzyTECH* to compute the results of a fuzzy project. Online debugging is based on generated code compiled for the target system. Both enable an interactive system development. Online modifications require access to system parameters in the implemented code. These changes must be restricted to a protected protocol. Thus, not all design parameters can be changed on the fly. In the off-line modes, the internal computation kernel supports all *fuzzyTECH* visualization features and editors. This allows you to review all fuzzy design steps but does not allow modifying the basic structure of the fuzzy project.

Basic Structure of Fuzzy Projects

A basic structure of a fuzzy system consists of the linguistic variables, interfaces and rule blocks. The design specifics of these objects can be changed within the editors, but creating or deleting these objects, or changing their properties, is disabled in all debug modes.

Debug Window

If a debug mode is active, the debug window is displayed (Figure 3–25). Simply close this window or deselect the debug mode to end a debug mode, as shown in Figure 3–24. The debug window always displays the input/output values of all system variables. In the interactive debug mode, the field Value may also be used for manual input. The field Steps[%] lets you specify the step width for the Value field scroll buttons for each variable.



Minimizing the Debug dialog speeds up the display in most debug modes.

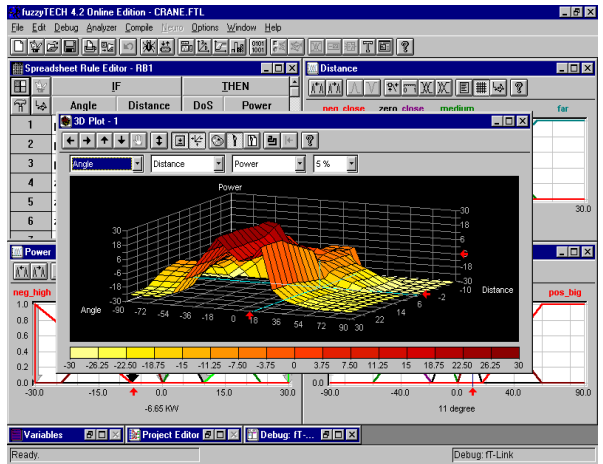


Figure 3–25: The debug window shows all I/O variables and their values.

No Rule Firing

If in any debug mode, no rule for the current combination of input values fires an output variable, then the value of this linguistic output variable is set to its default value. This situation is indicated by a question mark (?) displayed on the right side of the output variable for which no rule fired.


Switch between Debug Modes

You may switch from any debug mode to any other at any time.

Interactive Debugging

The *fuzzyTECH* interactive debugging mode provides system designers with an instant verification of every design step. This feature lets you visualize the results of your design while designing the fuzzy logic system.



Enable the interactive debug mode by selecting [Debug] [Interactive...] or click .

Enable Debugging

This performs a check of your system structure. If errors occur, an Error List window is activated. If no errors occur, the debug window is activated.



Specify input values for the actual system. Review the system behaviors by visualizing the entire fuzzy inference in the *fuzzyTECH* editors.

In debug mode, all design editors such as the Linguistic Variable editor and the rule editor graphically display the entire inference process. A system can also be designed in this debug mode. This accelerates development time because the effects of design changes are now visualized immediately. The effect of membership function or rule modifications made in this mode are immediate.

Though most of the system can be designed in the interactive debugging mode, some restrictions do apply. Structural changes to the system – such as adding new interfaces, linguistic variables or rule blocks – are not possible.

Rule Completeness

The completeness of a fuzzy system's rule base is sometimes difficult to verify. A rule base can be complete in two ways: within the entire control space or within the control space actually used by the system. While the former can be studied using the transfer plot or 3D plot analyzer, the control space actually used depends upon the process itself.

By using the File mode, combined with the Transfer Plot Analyzer in the trace mode, control space operating points are displayed within the Transfer and the 3D plot. This helps pinpoint the process states for which no rules were defined and helps distinguish superfluous rules.

Rule Tracing

Within the interactive mode, rule tracing is automatic. For every control state in the process (namely, a record in the file), the individual firing degree of every rule can be visualized.

Performance Evaluation Using Process or Generated Data

fuzzyTECH allows you to test the fuzzy system's behavior with data. The most significant data is recorded process data. Testing the fuzzy system with process data requires the use of preprocessed data:



Test your fuzzy system performance on real data with the Batch or File mode. Collect preprocessed data from your process and use the pattern generator to evaluate the system behavior at all operating points.

If your fuzzy system is already installed within your process or a simulation, the *fuzzyTECH* Trace Analyzer allows you to trace the process at the interfaces of the fuzzy controller (See the analyzer section of this chapter).

If the preprocessing is not yet determined, the *fuzzyTECH* DataAnalyzer module lets you apply signal conditioning and statistical methods to the data. The File mode enables you to connect your system to pre-recorded sample data files for interactive analysis. Batch mode processes files with input data and generates a result file for further processing with other software.

File Format

Both Batch mode and File mode use the same data format. Two formats are possible: a tiny and a descriptive format. Both formats are depicted in detail in the Reference Manual. The tiny format contains:

- a line where the variable names are printed; a # as the first character of the file indicates the tiny file format and the variable name line.
- data lines containing a column for each variable, separated by blanks or tabs.

The descriptive format uses:

- comment lines as the first and third lines
- the second line to print the variable names (without # as an indicator).
- data lines beginning with a sample reference (a name with a maximum of 10 ASCII characters), followed by the data columns with one for each variable and separated by a blank or tab.

In File mode, the number of records in a file is limited; in Batch mode this number is unlimited.



Activate the Batch mode with [Debug][Batch]. A prompt for the input and output file names appears.

Batch Output File

The output file contains all input variables, the result for the computed output variables and a column “– flags–”. This column indicates output variables for which no rule has fired. The Batch mode overwrites the data line with the variable names.

File Names

Generally, you are free to choose both input and output file names. As defaults, *fuzzyTECH* uses the following conventions:

- The file names are the same as the FTL file name of the actual project.
- Input files containing real data are identified by the .IN extension.
- Input files containing data sets generated by *fuzzyTECH*'s pattern generator are identified by the .PTN extension.
- Input files containing trace records from the *fuzzyTECH* Trace Analyzer are identified by the .TRC extension.
- Output files from the Batch mode are identified by the .OUT extension.
- Sample files for the NeuroFuzzy module are identified by the .EXP extension.
- Files containing data to be clustered are identified by the *.DAT extension.
- Files containing data sets that were removed from a *.DAT file during clustering are identified by the *.SKP extension.

Batch Mode

Select [Debug][Batch] and specify the input file (.IN, .TRC, .PTN or other) and the output file. During processing, the inference flow is not displayed. You may interrupt the Batch mode by pressing [Stop] on the Batch Mode Message box that is displayed during computation.

File Mode

The File mode allows pre-recorded sample data to be used for interactive development. The File mode has all the functions of the interactive mode except that sample data, rather than manually entered data, are used as input values. The File mode uses the same data file format as the Batch mode, but instead of creating an output file, the entire processing of the sample data is visualized

graphically. In order to start the File debug mode, select the File Recorder option of the Debug main menu item and specify the input file to be used. After clicking [OK], *fuzzyTECH* reads the entire file to check for errors. You may stop this process any time by pressing the [Esc] key. Browsing through the data file is controlled by the File Control window, which is active as long as *fuzzyTECH* is in File debug mode (Figure 3–26).

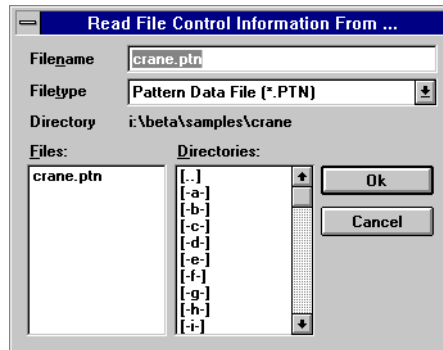
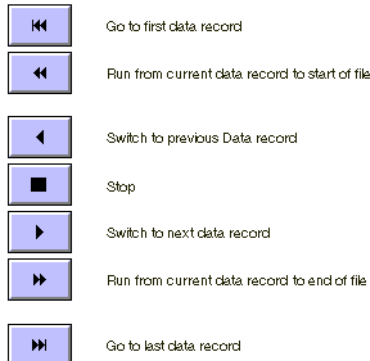


Figure 3–26: File Control window



The File Control window is always at the top of any other *fuzzyTECH* window.



By moving the scroll bar, you may directly access any record. In order to change the file, click on the [Read File...] button (Figure 3–26) and specify a new file name.

Performance Evaluation with Generated Data

Process data normally does not cover the complete operational range of the process. The pattern generator may be used to produce sets of input data in a file for later use with the File Debug mode and Batch Debug mode.

Pattern Generator

The pattern generator is started by selecting the [File][Pattern Generator] menu option. The pattern to be generated is specified in the Pattern Generator dialog box (Figure 3–27). Every input variable of the system is listed in the list box at the top by name, range (from; to), and the step width for the pattern to be generated. First, select a variable in the list box to change the pattern specification. The upper and lower margins of the pattern's interval are specified in the "From:" and "To:" edit field, while the resolution (size of the steps) is specified in the "Step:" field.

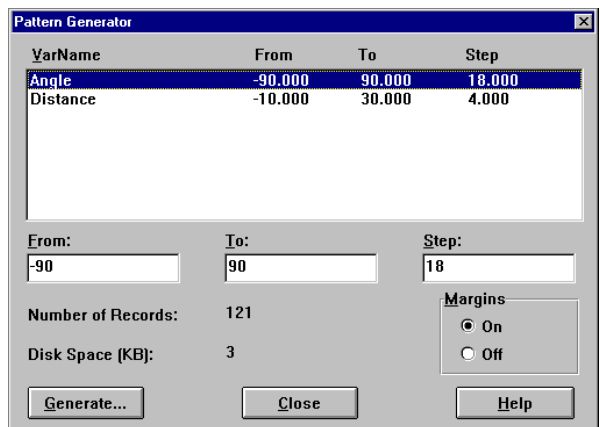


Figure 3–27: Pattern Generator dialog box

Margins

If you specify an interval for a variable that is not a multiple of the step size, the upper margin (the value specified in the “From:” field) is included in the pattern. You may however suppress this by selecting the “Margins” off setting in the Pattern Generator window (Figure 3–27). If, for example, you specified “From: 0”, “To: 10”, and “Step: 3” for a variable, then the pattern would comprise the values 0, 3, 6, and 9, if the margin option is set to off or the values 0, 3, 6, 9, and 10 if the margin option is set to on.

Disk Space

Generating complex patterns may result in large data files. The field “Number of Records:” shows the computed number of records which would be generated with the current settings. On this basis, the field “Disk Space (KB):” indicates an estimation of the disk space required for this data file (Figure 3–27). Click on the [Generate] button to start the generation of the pattern. If the pattern consists of more than 10000 records, a warning box lets you confirm the action.

File Format

The file format in which the pattern data is generated is compatible to the file format used in the Batch Debug mode and File Debug mode. The suggested file extension for the generated pattern is *.PTN. During the generation of the pattern, a dialog box shows the progress and lets you interrupt the process by clicking on the stop button.

fuzzyTECH Analyzers

fuzzyTECH offers analyzer tools to verify the fuzzy system's behavior:

Analyzer	Usage
Transfer Plot	Lets you analyze the static input/output characteristic of a fuzzy logic system as a color plot, showing cross-sections for the current operating point.
Trace Transfer Plot	Lets you analyze the dynamic input/output characteristic of a fuzzy logic system.
3D Plot	Lets you analyze the static input/output characteristic of a fuzzy logic system as a scaleable, rotational 3D plot.
Trace 3D Plot	Lets you analyze the dynamic input/output characteristic of a fuzzy logic system in the 3D plot.
Time Plot	Lets you analyze the time response characteristic of a fuzzy logic system.
Statistics	Lets you control the use of fuzzy rules by a statistics column in the Spreadsheet Rule Editor.
Trace	Lets you log system inputs. The logged data can be uploaded and saved. The File Debug mode allows you to replay, step through, visualize and analyze the trace.

All analyzers can be used in any debug mode; only the Batch Debug mode does not support system visualization and analyzers. If no debug mode is

activated, the [Analyzer] menu bar item is disabled. Disabling a debug mode closes all active analyzers.

Save Window Positions

The configuration and the positions of all open analyzer windows are saved if the option Save window position in the [Options][Preferences] dialog is enabled. *fuzzyTECH* uses a .CFG project file to save the project configuration even when you exit *fuzzyTECH*.

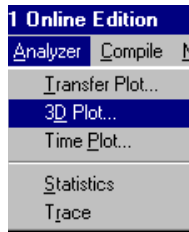


Figure 3–28: Different analyzers can be activated in the analyzer menu.

Trace Transfer and Trace 3D plot

Trace Transfer Plot and Trace 3D Plot are options that plot current operation states dynamically in the plots of the transfer characteristic. You may switch to these options using the Transfer Plot's or 3D plot's toolbar or pop-up menu.

Transfer Plot Analyzer

The transfer plot (Figure 3–29) shows a cut in the control space. Two input variables and one output variable are always shown in this graph. In a fuzzy system with more than two input variables, the non-selected variables possess a given value, which is either defined (in interactive debug mode) or determined by the linked process or data file (in other debug modes). The two input variables are shown on the horizontal and vertical axis, and the value of the output variable is displayed as the color

of the area spanned by the input variables. The color bar on the lower part of the window shows the color-value relation. The resolution of the transfer plot can be specified separately for each input variable with the “Step Width [%]:” field in the Transfer Plot Configuration dialog.

Control Areas not Covered by Fuzzy Rules

The plot only paints colored values in regions where the result is evaluated by fuzzy rules. If no rule covers an operation point, a default value is used for computation and the plot shows a black colored area.

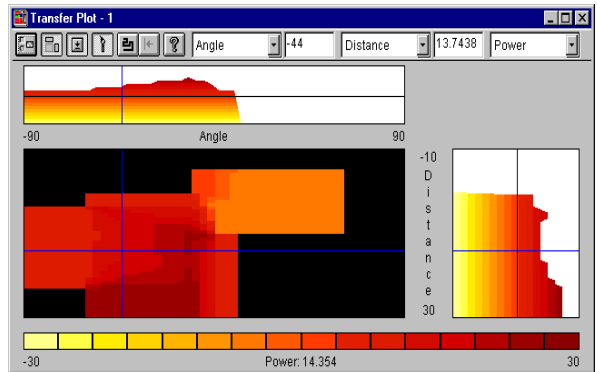


Figure 3–29: Transfer Plot analyzer window



Use the transfer plot to check the completeness of the rule base. Make sure for all operational states not covered by fuzzy rules that the default value is the appropriate result. If you work on arbitrary rule bases, use additional interfaces for intermediate variables to display partial transfer characteristics of all used rule blocks.

Cross Sections

For a given control state, two cross section views – vertical and horizontal – of the transfer characteristic are also displayed above and to the right side of the transfer plot. In Figure 3–30, the output value “Power” is displayed over the variation of the input variable “Angle”. All other input variables remain the same.

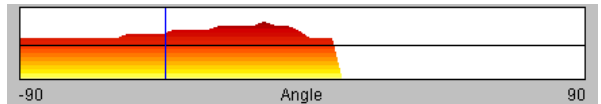


Figure 3–30: Cross section view for “Angle”

Trace Transfer Plot

The transfer plot can be controlled via its own toolbar. Selecting the [Trace] button switches the transfer plot to the trace transfer plot and back. Rather than just the actual operating point, the trace transfer plot draws a history of all operational states since the trace transfer was enabled. In order to reset the trace, select the [Reset] button from the toolbar. Note: the cross-sections are not displayed in the Trace Transfer Plot mode.

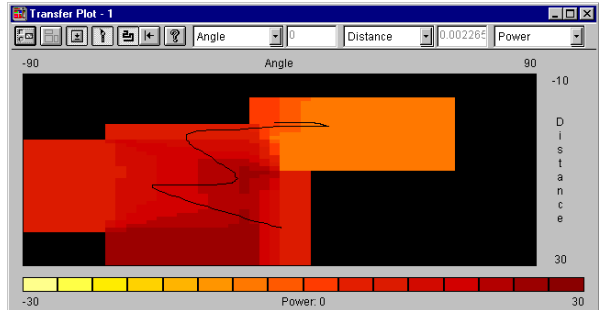


Figure 3–31: Trace Transfer Plot



Enable a debug mode and press the 3D button in the main window toolbar. For a project with two inputs and one output, the 3D plot is automatically displayed.

3D Plot

You can display the cut in the control space as a 3D picture. The 3D plot has its own toolbar. Use the rightmost drop down list box (the arrow-down button displays the list of the possible resolutions) to change the resolution of the 3D Plot. A high resolution can result in slow response times of the plot, depending on the video and computing performance of your PC. You can select the variable used in the plot by using the three fields displaying the variable names.

Rotate the Plot

The four arrow buttons on the left side of the toolbar rotate the plot in all directions. A double click on an arrow button starts a (continuous) rotation; the hand button stops this rotation. The double-arrow buttons flip the 3D Plot for better visibility.

Repaint and Background Paint in Dynamic Plots

If the output variable selected for the 3D Plot depends on more than the two inputs displayed, the control surface dynamically depends on these variables. If the Repaint button is pressed, the 3D Plot is repainted every time one of these other input variables changes to reflect the change in the control surface shape. Note: this only results in a smooth display if you have a PC with very high video and computing performance. You can enhance this effect by enabling the background paint mode. This creates the picture in the background and dumps it to the screen.



Cross-check your fuzzy rule base by displaying the transfer characteristic of all rule blocks. For systems with more than three (3) inputs, make use of the dynamic plot behavior by enabling the background paint mode. Changing variables not assigned to a plot axis lets you display the dynamic change of the transfer characteristic. Rotate or flip the plot to display hidden plot parts.

Trace Operation Points in 3D Plot

The red arrows at the three axes and the cyan plot lines always show the current value of the input and output variables. The Tracing button also lets you trace the operating point over time (green trace). Enable Tracing, then reset and start the crane simulation. The “green trace” begins at the start position of the crane, that is Angle=0 and Distance=22, and moves to the target position Angle=0 and Distance=0. The height of the control surface and its color indicate the reaction of the fuzzy logic controller to the combination of the input variables.


Trace

The Trace option lets you configure a trace buffer on the target hardware or in the internal computation kernel of *fuzzyTECH*. A trace buffer must be assigned to your fuzzy controller to enable the Trace Analyzer. Activate the Global Options dialog by selecting [Options][Global]. The Global Options dialog can not be accessed when a debug mode is open. Enable the trace check bar and select a trace buffer size. Now the trace window can be activated with [Analyzer][Trace].

Using the Trace

A trace can be triggered either from the target hardware or from the [Start] button in the Trace dialog in *fuzzyTECH*. After completion, the contents of the trace buffer can be [Upload]ed and stored in a file in *fuzzyTECH* data format. The file recorder lets you browse through the trace data and “what-if” analyses can be made.

Close Analyzer

Double-click on the system icon or  of the analyzer window to close a plot analyzer. If you leave the debug mode, all analyzer windows are automatically closed. If you have enabled “Save Window Position” in the Preferences dialog, all plots closed when leaving the debug mode are automatically re-opened.

Online Optimization

Although the resulting system may work well using the process data or the simulation, the performance of the fuzzy controller on the real system varies in most control applications. This is due to various reasons. When real process data has been used, feed-back of the control actions is not taken into consideration. When using a simulation, model-based simulations are almost always approximations or simplifications of the actual system’s behavior. In this case, online optimization is the appropriate tool. However, MCU and Precompiler Editions do not support online optimization. A limited “Online” function for MCU-Edition-based development is provided by the RTRCD modules that are available as add-on modules.

Online

Follow these steps to optimize a real crane online:

- Change the properties of a variable term (S shape, L shape)
- Change the output defuzzification method (CoA, CoM, ...)
- Load the fuzzy code of the running system from the target platform to the PC.

Monitor

The Monitor debug mode can only be used for visualizing the fuzzy system running on the target

platform. Modifications cannot be carried out online. However, it is possible to use the Trace analyzer to record data from the system.

Necessary steps:

- ▶ Activate the “Test and Commissioning” icon in the Sucosoft Manager.
- ▶ Select the “POU Editor Online” button in the main editor.

The term “Resource” is shown in the left-hand part of the Program window.

- ▶ Select the instance of your fuzzy system and then click the “Display/Change POU” button.

Fast Processes

Some processes are too fast to be optimized on-the-fly. Here, the trace mode should be used. After triggering, the controller stores the system’s real-time data for a defined interval. This data is transferred by the link to the development system for further analysis, allowing the optimization of even very fast processes. The trace mode can also be used for documentation purposes, working as a ‘data recorder.’ The file format of the trace mode is compatible to the File mode and may be used for further processing.

Revision Control System

Like most design software, *fuzzyTECH* stores all information on a system under development in a file. In the case of *fuzzyTECH* this is an ASCII format file in FTL syntax with the suffix *.FTL. Optionally, *fuzzyTECH* stores the information on editor and analyzer configurations in a *.CFG file that is an ASCII format file following the same syntax rules as most MS-Windows *.INI files. While this is the standard way to handle project files in software development, often the modifications to a system under design must be documented. This is a must for developers that are required to document system modifications in an ISO9000 compliant fashion. In software engineering, such documentation is handled by a Revision Control System.

The objectives of a Revision Control System are to:

- Ensure that only the appropriate persons can conduct modifications,
- Document the modifications and the related comments of the developer, and
- Allow the user to scroll back to an earlier stage in development.

Such Revision Control Systems are available from a large number of vendors, and you can integrate *fuzzyTECH* and its FTL files as source files with any commercial Revision Control System. However, Revision Control Systems are powerful and complicated tools, and if you do not already use one that has been set up, you can use the integrated Revision Control System of *fuzzyTECH*. Because of its tight integration with *fuzzyTECH*, revision control becomes an easy task, plus the overhead cost required to work with a revision control system is easily recovered the first time you decide to revert to an earlier stage of development.

Revision Control System

fuzzyTECH's Revision Control System (RCS) uses *.REV files to store multiple *.FTL files, plus additional information describing the modifications from previous development stages. Each development stage is represented by its *.FTL file. In brief, a *.REV file contains the sequence of *.FTL files of a development project. You can store each *.FTL file that represents a specific stage of development in the *.REV file and reclaim each *.FTL file from it later. The RCS uses a compressed and encrypted format for *.REV files. Hence, even a large number of *.FTL files can be stored in a *.REV file in a compact fashion. The RCS enforces password encryption to ensure that only appropriate persons can modify revisions of the fuzzy logic system.

Work with Existing REV Files

Open the existing file CREDIT.REV, which is contained in the sub-directory \SAMPLES\CREDIT\..., to become familiar with the RCS. You can either select File\Open... from the main menu bar and set Filetype to Revision File (*.REV), or select File\Revision Control... from the main menu bar and click on the [Open REV...] button. Both ways open the Revision Control dialog.

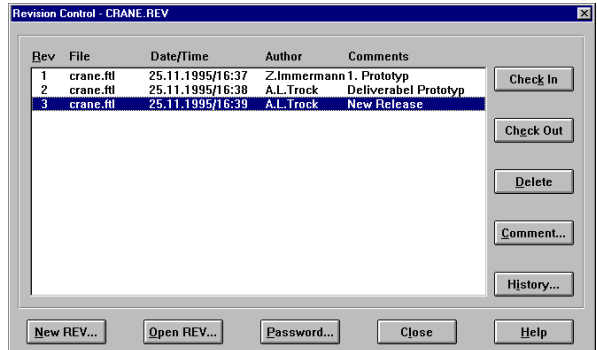


Figure 3–32: The Revision Control dialog lists all stages of a Fuzzy System development.

Password Protection

The RCS enforces the use of passwords to protect the REV file from unauthorized viewing and editing. For CREDIT.REV, the password is “fuzzy”. You can change the password of an opened *.REV file by selecting the [Password...] button from the Revision Control dialog.

Working with Revisions

The list box in this dialog reports all revisions that are stored in the *.REV file. In CREDIT.REV, five stages of CREDITx.FTL, are stored. The first column of the list box shows the revision number. The numbers show the sequence in which revisions have been stored in the RCS, and cannot be changed. The File column shows the original filename of the *.FTL file that was stored in the *.REV file. The Date/Time column shows when the *.FTL file was added to the *.REV file. The Author and Comments columns show who developed the revision and what was changed with respect to the previous revision. The column Comments shows only the first part of the first line of the comments. You may look at the comments in detail by selecting the revision in the

list box and clicking on [Comment...]. You cannot modify the columns Rev, File, Date/Time, Author, and Comments of an existing revision.

Retrieving FTL Files from REV files

If you want to view or modify a revision, you need to log out this revision from the RCS. First, select the revision in the list box, then press [Check Out]. Because an *.FTL file with the same name exists, the RCS lets you now specify a new file name. Then, the revision is automatically opened with *fuzzyTECH*. You can store the file during development as any other *.FTL file.

Storing FTL Files in a REV File

When a new stage in development is reached that you would like to document and store in the RCS, you have to open the respective *.FTL file with *fuzzyTECH*. Then, open the Revision Control dialog again by selecting File\Revision Control... from the main menu bar. If you have restarted *fuzzyTECH* since the last time you opened the *.REV file, you have to open it again by clicking on the [Open REV...] button. Click on [Check In] to store the current *.FTL file as new revision. This opens a dialog that lets you specify the author's name and describe the modifications made.

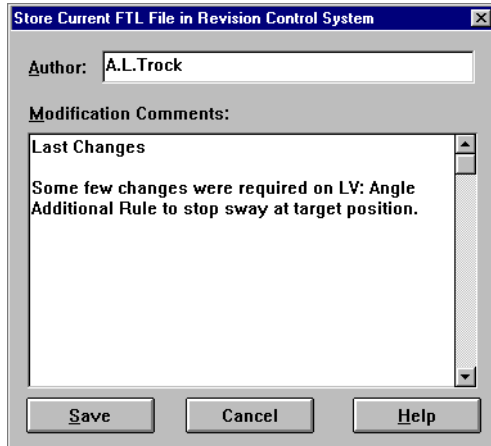


Figure 3–33: When you store a revision in the RCS, you specify the author's name and describe the modifications

In order to create a new file, just click the [New REV...] button in the Revision Control dialog and specify a password for this *.REV file. Make sure that you note the password, as there is no way to access a *.REV file without this password. The password can only consist of standard alphanumeric characters and must be at least five (5) characters long.

Printing a Complete Development History

Click on the [History...] button to print a complete development history that covers all revisions. This exports all the revision information plus all comments in a single text file that can be edited, printed, or integrated with other documents.

Compact File Format

Note: the binary *.REV file format is much more compact than the textual *.FTL format. While the five CREDITx.FTL files together use more than 50 KB of disk space, the CREDIT.REV file that contains all of them in condensed format uses less than 10 KB of disk space. You can also use the *.REV file format, which stores only one *.FTL file, if you either need a more compact representation than *.FTL or if you want to protect the *.FTL file from being read by unauthorized persons.

Index

Symbols

Steps	136, 147
.DAT Extension	141
.EXP Extension	141
.IN Extension	141
.OUT Extension	141
.PTN Extension	141
.SKP Extension	141
.TRC Extension	141

Numerics

3D Plot Analyzer	18, 149
------------------------	---------

A

Aggregation	81, 95, 123
Analyzers	145
AND	82, 95, 96
Applications of Fuzzy Logic	62
Approximate Logic	87
Arbitrary MBF	94

B

Base Variable	23, 70, 90
Base Variable Dialog Box	107
Batch Debug Mode	134, 141
Boolean Logic	62, 82

C

Center-of-Area Method	100, 122
Center-of-Gravity Method	87, 100, 122
Center-of-Maximum Method	87, 98, 122
CoA	100, 122
CoA Defuzzification	100
Code Range	108
Code Values	24, 107
CoG	100, 122
CoM	98, 122
CoM Defuzzification	98
Compensatory Operators	96
Composition	81, 83

Composition Operator	97
Compute MBF	121
Consequence	95
Container Crane Control	73
Continuous Logic	62
Cross Sections	148

D

DDE	134
Debug Dialog	136
Debug Modes	135
Default Value of Linguistic Variables	108
Defuzzification	84, 89, 98, 120
Defuzzification Methods	122, 123
Degree of Membership	89
Degree of Support	97, 131
Display Selection with the Show Group	133

E

Enable Debugging	138
------------------------	-----

F

FAM	97, 131
FAM Inference	97
FAM Rules	125, 131
Field	136, 147
File Debug Mode	134
File Mode	141
File Names	141
fT-Link Debug Mode	134
Fuzzification	77, 78, 88, 89, 120
Fuzzy Associative Map	97, 131
Fuzzy Control	88
Fuzzy Design Wizard	54
Fuzzy Input	121
Fuzzy Logic	
Comparing Fuzzy vs. Conventional Control	74
Inference	78, 80
Operators	82
Primer	62
Rules	88
Fuzzy Output	122
Fuzzy Rule Inference	88, 95
Fuzzy Set Theory	68

G	
Grid	114
I	
IF-part	95
Implementation	103
Inference Method	123
Input Interfaces	121
Interactive Debug Mode	134, 137
Interfaces	116
Inverse Term	113
K	
Keyboard Use	116
L	
Label	89
Lambda-Type Membership Functions	90
Linguistic	
Labels	78
Terms	72, 78
Variable	71, 78, 89
Linguistic Control Strategies	75, 76
Linguistic Variable Editor	105
Linguistic Variables Wizard	103
Look up MBF	121
Look-up Table	121
L-Shaped MBF	109
M	
Margins	144
Matrix Rule Editor	126
MAX Operators	96
Maximum Operator	82, 96
MAX-MIN Inference	96
MAX-PROD Inference	96
MBF	90
MBF Definition	112
MBF Type	109
Mean-of-Maximum Method	99, 122
Membership	
Degree of	71
Membership Function	71, 90, 105, 109
Definition of	110

MIN Operator	96
Minimum Operator	82, 96
Model-based Control	74
MoM	99, 122
MoM Defuzzification	99

N

No Rule Firing	137
NOT	82

O

Off-line Optimization	102, 133
Online Debug Mode	134
Online Optimization	102, 152
OR	82, 95, 96
Output Interfaces	122

P

Pattern Generator	143
Performance Evaluation with Generated Data	143
Performance Evaluation with Real Data	139
PID Control	74
Pi-Type Membership Functions	90
Probability and Fuzzy Logic	65
Project Editor	116
Prototyping	102
Psycholinguistics	64
Psycholinguistic Research	92

R

RCU Debug Mode	134
Real-Time Optimization	102
Remarks	116, 119
Result Aggregation	97, 123
Rule	
Condition	95
Consequence	95
Rule Block Icon	122
Rule Blocks	116
Rule Completeness	138, 147
Rule Editor Window with FAM Rules	132
Rule Tracing	139
Rule-based Fuzzy Logic Systems	72
Rules, Browsing the	132

S

Shell Values	24, 107
Singleton Membership Functions	100
Spreadsheet Rule Editor	126, 127
S-Shaped MBF	92
Standard MBF	90
Statistics Analyzer	43
Structure of a Fuzzy Logic Controller	77
S-Type Membership Functions	90

T

Term	89
Term Name	108
THEN-part	95
Time Plot	14
Trace	45
Trace Transfer Plot	149
Transfer Plot Analyzer	147

U

Uncertainty	
Lexical	64
Mathematical Principles of	63
Stochastic	63

V

Verification of a Fuzzy Logic System	102
--	-----

Z

Z-Type Membership Functions	90
-----------------------------------	----