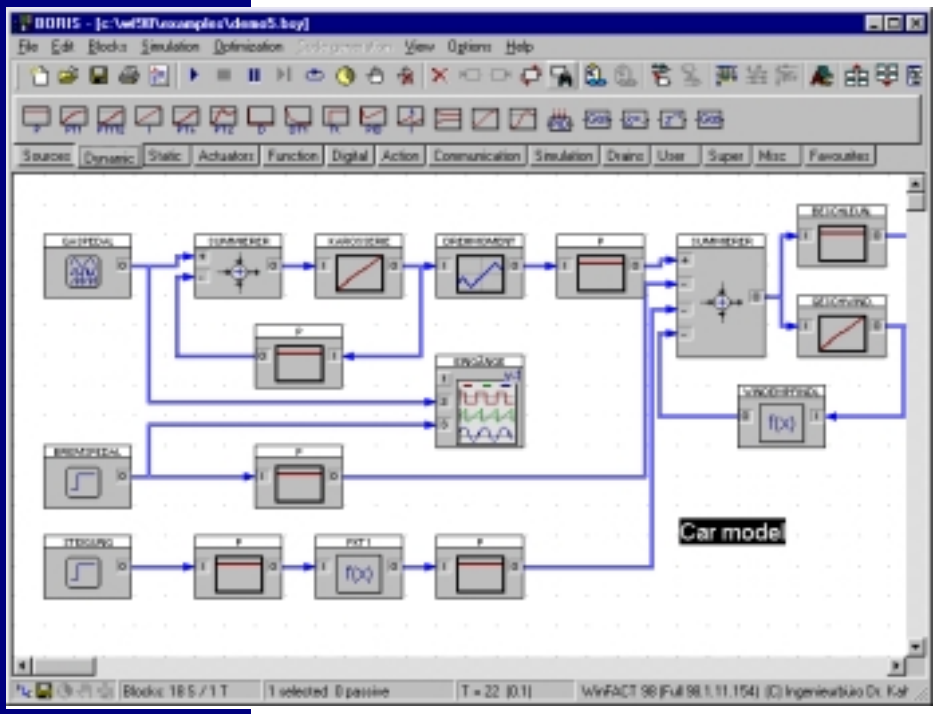
WinFACT



WinFACT 6

User Manual

Ingenieurbüro Dr. Kahlert

Ludwig-Erhard-Str. 45
D-59065 Hamm

Contents

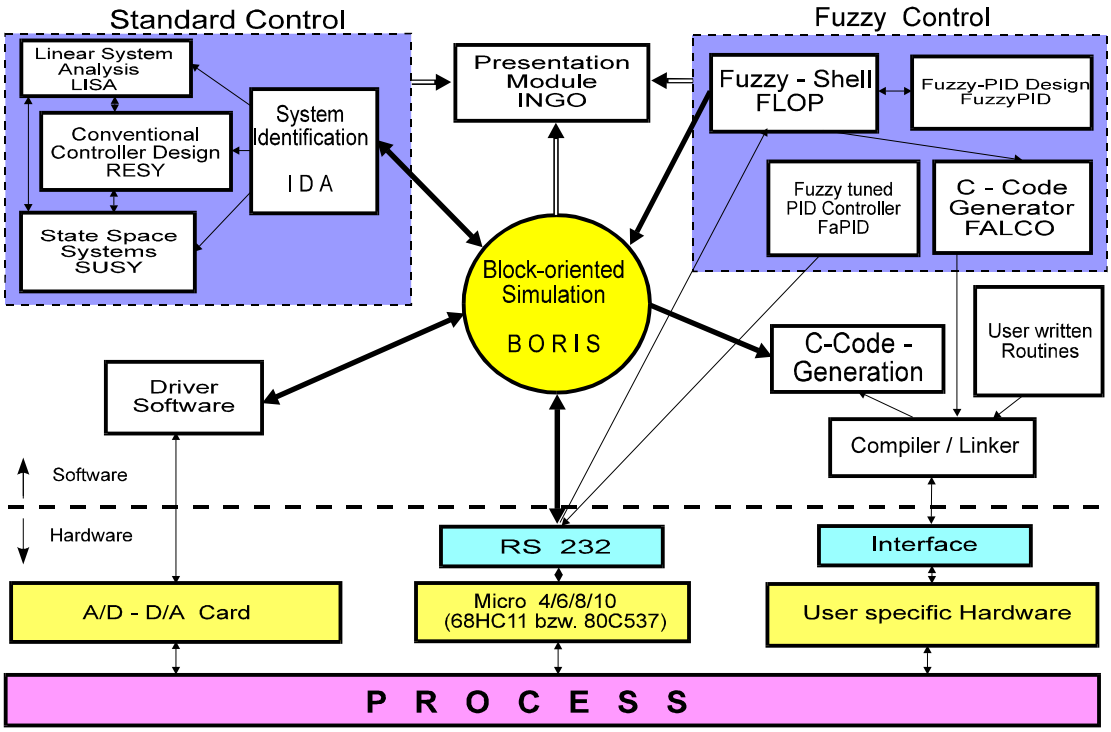
About this manual

This documentation has primarily been written for WinFACT-Users who want to get an overview of WinFACT's possibilities parallel to first experiments with the software itself and to the use of its help documents. For experienced users the use of the expanded help options of the programs is recommended.

This documentation mainly consists of two parts:

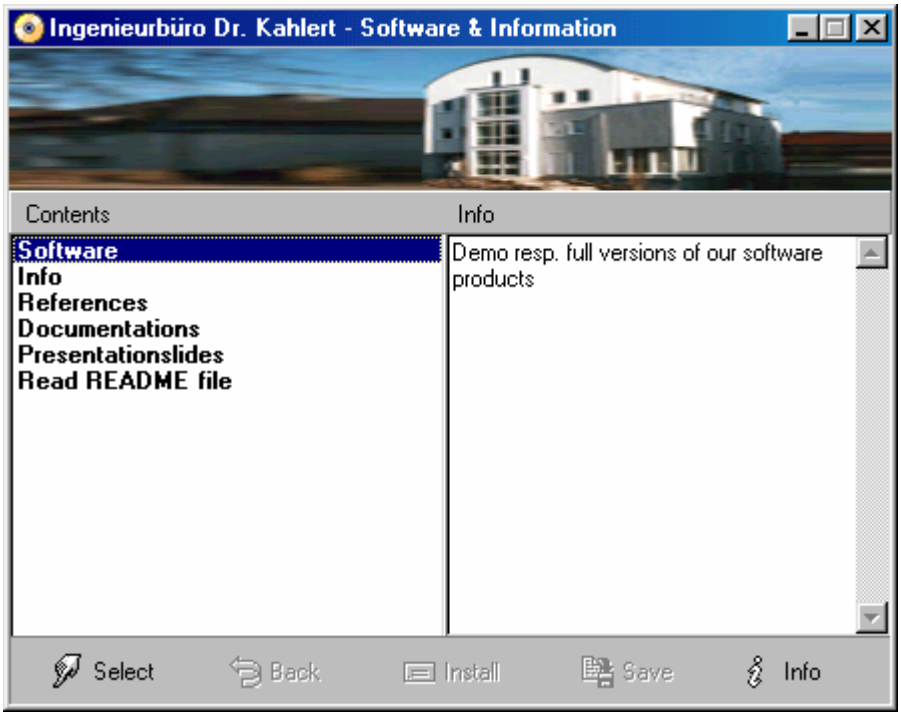
- A short description of the most important features of WinFACT in general
- An overview of all WinFACT-modules and their specific features and options. Included is the description of sample files delivered with the EXE-files which can be used to test all options of the individual modules.

The following diagram shows the interaction of all WinFACT-modules and the different interfaces between software and hardware resp. real processes.



Installing WinFACT

To install WinFACT insert the WinFACT 98 / 6 CD-ROM into your CD-ROM drive. If the Setup program does not start automatically, just start it from the CD by START.EXE. The installation is completely assisted by a dialog so that you have nothing more to do than to follow the instructions given on the screen.



WinFACT 98 / 6 Installation Program

General instructions

WinFACT-File types

All files that are used by WinFACT for the storage of system data are ASCII text files. The main advantage of this file format is that - parallel to the handling within WinFACT - the user has the possibility to view and/or edit all files by a standard text editor. For different purposes and system types WinFACT uses different file types which can be identified by their extensions. The following file types are available:

Extension	File type
UFK	Transfer function
SIM	Simulation results (time responses $y(t)$)
XY	General pairs of values
MXY	Trajectories stored by module SUSY
BD	Frequency response (Bode plot)
OK	Frequency response (Nyquist plot)
VEK	Vectors
MAT	Matrices

Extension	File type
ZRM	State space systems
FWM	Function value matrices (Contour lines resp. 3D-graphics)
FUZ	Fuzzy system files
BSY	BORIS-simulation structures
SBL	BORIS-superblock files

File operations

File operations are managed by the *File* submenu, which is the first submenu of all in all modules. A new file can be opened by *File / Open...*, the F3-key or the toolbar. Existing data can be stored by *File / Save..*, the F2-key or the toolbar (not available in demo version!). If a set of data is stored for the first time, a dialog for editing the filename will be shown.

The menu option *File / New* or the specific toolbar button normally clears the actual display and all user data and resets the program to its state when it was started. If the actual data have not yet been stored, a warning is displayed.

The menu option *File / Project info* or the Strg I-key combination gives you the possibility to look at or edit the file information stored in the data file in addition to the data themselves. If you modify these file information it will be stored together with the data next time you save the file.

Printing and exporting graphical results (not available in demo version)

The graphic that is displayed in the active window of each module can be printed and - in most cases - exported at any time the module is running. Alternatively - as usual in standard WINDOWS-applications - the graphic can be copied as a bitmap to the WINDOWS-clipboard by pressing the <Alt> <PrintScreen> resp. the <PrintScreen> key. From there the bitmap can be pasted into any other WINDOWS-application later.

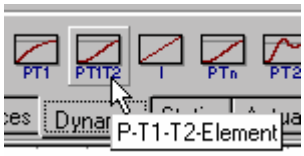
To print the actual graphic use the menu option *File / Print*, the Strg P-key or the toolbar button with the printer symbol. The print is sent to the standard printer that is actually configured. To choose another printer, simply use the menu option *File / Setup printer*. The quality of the print only depends on the printer resolution (not on the resolution of your graphic adapter). To export the graphics in WMF-format (a format that can be imported by nearly any WINDOWS-graphic program, e. g. CorelDraw) use the keys Strg X or the toolbar button with the Export text. Before the graphic is saved you have to choose the filename with the WMF-extension.

Help options

First WinFACT offers the typical help funtions of WINDOWS-applications, which can be started by the menu option *Help / Index*. The following help application is WINDOWS standard so that we don't have to talk about it in detail.

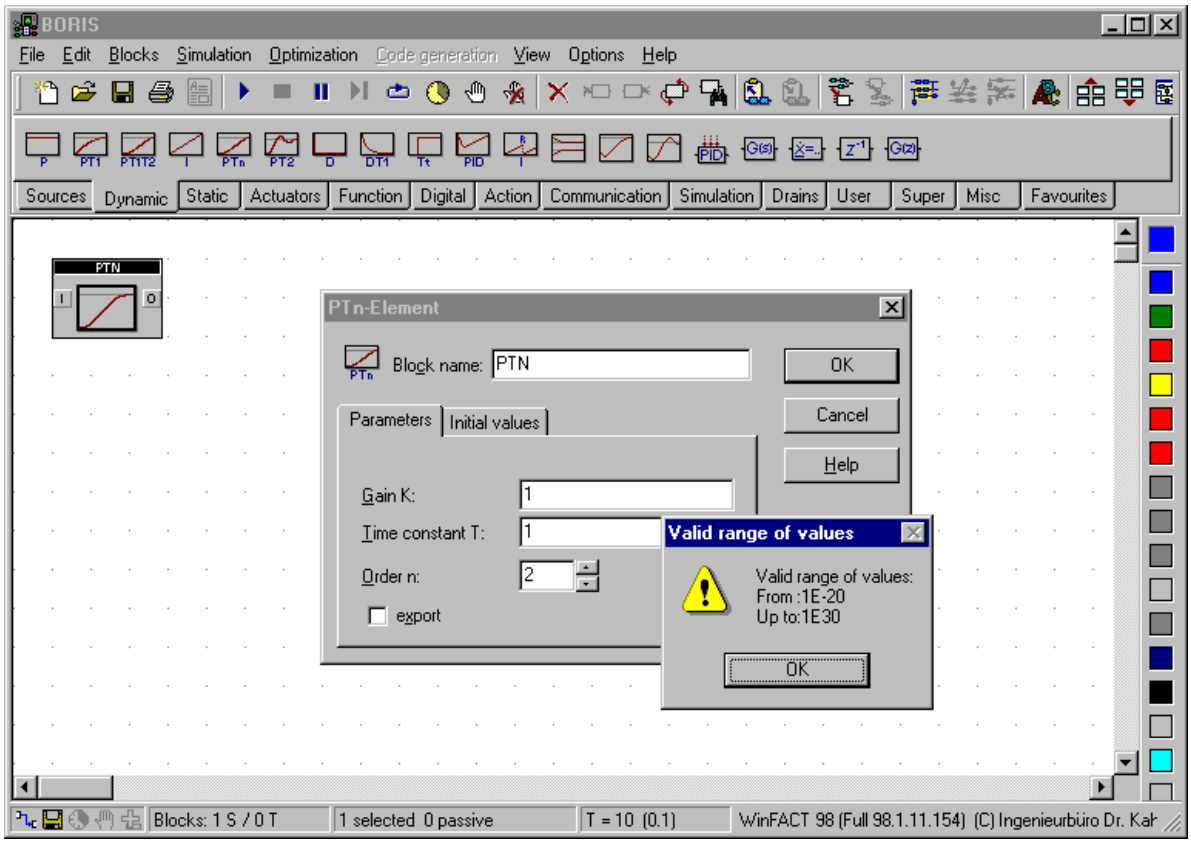
In addition to this standard help function some WinFACT modules offer a *menu help*. This help function explains the function of all menu options in the status bar if you select the specific option by mouse or keyboard.

Especially for new users of WinFACT the *toolbar help function* is of great importance. This function can be activated (default) by the module WFSETUP (see later). If it is activated, any time you move the cursor on a toolbar button for more than one second, a little help window appears which gives you information about the function of this button. By that you are able to learn the operations linked with all toolbar buttons in very short time.



Toolbar help function

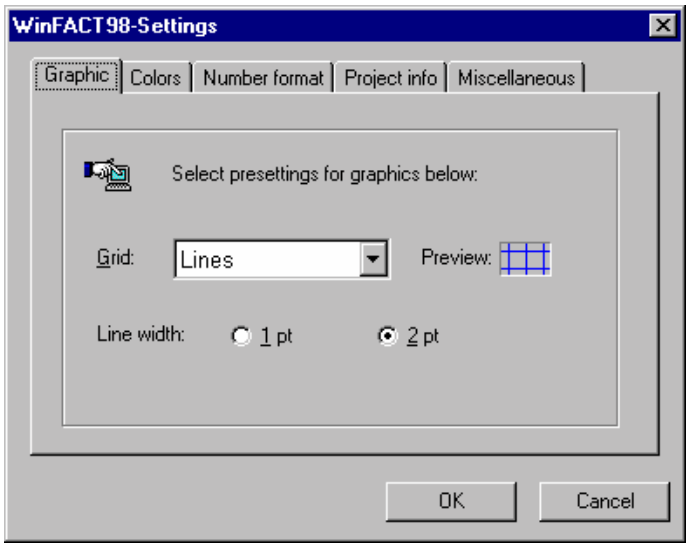
When editing numerical values in parameter dialogs you often have the problem not to know the range that is valid for the variable. To avoid warnings or error messages, WinFACT offers a *numerical range help* for all edit fields in that way, that a message box with the valid range is displayed if you push the *right* mouse button within the edit field. In some cases prior to that a popup menu appears where you have to select the *Range...* menu option.



Numerical range help of WinFACT

Configuring WinFACT by the WFSETUP-module

By using the setup-module WFSETUP, which is automatically installed in your WinFACT-program group during installation (*WinFACT 98 / 6 Settings*), you are able to predefine some default settings for all WinFACT-modules. All your changes you have made are saved in the Windows registry.



Setup program WFSETUP

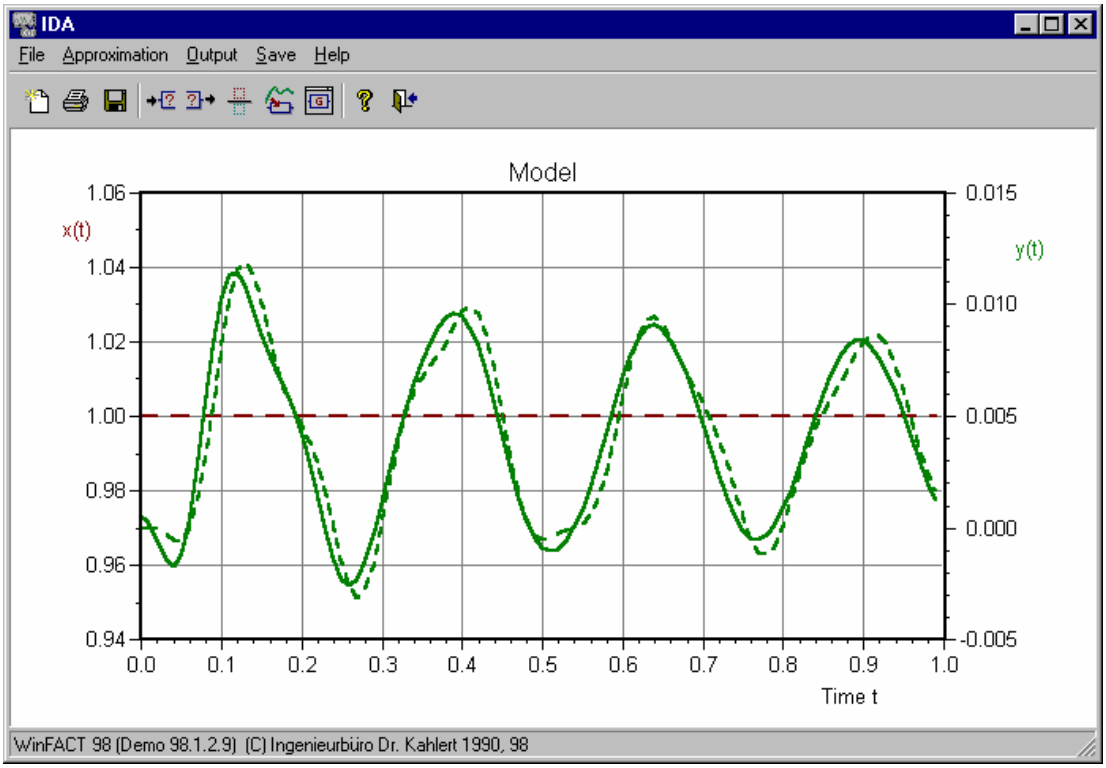
Systemidentification by IDA

IDA offers the automatical identification of linear systems by measured input/output data. The input signal used for identification can be of any kind. The numerical algorithm used for identification is the *multiple integration method*, which especially shows very great robustness in the case of noisy data. The mathematical model found by IDA is a rational transfer function

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} e^{-T_t s}.$$

The order of the numerator *m* and the denominator *n* may be set by user before identification or they are determined by IDA automatically. The dead time T_t also is determined automatically after the lower and upper bound of the dead time have been specified by the user.

The measured data for system input and output have to exist in separate SIM-files. To guarantee correct results it is very important that input and output data were measured at the same *sampling points* so that both files for system input and output contain the same number of data pairs. IDA automatically tests the data files for correctness.



Identification of linear systems by IDA

The input file can be opened by the *File / Input signal x(t)...* menu option or the toolbar of the program. The output file is opened by *File / Output signal y(t)* or the toolbar. After reading both files IDA displays the data in its graphic window.

Next you have to specify the control parameters for the identification by the menu option *File / Control parameters...*or IDA's toolbar. The control parameter dialog includes the following data:

- the (maximum) order *m* of the numerator of $G(s)$,
- the (maximum) order *n* of the denominator of $G(s)$,
- the *identification mode*,
- the *time window* used for identification,
- the control parameters for the calculation of the systems *dead time*.

Normally the identification mode *Single* is selected. In this case the identification is executed for the numerator and denominator order selected in the control parameter dialog. In case of other identification modes the optimal transfer function is calculated for all combinations of numerator resp. denominator orders beginning with 0 up to the order selected in the control parameter dialog. So the optimal orders are found by IDA automatically. This kind of automatic identification can be cancelled by the user after each step.

If a system containing dead time is to be identified, the settings within the *dead time calculation* group box are of importance. These settings determine the discrete values for the dead time T_t the identification is executed for: $T_{t\ min}$ determines the lower bound, $T_{t\ max}$ the upper bound and *Intermediate steps* the number of intermediate steps.

After selection of control parameters the identification can be started by the *Approximation* menu option or the toolbar. The calculation time for the identification depends on the order of

the transfer function, on the number of measurement data and of course on your computers performance. Normally the results are available after a few seconds. If an identification mode with automatic adjustment of the numerator/denominator or with dead time calculation has been selected, the calculation time rises correspondingly.

After identification is complete the results are displayed graphically. Displayed are

- the measured input data (red dashed line),
- the measured output data (green dashed line),
- the output data based on the identified mathematical model (green line).

Sample files

`CHEN3_X.SIM, CHEN3_Y.SIM:`

Sample system with *m*=6 and *n*=8, can be approximated very well by a system with *m*=3, *n*=4

`EX1_X.SIM, EX1_Y.SIM:`

Sample system with *m*=1 and *n*=2, the input signal is an exponential function in this case

`N4_X.SIM, N4_Y.SIM:`

Sample system with all-pass behaviour (*m*=2 and *n*=4).

`RAUSCH_X.SIM, RAUSCH_Y.SIM:`

Sample system with noisy data (*m*=0 and *n*=2).

`SIN_X.SIM, SIN_Y.SIM:`

Sample system with a sinus input (*m*=0 uand *n*=2).

Linear system analysis by LISA

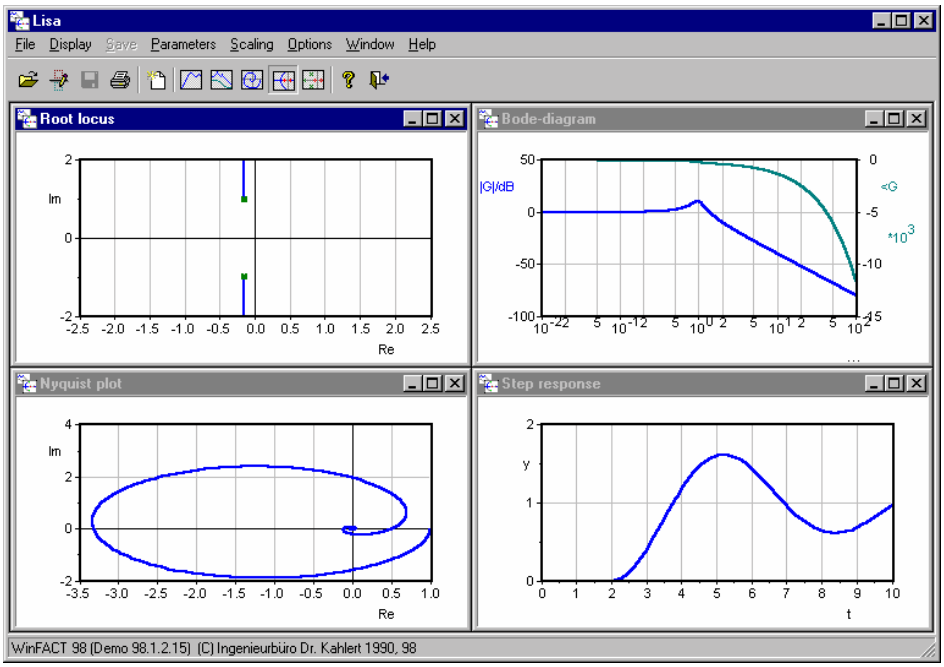
LISA enables you to analyse linear dynamic systems with input $u(t)$ and output $y(t)$ given by a rational transfer function with dead time

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} e^{-Ts}.$$

LISA offers the following features:

- Calculation of the systems step response
- Calculation of the systems frequency response as a Bode plot
- Calculation of the systems frequency response as a Nyquist plot
- Calculation of the systems root locus
- Calculation of the systems zeros and poles (eigen values)

LISA has the so called *multiple document interface* so that the same system can be displayed in different modes at one time, with different scalings etc.



LISA after opening some graphic windows

After opening a new display window the display mode *step response* is preset. This mode can be changed by the *Display* menu option or the toolbar. Step responses as well as Bode or Nyquist plots can be saved in specific WinFACT files of SIM, BD or OK type (not available in demo version!). For this purpose choose the *Save* submenu.

All display modes except the Pole/zero output offer a special zoom mode. This mode enables the enlargement of a user specified rectangular section of the graphic by the mouse. Simply push the left mouse button at the upper left corner of the rectangle you want to zoom and then while keeping the mouse button down zoom up the rectangle. After reaching the lower right corner release the mouse button and the selected area will be displayed again now filling the whole window. Internally this is realized by setting the scaling to manual mode. So to show the whole curves again just set the scaling mode back to automatic or push the corresponding toolbar button.

Sample files

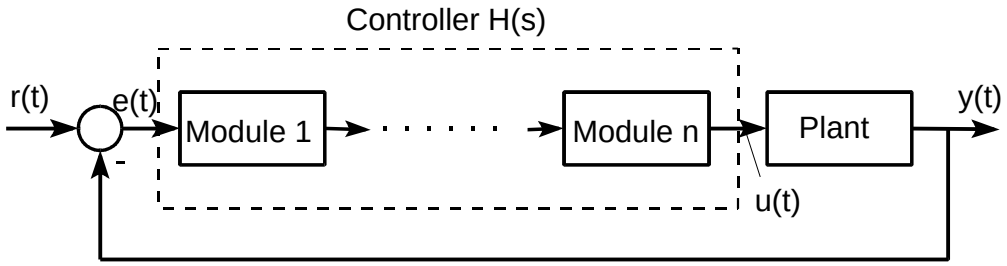
`LISA1.UFK`: System with m=1, n=2 and all-pass behaviour

`LISA2.UFK`: Same as `LISA1.UFK` with additional dead time

`LISA3.UFK`: System with m=6, n=8 and all-pass behaviour

Designing linear control systems by RESY

RESY enables you to analyse, design and simulate linear control systems with one feedback loop of the following structure:



The plant has to be given in form of a rational transfer function $G(s)$. The controller can be built up step by step by using the following standard modules:

- P-, I-, PI-, PD- and PID-components with transfer function

$$H_i(s) = K_R\left(1 + \frac{1}{T_N s} + \frac{T_V s}{1 + T_{Vz} s}\right)$$

- Lead-Lag-components with transfer function

$$H_{i,\mathrm{Lead}}(s) = \frac{1 + \dfrac{s}{\omega_i}}{1 + \dfrac{s}{m\omega_i}} \; bzw. H_{i,\mathrm{Lag}}(s) = \frac{1 + \dfrac{s}{m\omega_i}}{1 + \dfrac{s}{\omega_i}}$$

- General rational transfer function with dead time

$$H_i(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0} e^{-Ts}$$

Based on these data RESY calculates and displays

- the controllers summarized transfer function $H(s)$,
- the open loop systems transfer function $L(s) = G(s)H(s)$,
- the closed loop systems transfer function $T(s) = L(s)/(1 + L(s))$,
- the corresponding frequency responses $G(\mathrm{j}\omega)$, $H(\mathrm{j}\omega)$, $L(\mathrm{j}\omega)$, $T(\mathrm{j}\omega)$,

and in addition in time range for the case of a step function $r(t)$ the responses of

- the control error $e(t)$,
- the control output $u(t)$,
- the controlled variable $y_T(t)$

and the step response $y_G(t)$ of the plant itself.

In frequency range as well as in time range some characteristic values can be calculated that serve for the characterization of the closed loops dynamics. These values are in time range:

- The control variables overshoot M_p which gives the maximum value of the control variable during the step response.

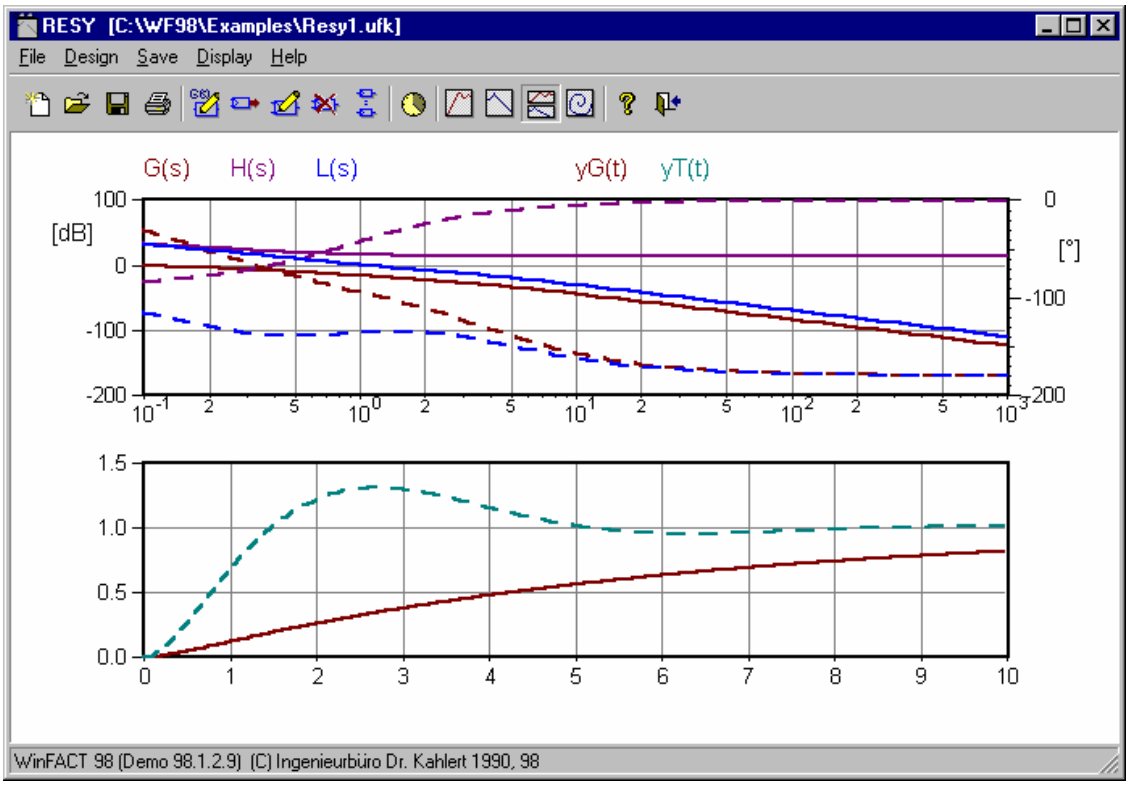
- The settling time T_a corresponding to a 10%-tolerance range around the stationary value of the controlled variable. This time serves as a measure for the systems dynamic.
- The stationary value for the control error $e(t \to \infty)$.
- The maximum of the control output u_{max}. This is the maximum value $u(t)$ taken during the step response.

The open loop criteria in frequency range are:

- The gain crossover frequency ω_c. This is the frequency where the gain of the open loop just crosses the 0 dB line.
- The phase margin Φ_r. This is the difference between the open loops phase and the -180° line at the gain crossover frequency. This value serves as a criterium for the closed loop systems stability.
- The amplitude margin A_r. This is the negative value of the open loops gain at that frequency where the phase crosses the -180° line.

The closed loop criteria in frequency range are:

- The band width ω_b. This is the frequency where the closed loops gain crosses the -3 dB line.
- The frequency overshoot M_m. This is the maximum value of the closed loop systems gain.



RESYs main window

The frequency response can be displayed alternatively as a Bode or Nyquist plot.

To configure the closed loop system first you have to define the plants transfer function. This can be done by manual input or by reading the data from an external UFK-file. Manual input

or later modification is selected by the *File / Edit plant* menu option or the corresponding toolbar button.

After definition of the plant the controller can be built up step by step. For that the submenu *Design* is used. It offers the following options:

- Inserting new controller modules,
- Deleting existing controller modules,
- Modification of existing controller modules,
- Listing all existing controller modules.

The selection of the controller modules that has to be deleted resp. modified is made by a corresponding list dialog.

Sample file

`RESY1.UFK`

Simulation and design systems in state space representation by SUSY

The WinFACT module SUSY helps you in analysis, design and simulation of linear dynamic systems in state space representation of the form

$$\dot{\underline{x}} = \underline{A}\underline{x} + \underline{b}u,\ y = \underline{c}^{\mathrm{T}}\,\underline{x} + du$$

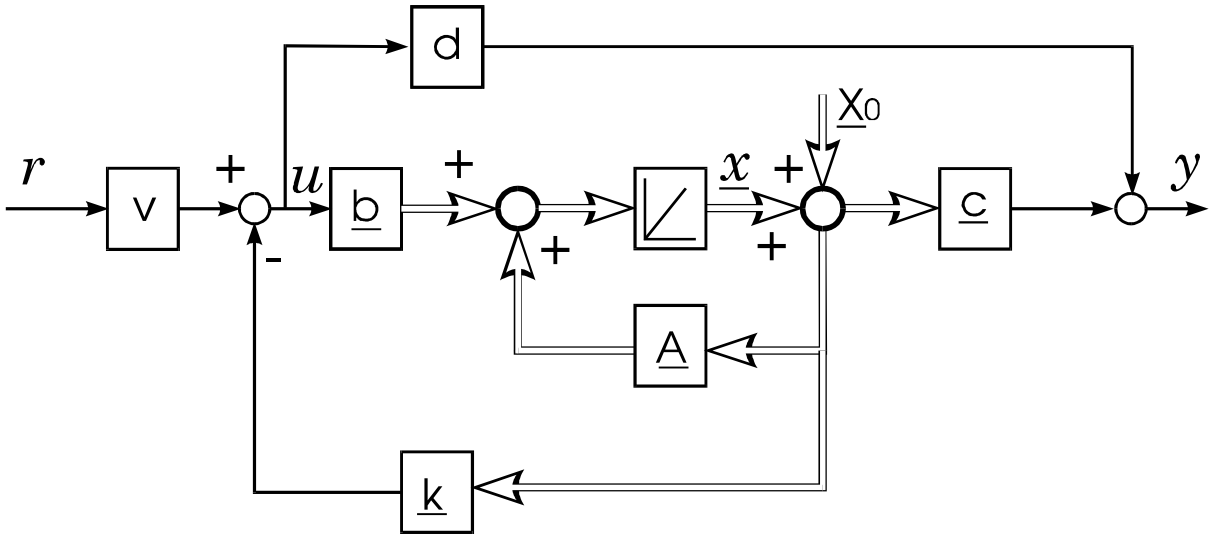
which can be completed to a closed loop system by an additional linear state controller

$$u = -\underline{k}^{\mathrm{T}}\,\underline{x} + Vr\,.$$

SUSY allows the analysis of systems up to 20th order. The main features of SUSY are:

- Comfortable and clearly arranged input and modification of system models
- Display of simulation results by single trajectories, trajectory fields or time responses
- Parallel handling of multiple systems
- Parallel handling of a system with different controllers (Piping)
- Quick change between open and closed loop system
- Calculation of open and closed loops eigenvalues
- Structured system output in matrix form
- Design of state controllers by manual, by pole placement or Riccati design
- ASCII-document generation of all interesting data

The basic structure of the closed loop system handled by SUSY is shown by the following figure.



Closed loop system with state controller

The variables have the following meanings:

$\underline{A}$ System matrix of the open loop system

$\underline{b}$ Input vector

$\underline{c}$ Output vector

d Direct influence of the input on the output

$\underline{x}_0$ Initial values of the state variables

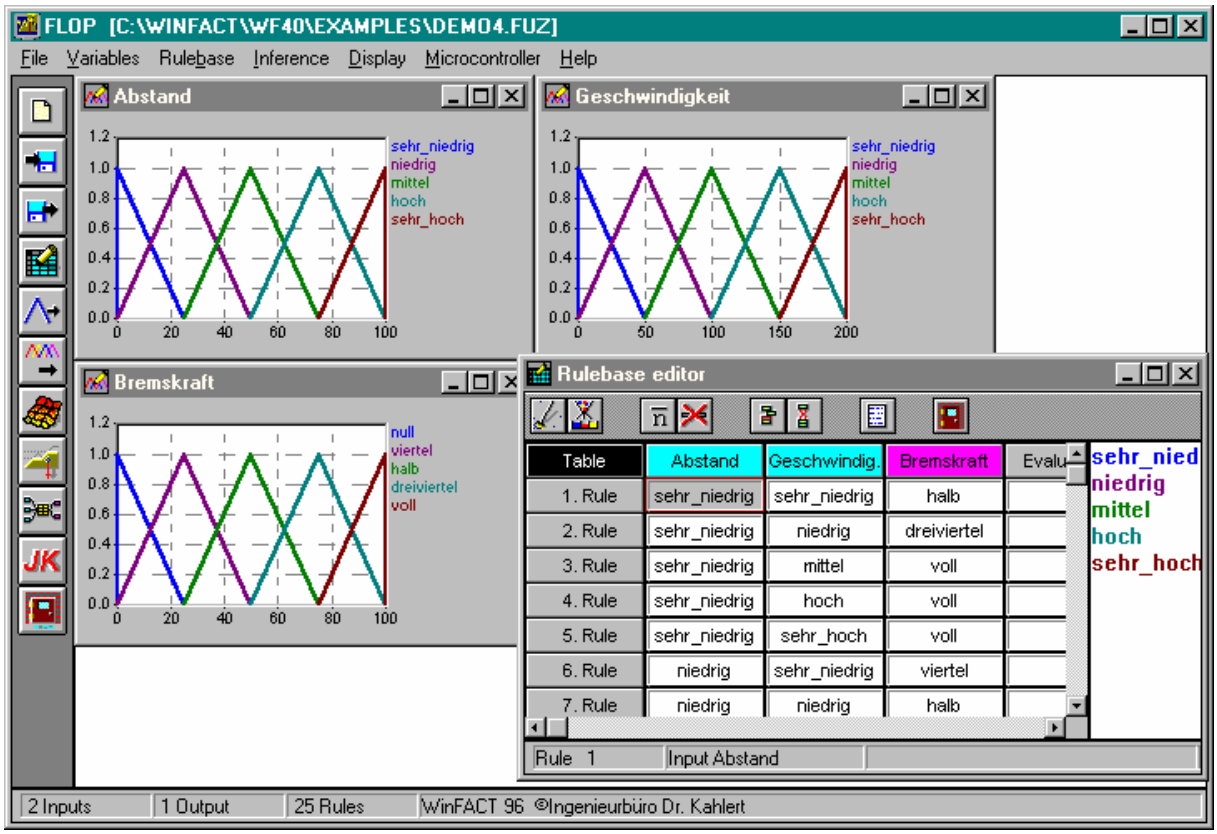
V Gain (Precompensator) to avoid a stationary control error

$\underline{k}$ Control vector (state controller)

The fuzzy shell FLOP

The fuzzy shell FLOP (**F**uzzy **L**ogic **O**perating **P**rogram) enables the design and analysis of rule based systems represented by fuzzy logic. The module offers the following features:

- Definition of linguistic variables and linguistic terms
- Operations on fuzzy sets
- Evaluation of membership values
- Construction of rule bases
- Evaluation of inference rules
- Calculation and graphical display of characteristic curves and 3D-transfer characteristics
- Simulation based on external data
- Programming of microcontroller boards (in combination with specific hardware)
- Design of fuzzy data files for the simulation system BORIS



Fuzzy shell FLOP with loaded sample file DEMO4.FUZ

Different types of operators, inference mechanisms and defuzzification strategies are available for all kinds of operations. The results are presented in a graphical way in general. For the representation of fuzzy sets three types of membership functions (triangular, trapezoid and singleton) are available.

Editing linguistic variables

For the clearness of the fuzzy shells user interface it is of great importance that all available data - the actual linguistic variables, their fuzzy sets and the rule base - are presented at each moment. Each linguistic variable is displayed in its own window which can be moved, enlarged, reduced, hidden or shown again. The first linguistic variable named *untitled* is defined automatically when the program is started. At first this variable however does not contain any linguistic term.

Each linguistic variable is represented by the following items:

- the *type* of the variable (input or output corresponding to premise or conclusion of a rule)
- its name (maximum number of characters: 15)
- its numerical *range*

A new linguistic variable is inserted by the menu option *New / New linguistic variable...* , the Strg V-keys or the corresponding toolbar button.

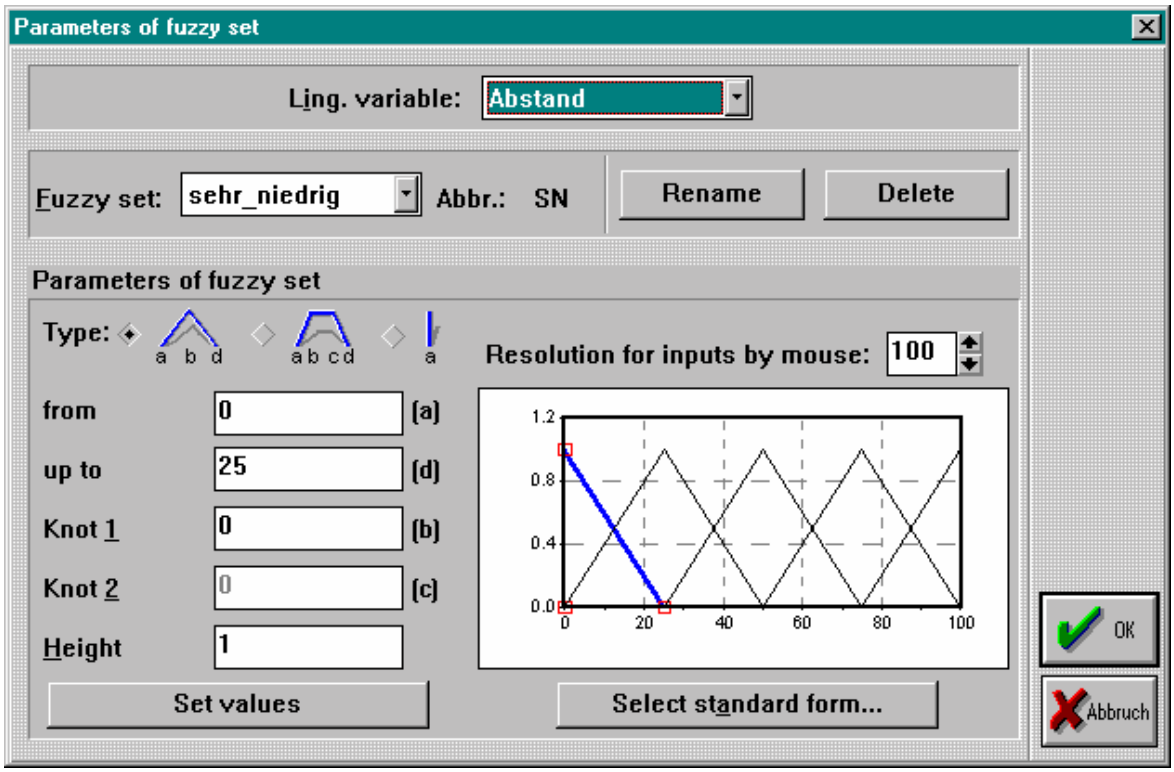
New linguistic terms (fuzzy sets) of a linguistic variable can be inserted by the menu option *Variable / New fuzzy set*, the Strg N-keys or the toolbar. The parameters of all existing fuzzy sets can be modified by the central fuzzy set dialog which can be reached in different ways:

- By the *Variables / Edit fuzzy set...* menu option.
- By pressing the Strg F-keys
- By a double click with the left mouse button within the corresponding variable display window. This is the most recommendable way, because the selected variable is preset within the dialog in this case.

Within this dialog the following actions are available:

- Modification of all membership functions of all linguistic variables
- Deleting and renaming of linguistic terms

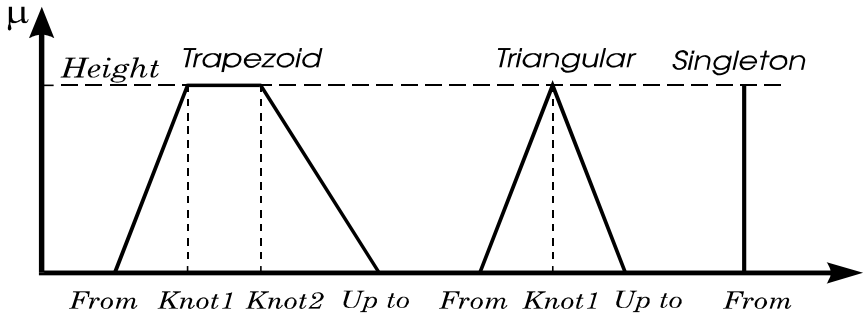
The actual linguistic variable and its actually selected membership function are set by list boxes at the top of the dialog. All linguistic terms of the actual variable are displayed graphically within the corresponding window. The selected membership function is highlighted by its blue color and red markers at its knots. All changes done by the users are recorded immediately. If the membership function type *Singleton* is selected, only the edit field named *From* has to be filled. The edit field *Knot 2* only is enabled in the case of *trapezoid* membership functions. By the *Factor* edit field all points of the membership function can be modified in one step by a multiplication with a constant value. A factor smaller one corresponds with a compression of all fuzzy sets in combination with a shift to the left while a value greater one corresponds with a expansion and a shift to the right. Using this factor may be efficient for variables which represent a control output to enlarge or reduce the controllers gain.



FLOPs fuzzy set dialog

Changes within the edit field *Height* lead to subnormal fuzzy sets. Normally these types of fuzzy sets should be used with care. All changes within the dialog (also changes of the membership function type) will be actualized not before pressing the *Set values* button.

The following figure presents an overview of the characteristic points of all types of membership functions.



Characteristic values of all types of membership functions

As an alternative to the numerical input of the characteristic values a mouse based modification of the fuzzy sets is available. To do so, first click at the desired fuzzy set with the left mouse button so that the fuzzy set will change to blue and red markers will appear at the characteristic points. Now you can move these points by clicking at them and moving them while keeping the mouse button pressed. The values within the corresponding edit fields are actualized automatically while the markers are moved. The resolution of the movement can be changed by the *Res.* edit field. Only changes of the fuzzy sets height have always to be made by numerical input.

Another option of great importance is the *Standard form* button. Pressing this button leads to a fast standard distribution of the fuzzy sets of the actually selected variable. This standard distribution includes symmetrical triangular fuzzy sets with full overlap.

Defining the rule base

The menu option *Rulebase / New...* leads to the rule base editor. The editor windows as well as the variable windows can be moved, enlarged, reduced or hidden at any time you want. It also is part of the list of all actual windows under the *Display* submenu of FLOP. The following graphic shows the rule base editor immediately after it has been opened for the first time. We see that the rule base is organized in table form where each line represents one rule. Columns for input variables (rule premises) are turqoise, columns with output variables (rule conclusions) are violet.

The definition and modification of rules are realized completely by mouse actions. Therefore the list box at the right margin of the editor contains all linguistic terms of the selected variable. By a double click with the left mouse button on a term this is taken into the rule table. The status line of the editor shows the item actually selected (number of the rule, name of the linguistic variable and the rule weight) and the number of all rules actually defined.

The *Clear up* button allows you to delete all invalid rules of the actual rule base. Invalid are those rules that do not contain an entry in at least one output variable. In addition to this a clear up action is carried out automatically always a file is saved.

If the weight of a rule has to be modified, first select the corresponding item within the last column of the rule base. After this at the right margin a scroll bar appears which enables you to modify the actual weight to any value between 0 and 1.

In addition to the modification of single entries of the rule base FLOP offers you powerful features for the parallel modification of groups of entries which are especially useful for the handling of complex fuzzy systems with a lot of rules:

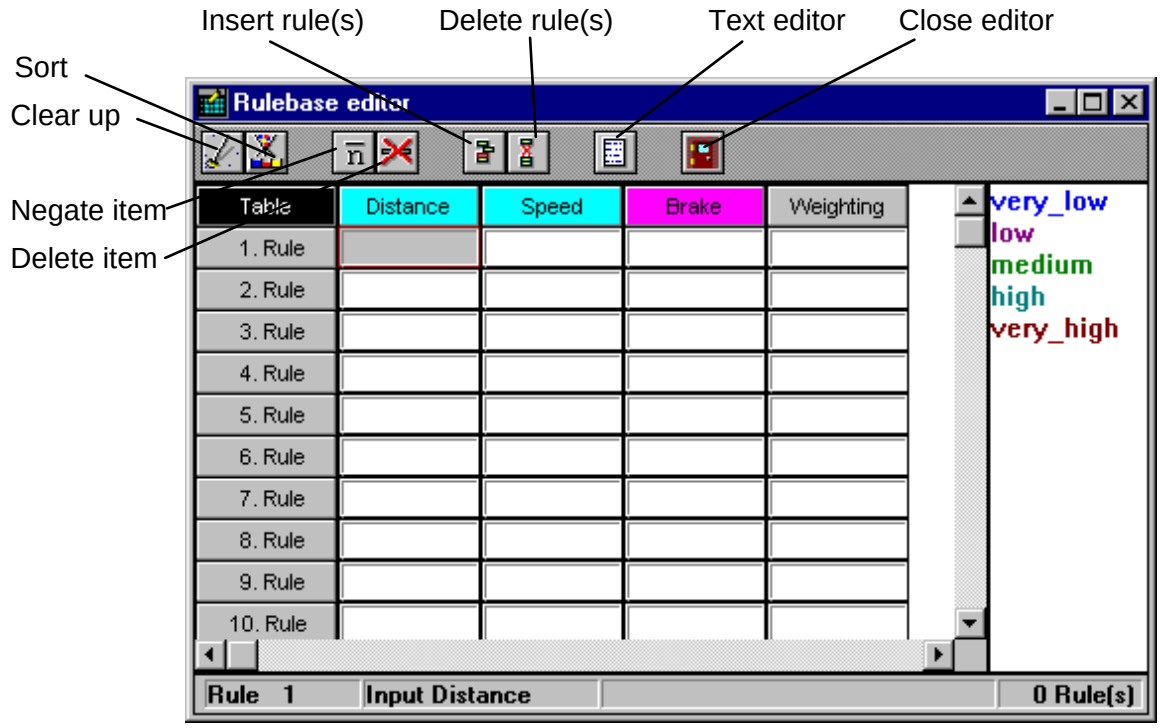
- By keeping the ⇧-key pressed when selecting items of the rule base several items lying one beneath the other can be selected at the same time
- By keeping the Strg-key pressed when selecting items of the rule base several items of the same column - which do not have to lie one beneath the other - can be selected at the same time.

In the following all items selected in this way can be deleted, filled by the same linguistic term or weight or moved to another location at the same time. If you want to copy the selected items, simply move them to the target position while keeping the left mouse button pressed. After reaching the target position release the mouse button again.

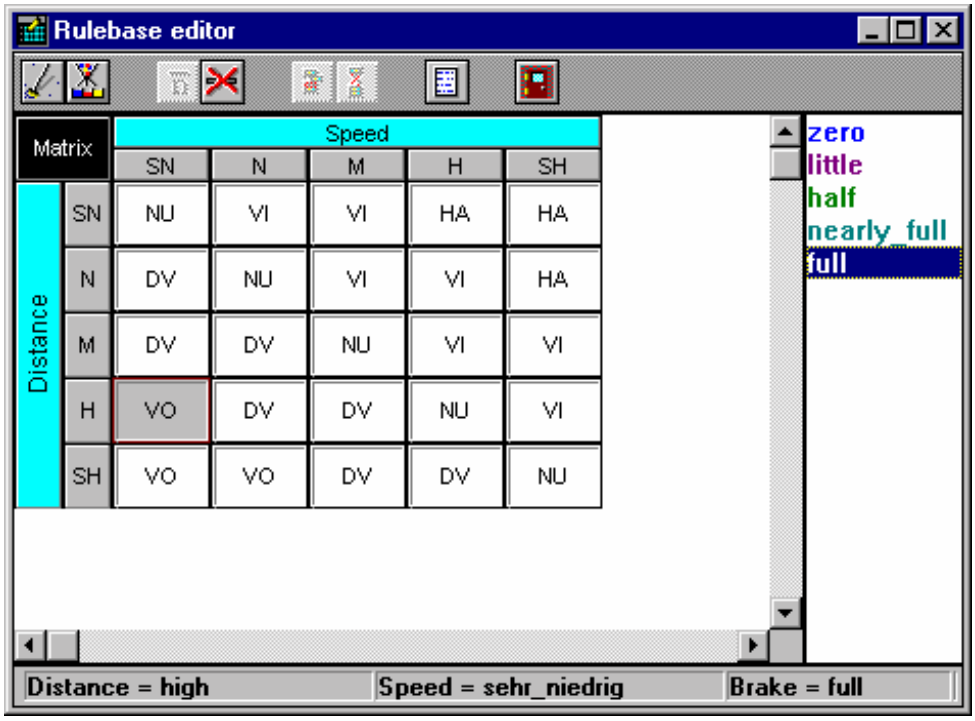
In the same way complete rules can be selected. For that simply select the corresponding items within the first colums of the rule base editor.

For fuzzy systems with two input variables and one output variable the rule base editor can also be represented in a *matrix form*. This form is especially efficient because you only have to fill the conclusion entry of each rule, not the premise entries. To switch between both table and matrix form, use the *Rulebase / Table form* resp. *Rulebase / Matrix form* menu option. The following graphic shows the rule base editor in matrix form filled with all rules.

If no specific rule is to define in matrix form, simply let the item empty. The extended editor options (multiple selection of entries, ...) are not available in matrix form.



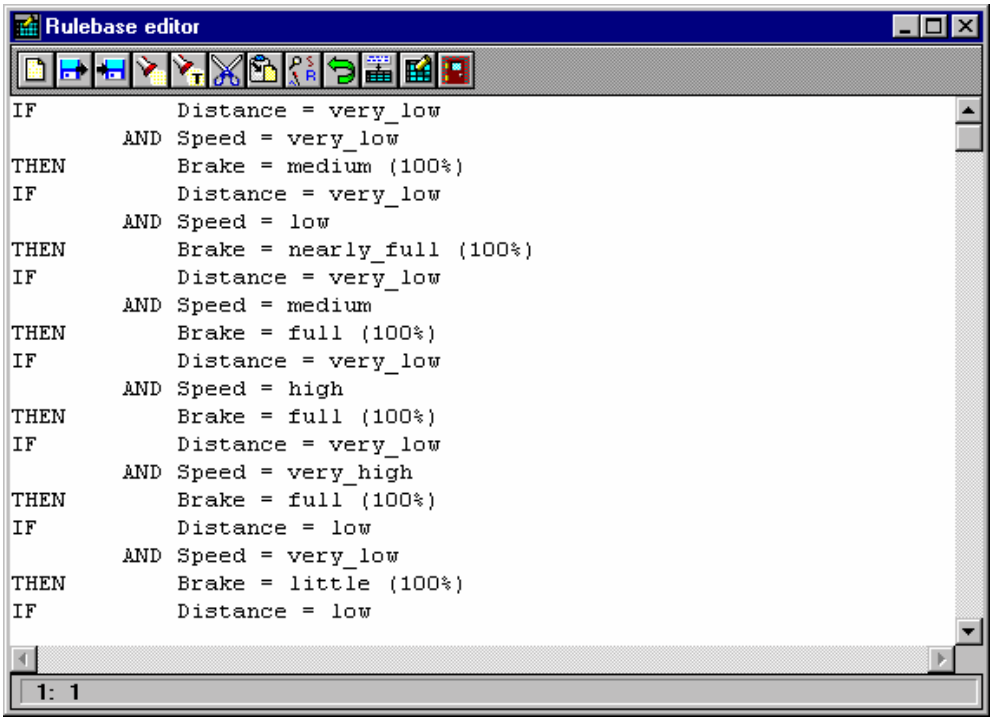
Functions of the rule base editor in table form



Rule base editor in matrix form

In matrix form as well as in table representation a text editor is at disposal which allows to formulate the rules in ASCII text form. The text editor is started by pressing the corresponding button of the editor. The following features are available:

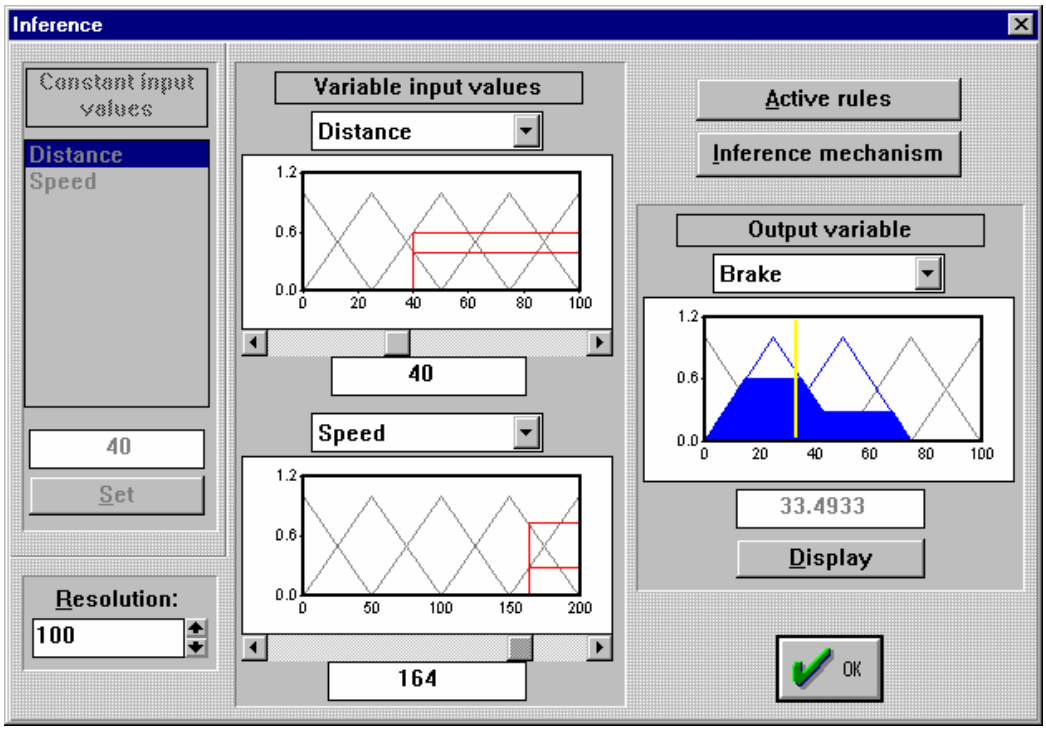
- Loading and saving of text files (ASCII format; default extension is RB). If a new file is loaded, it will be appended at the end of the text already within the edit window.
- All standard editor options as Search, search and replace, cut and paste etc.



Rule base editor in text form

Inference

After having defined the rule base the sub menu *Inference* is now enabled. Choosing the submenu option *Single step* leads us to the inference dialog. This dialog enables us to perform single inference steps, i. e. to evaluate crisp output values for given input values by fuzzification, inference and defuzzification. The following screen shot shows the evaluation of crisp values of the output variable *Brake* for given values of the input variables *Distance* and *Speed*.



Inference dialog in case of two input variables and one output variable

The main components of the inference dialog are:

- The area for the input of constant values (left margin)

  Only two input variables can be varied interactive at the same time. If more than two input variables exist, the values for the other inputs have to be fixed. For that select the input variable within the list box at the left margin of the dialog, set the value of the input within the edit field below the list box and press the *Set* button. Repeat this proceeding for the other variables.

  If only two input variables exist, this part of the dialog is disabled.

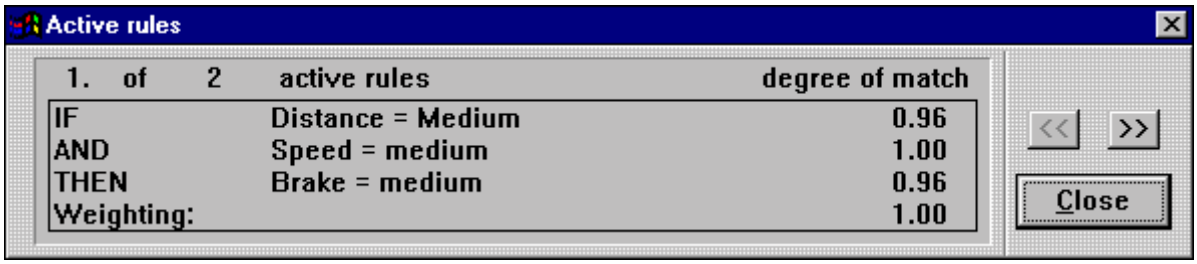
- The display windows for the inputs (in the middle of the dialog)

  The input variables that shall be varied interactive can be selected by the list boxes at the top of the windows. The actual values for these variables can be edit within the corresponding edit field below the window (and pressing of the *Display* button) or modified by using the scroll bars directly below the windows. The resolution for the scroll bars can be modified by the *Resolution* edit field. If only one input variable exists, the lower window is disabled.

- The display window for the output variable (right margin).

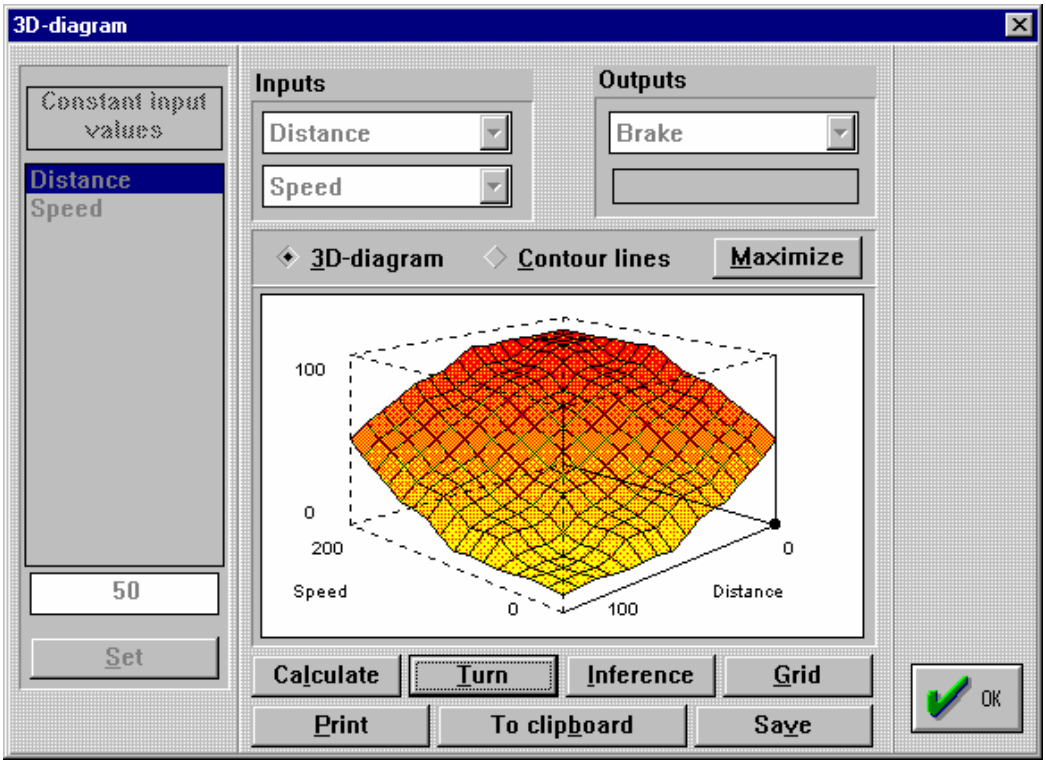
This display window shows the resulting output fuzzy sets and the crisp output value evaluated from that graphically. In addition to that the crisp output value is displayed numerically within the edit field below the window. If more than one output variable exist, the requested variable can be selected by the list box at the top of the window.

For each inference step all active rules together with their degree of match can be displayed by pressing the *Active rules* button. In the following a non modal window appears above the inference dialog. This window can be moved and is updated after each calculation as long as it is visible. If more than one rule is active the buttons >> resp. << enable you to scroll within all active rules.



Active rules

The transfer characteristic of a fuzzy system can be displayed in form of a characteristic curve or a 3D graphic resp. field of contour lines. For this purpose use the *Inference / Transfer characteristic*... menu option. If only one input variable exists, the characteristic curve will be displayed while in the case of two inputs a three dimensional graphic will be shown. For fuzzy systems with more than two inputs you can select two variables for the graphical representation by list boxes while for the other inputs numerical values have to be set (see also single-step-inference).



3D transfer characteristic

Alternatively to the single step or 3D graphic mode the transfer characteristic of the fuzzy system can be analyzed by a simulation. In this case the input data are read from an external file and the corresponding output data are calculated simultaneously. To do so select the *Inference / Simulation* menu option. The input file must have the extension FSI (see later). For the simulation procedure a dialog appears which offers the following features:

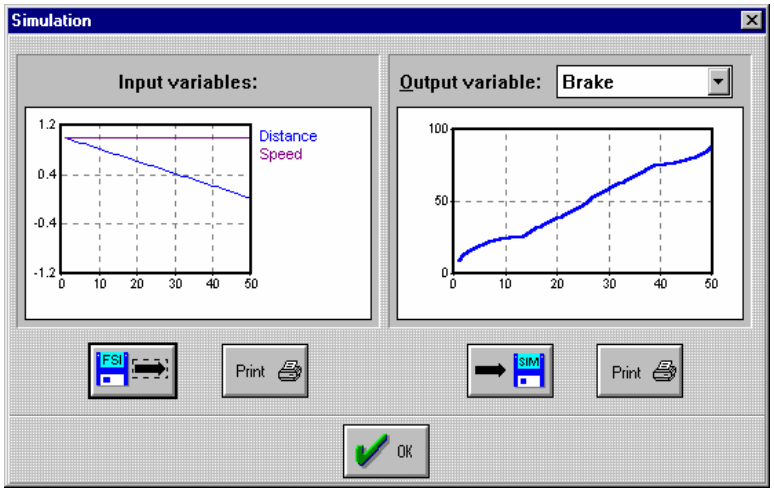
- The loading of the input data (button at the left). The input data read in from the input file are plotted within the left graphic window. All input data are scaled to their maximum value by FLOP so that all curves can be plotted in one diagram with a scaling range of [-1, 1].
- The printing of the graphic windows contents (buttons with printer symbol).
- The storing of the calculated output values in a file of SIM type (button on the right).

The input file (FSI-file) must contain all inputs for one sample point within one line, separated by one or more blanks. A parametrization of the time is not needed. In this way for a fuzzy system with two inputs each lines contains two values. The following table shows an example for such a two input system.

```
1       10
2       10
3       10
4       10
5       10
```

Example for a FSI-simulation file

In this example the transfer characteristic of the fuzzy system is simulated for the case that the first input rises linear from 1 to 5 while the second input has a constant value of 10 (see next screen shot).



FLOPs simulation dialog

Sample files

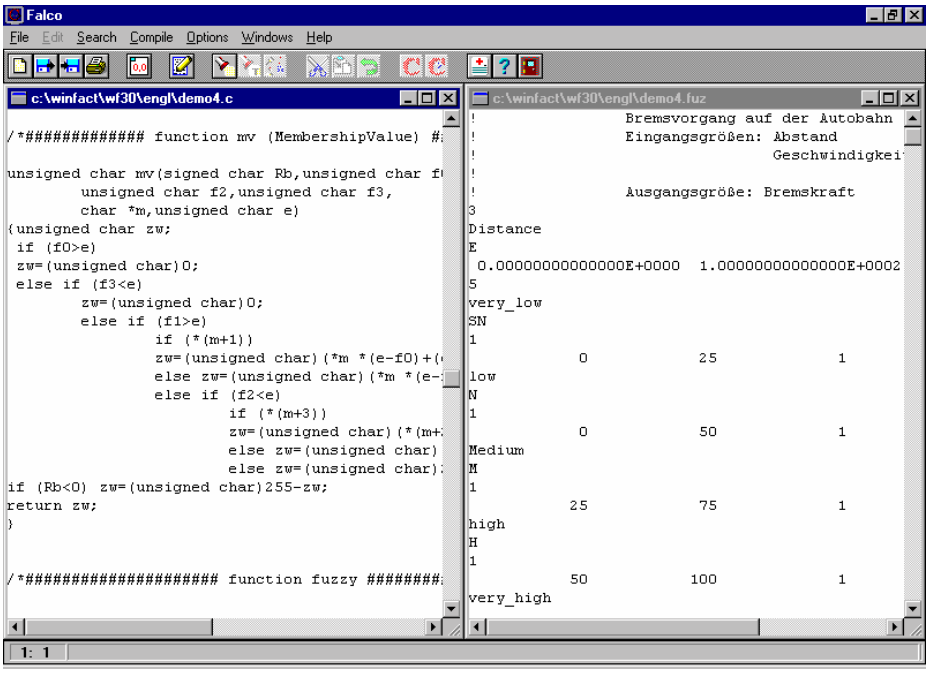
DEMO1.FUZ
DEMO2.FUZ
DEMO3.FUZ

DEMO4.FUZ (in combination with DEMO4.FSI)
DEMO7.FUZ

Generating Fuzzy-C-Code by FALCO

The C-Code-Generator FALCO (**F**uzzy **A**ppl**i**cation C-**Co**degenerator) enables you to generate ANSI-C-Code for fuzzy systems designed by the fuzzy shell FLOP. The C-Code mainly contains a C function which can be included in user written software later or used for the programming of user specific hardware. FALCO offers the following features:

- Comfortable editor for FUZ- and C-files with Multi-Document-Interface containing all standard editor options as copy and paste, search and replace etc.
- Parallel editing of multiple systems
- The code generated by FALCO only consists of one file - no further library is necessary. The code is very compact and contains only the instructions, operations and variables needed for the specific system.
- The C-types for the linguistic variables (inputs and outputs of the fuzzy system) and the membership values can be selected by the user (short int, long int, float, ...)
- On request a main function can be generated for the debugging of the generated code
- Compiling, linking and running of the generated code directly from the FALCO environment; selection between different compilers and/or linkers possible
- Command line calling up FALCO from the FLOP environment



Generating C code with FALCO

The fuzzy system to be compiled to C code must exist as a FUZ-file like generated by the fuzzy shell FLOP. This file can be opened and loaded into the active editor window by the *File / Open* menu option. A possibly already existing file within this window is overwritten. Shall the file be loaded into a new window, this window first has to be created by the *File / New* menu option. The contents of an editor window can be deleted completely by the *Edit / Delete all* command. By the *File / Save* command the contents of the active editor window is saved, by the *File / Print* command it is printed on the standard printer.

After loading a FUZ-file the file is checked for errors first. This error check concerns the file syntax on one hand and plausibility checks (conformity of parameter combinations etc.) on the other. The error check is of special importance if the FUZ file was created or modified manually. If errors are found a corresponding error message is displayed and the cursor is automatically moved to the place where the error was found.

After some settings by the *Options / Code-generator* command the code generation itself can be started. The code always is generated for the FUZ-file within the active window. Two ways of code generation are possible:

- Generating only the C function to describe the fuzzy system itself (no main function). This mode is started by the *Compile / Generate C-code* command.
- Additional generation of a main function for the manual debugging of the fuzzy code. This main function enables the user to input the crisp values for the input variables by the keyboard and to display the values calculated for the systems output variables on the screen. This mode is started by the *Compile / Generate C-test-code* command.

After the code generation is completed a separate editor window is created automatically which contains the generated code. The name of the C file is the same as that of the corresponding FUZ file but with extension C.

The code mainly consists of a function called *fuzzy*. Input parameters of this function are the crisp inputs of the fuzzy system, output parameters the crisp outputs. The function declaration for a fuzzy system with *m* inputs and *n* outputs is of the following form:

```
void fuzzy(char *changed,
           Inputtyp1 Input1, ...,
           Inputtypm Inputm,
           Outputtyp1 *Output1, ...,
           Outputtypm *Outputm)
```

The types of the input and output parameters depend on the data types of the linguistic variables selected by the user before the code generation was started. The names of the input and output parameters correspond to the names of the linguistic variables within the FUZ file extended by a characteristic number and the string *Value*.

If a C compiler is located on the PC the generated C code can be compiled directly from the FALCO environment and - if test code was generated - started automatically. To compile the code use the *Compile / Generate test compilat* resp. *Compile / Generate target hardware-compilat* commands. By these menu options the command lines specified by the *Options / Generate code* menu option are executed. If an EXE-file is generated this can be started immediately after the compilation by the *Compile / Execute* command.

Designing Fuzzy-PID-Controller by FuzzyPID

The module FuzzyPID offers an interactive design of a Fuzzy-PI- or Fuzzy-PD-Controller for a closed loop system with single feedback. The plant must be of linear type. The closed loop system can be simulated with a simultaneous graphical output of all interesting variables. Therefore FuzzyPID is especially suitable for first steps in the area of fuzzy control as well as for demonstrations and representations in education and research. More complex fuzzy controller should be designed by the fuzzy shell FLOP and the simulation module BORIS.

The concept and handling of FuzzyPID is nearly identical to the fuzzy shell FLOP. In addition the following restrictions have to be kept:

- The linguistic variables have the names

  e for the control error *e*,

  de/dt for the derivation $\dot{e}$ of the control error,

  u for the control output *u* (Fuzzy-PD-Controller),

  du/dt for the derivation $\dot{u}$ of the control output (Fuzzy-PI-Controller).

  These names are declared internally and cannot be changed by the user. Also a deletion of linguistic variables is not allowed.

- The number of linguistic terms per linguistic variable is internally set to five. This number cannot be changed. The terms have the same names for all variables. These are

Negative_Big	(Abbrevation --)
Negative_Small	(Abbrevation -)
Zero	(Abbrevation 0)
Positive_Small	(Abbrevation +)
Positive_Big	(Abbrevation ++).

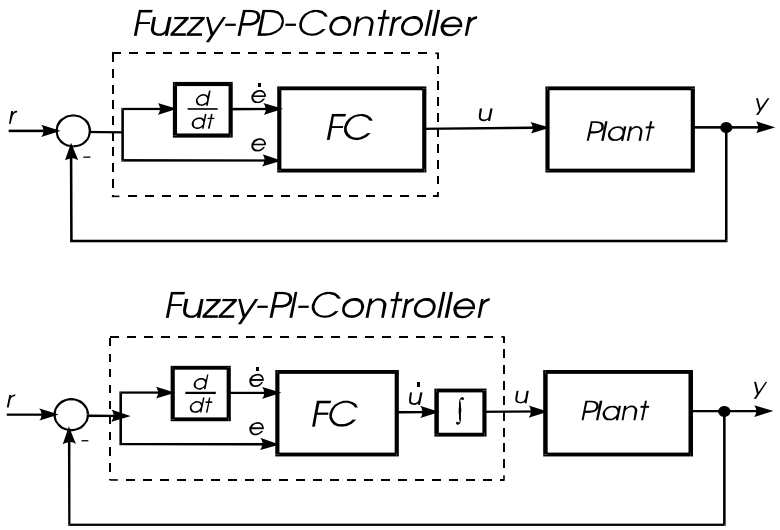
  These names and abbrevations cannot be changed.

- The controller and thus the rule base must have two inputs, e and $\dot{e}$. If a Fuzzy-P-Controller has to be realized, first select the Fuzzy-PD-Controller type and then make the generated output independent of $\dot{e}$. This can be achieved by choosing the same linguistic term for each entry in a rule base line.

- All rules have the weight of one. Negated premises are not allowed.

All relevant information is presented within the FuzzyPID program window simultaneously. These are:

- The structure of the underlying closed loop system,
- the membership function (fuzzy sets) of the controllers inputs and outputs,
- the actual rule base,
- the simulation results.



Closed loop system with Fuzzy-PD-Controller (top) resp. Fuzzy-PI-Controller (bottom)

The underlying closed loop system has a single feedback of the controlled variable. For first experiments in the area of fuzzy control the plant model can be selected from five predefined models. The following models are available by the *Control loop / Plant* command:

- PT_2-Plant (non oscillating)

  This type of plant has a gain of one und two real eigen values. The corresponding time constants are $T_1 \approx 0.2$ s and $T_2 \approx 5$ s. The corresponding transfer function of the plant is

$$G(s) = \frac{1}{s^2 + 6s + 1}$$

- PT_2-Plant (oscillating)

  This plant has two complex eigen values and therefore is able to oscillate. The corresponding damping of the plant is $\zeta = 0.25$, the eigen frequency is $\omega_n = 1$. The corresponding transfer function is

$$G(s) = \frac{1}{s^2 + 0.5s + 1}$$

- PT_1-I-Plant

  This plant is a serial combination of a PT_1-Element with a time constant of $T = 10$ s and an integrator. The corresponding transfer function is given by

$$G(s) = \frac{0.2}{s(1 + 10s)}$$

- PT_1-T_t-Plant

  This plant is a serial combination of a PT_1-Element and a dead time. The PT_1-Element has a time constant of $T = 5$ s. The dead time is $T_t = 3$ s. Therefore the corresponding transfer function is given by

$$G(s) = \frac{1}{1 + 5s} e^{-3s}$$

- PT_2- Plant (time variant)

  This type of plant has the same structure as the oscillating PT_2-Plant, but combined with a time variant gain. The transfer function is given by

$$G(s) = \frac{K(t)}{s^2 + 0.5s + 1}.$$

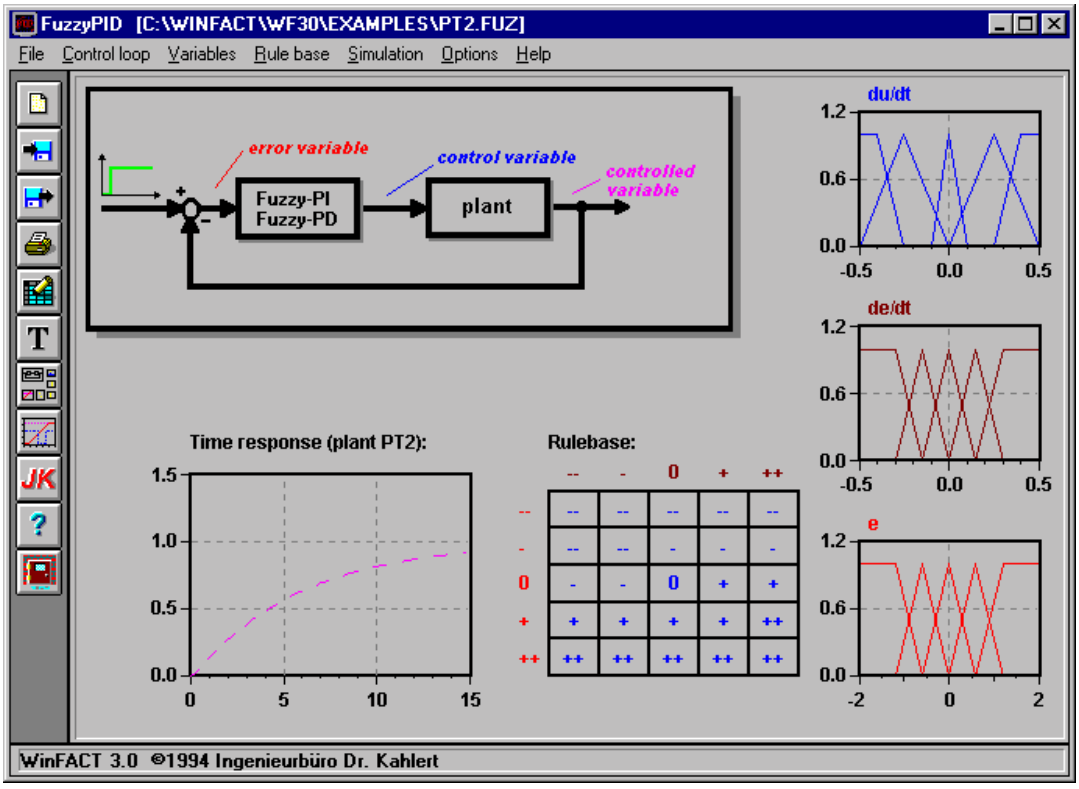
  For $0 \le t \le 5$ the gain increases linear from 1 up to 2 and then decreases again linear to 1 for $5 \le t \le 10$:

$$K(t) = \begin{cases} 1 + t/5 & for \quad 0 \le t \le 5 \\ 2 - (t-5)/5 & for \quad 5 < t \le 10 \\ 1 & for \quad t > 10 \end{cases}$$

For all these plants a modification of parameters is not allowed.

In addition to the selection of such a predefined plant the last three options within the *Closed loop / Plant* submenu allow the input of a user defined linear plant in form of a rational transfer function. This transfer function can be read by keyboard or an external file of UFK-type as well as stored in a UFK-file after modification.

Available controller types are Fuzzy-PI- and Fuzzy-PD-Controller. Inputs of both controller types are the control error *e* and its derivative $\dot{e}$, the output is the control output *u* (Fuzzy-PD-Controller) resp. its derivative $\dot{u}$ (Fuzzy-PI-Controller). The controller type is selected by the *Closed loop / Controller type PI* resp. *Closed loop / Controller type PD* command. By the *Closed loop / Simulation parameters*...menu option or the Strg P-keys an input dialog for the simulation parameters can be called up.



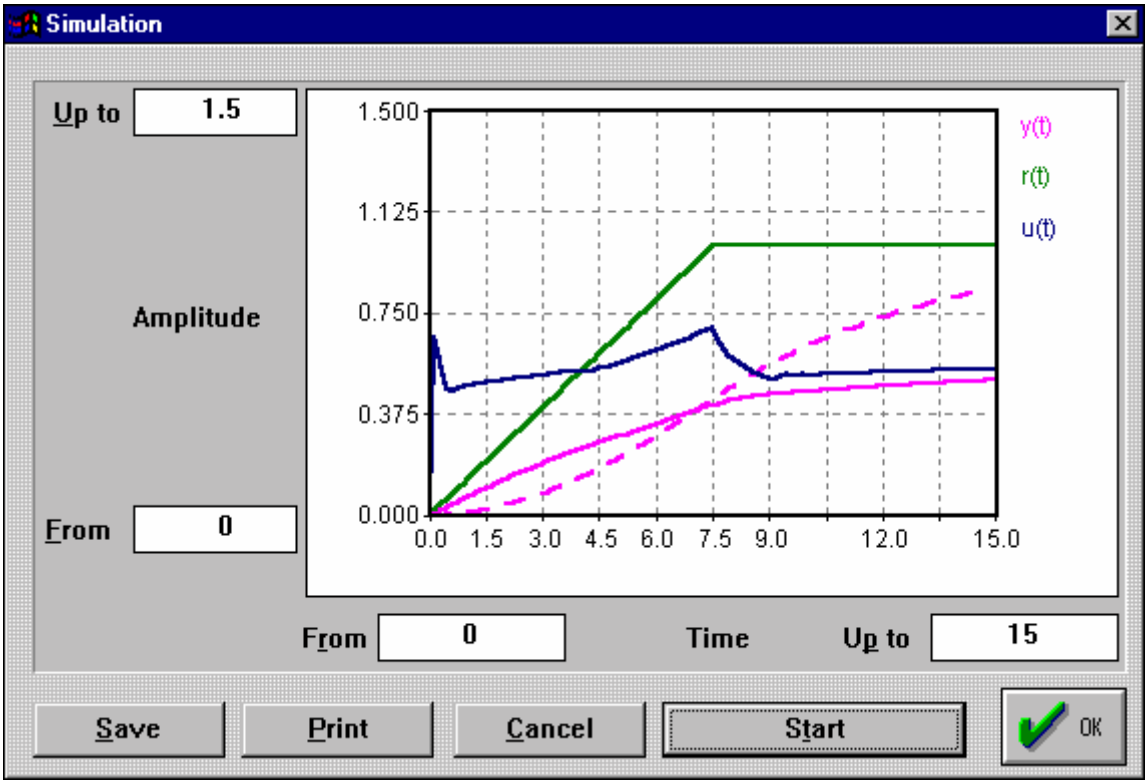
The FuzzyPID main window

After all membership functions and the rule base are read by keyboard or an external file the simulation can be started. For this purpose two different modes of simulation are available. By the *Simulation / Display all* or the Strg S-keys the first simulation mode is started. In this mode the complete dynamic behaviour of the fuzzy system is represented graphically. The following information is shown online during the simulation:

- Within the diagrams with the membership functions of the controllers in- and outputs (right margin of the program window) the actual values are shown by yellow line cursor. The diagram for the controllers output u resp. $\dot{u}$ at the top additionally shows the active output fuzzy sets and their degree of match (hatched blue).
- Within the rule base the active rules are changing their colour to yellow.
- The diagram in the lower left corner shows the time responses of all those variables of the control system that were specified by the *Options* command of the programs main menu:
  - The time response of the plant itself, e. g. open loop without controller (dashed violet curve),
  - the progress of the reference variable $r(t)$ of the closed loop (full green curve),
  - the manipulated variable $y(t)$ (full violet curve),
  - the error variable $e(t)$ (full lightred curve),
  - the derivative $\dot{e}(t)$ of the error variable (full red curve),
  - the control variable $u(t)$ (full darkblue curve),
  - the derivative $\dot{u}(t)$ of the control variable (full lightblue curve). This curve can only be shown in the case of an PI-Controller.

By pressing the *Cancel* button during the simulation the simulation can be stopped at any time.

If only the time response of the system is of interest, the second simulation mode is more recommendable because it runs much faster. This mode can be started by the *Simulation / Time response* command or the Strg V-keys. This action leads to a simulation dialog shown by the next screen shot. This dialog offers a larger part for the display of the time responses and the possibility to scale the axes of the diagram individually. The time response of the plant is shown immediately after the dialog is called up; the simulation of the other variables is started by the *Start* button. It can be stopped at any time by pressing the *Cancel* button.



Simulation dialog of FuzzyPID

Sample files

The sample file collection includes one sample file for each type of plant that works more or less well and can be used as a base for own experiments on analyzing or improving the controller. The files have the following names:

`PT2.FUZ`	Fuzzy-PI-Controller for PT_2-Plant (non oscillating),
`PT2S.FUZ`	Fuzzy-PI-Controller for PT_2-Plant (oscillating),
`PT1I.FUZ`	Fuzzy-PD-Controller for PT_1-I-Plant,
`PT1TT.FUZ`	Fuzzy-PI-Controller for PT_1-T_t-Plant,
`PT2ZV.FUZ`	Fuzzy-PI-Controller for PT_2-Plant (time variant).

The corresponding controller type (PI resp. PD) is recognized and set by FuzzyPID automatically on the base of the variable names found within the file.

Simulating dynamic systems with BORIS

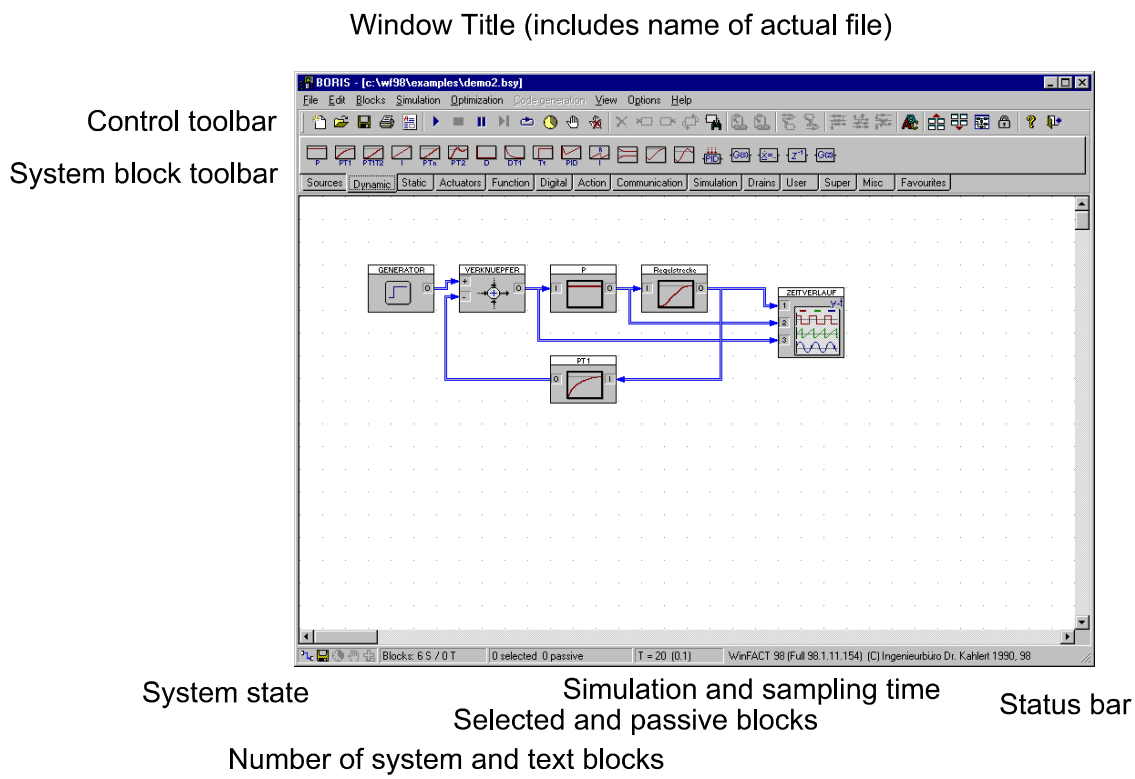
Survey

The **b**lock-**ori**entated **s**imulation system BORIS is a powerful tool for the simulation of all kinds of dynamic systems with or without fuzzy components. BORIS offers the following features:

- Extensive system block library:
  - Signal generators (Sinus, Pulse, Noise, test functions)
  - Linear standard elements (PT_1, PT_2, ...)
  - Linear systems of higher order
  - Nonlinear characteristics and functions, algebraic functions
  - Linear and nonlinear standard controllers, adaptive controllers
  - Fuzzy Controller with integrated Fuzzy Debugger
  - Virtual instruments (y-t-plot, x-y-plot, analog and digital display, oscilloscope, bargraph, status display)
  - File inputs and outputs
  - Spectral analysis by Fast Fourier Transformation
  - Statistical functions
  - Digital modules (logic gates, Flip Flops)
- Definition of hierarchical macros (Superblocks)
- Placement of system blocks flexible; extended scrollable workplace
- Different numerical integration methods

Components of the BORIS main window

The following screen shot shows the main window of BORIS with its components.



In addition to the WINDOWS standard components the window contains the following elements:

- A first horizontal toolbar (control toolbar) below the menu. This toolbar contains buttons for the most frequently used commands. Choose the menu item *Options/Toolbars/System toolbar visible* to activate or deactivate the toolbar.

- A second horizontal toolbar (system block toolbar) which is subdivided into several palettes. This toolbar allows the direct access to all system blocks. The palette *Favorites* can be configured by the user (see c*onfiguration of the system block toolbar*). By *Options/Toolbars/System block toolbar visible* the toolbar can be activated or deactivated.
- A status bar at the bottom of the window. This line contains informations about the actual number of system blocks and the number of selected resp. set passive block. Additionally the simulation parameters are displayed in the following form T = T_{Simu} (ΔT). In this term T_{Simu} is the simulation time and ΔT the simulation step size.

While the simulation is running the progress of the simulation is displayed unless this option had not been deactivated. Five icons at the left margin of the status bar inform about the actual state of the system:


autorouter activated


system has not been saved since the last change


simulation is running


breakpoint activated


optimization is running

- The paint window (workplace) for the design of the system structure. This window can be scrolled by its scrollbars in horizontal and vertical direction. After starting BORIS the clipping area is set to its origin. This initial state can be reset at any time by the *Options / Reset scrollers to origin* command.

  The paint window has a grid by default to which all blocks are adjusted with their upper left corner. This automatical adjustment can be deactivated by the *Options / Snap to grid* command. Further the grid itself can be deactivated by the *Options / Show grid* command indepent of the status of the block adjustment.

Inserting and editing system blocks

The following features concerning the block handling are available:

- *Inserting* blocks of the system block library
- *Selecting* single blocks or groups of blocks
- *Moving* single blocks or groups of blocks
- *Deleting* single blocks or groups of blocks
- *Turning* blocks
- *Copying* and *pasting* of blocks
- Setting the *parameters* of a block
- Changing the *block size*

Most options are available by a context pop-up menu which opens when clicking with the right mouse button.

Inserting blocks

To insert a new block to the actual system structure

- click the corresponding button of the system block toolbar or
- select the corresponding menu item of the *Blocks* submenu.

The inserted block appears in the upper left corner of the paint window or - if there is not enough space - a little bit moved to the right or the bottom. Alternatively it can be inserted by Drag & Drop.

Selecting blocks

Before any operation concerning a block or a group of blocks can be executed, the blocks have to be selected first. To select a single block or a group of blocks there are different ways:

- A single block is selected by clicking at it with the left mouse button. If another block is already selected at this time, the selection of that block is deactivated.
- If another block is to select in addition to already selected blocks, click at it with the Shift- or Strg-key pressed. Clicking at an already selected block takes the selection back again.
- Alternatively a group of blocks can be selected by drawing a rectangle with the mouse. This can be realized by moving the mouse within the paint window while keeping the left button pressed. All blocks lying completely within this rectangle are selected.
- By the *Edit / Select all* or the Strg A-key all existing blocks are selected simultaneously.

Clicking at any free location within the paint window with the left mouse button takes the selection of all selected blocks back.

Moving blocks

To move a single block or a group of blocks proceed as follows:

1. First select the block resp. the group of blocks to be moved.
2. With the left mouse button click within the block to be moved resp. - if a group of blocks is to move - within any of the blocks of the group and move the mouse while keeping the left button pressed. While moving the cursor takes the form of a cross. If the margin of the paint window is reached the window is scrolled automatically.

After the moving is finished all blocks are adjusted automatically so that no blocks overlap.

Moving the complete system

During the design of the system structure it often happens that new blocks are to be inserted at the left or top of the structure already existing. Therefore BORIS allows to move the complete system structure (all system and text blocks, all connections) in any direction without selecting all blocks before. The corresponding commands are:

Options / Move right

Options / Move left

Options / Move down

Options / Move up

Each command moves the system by one grid unit to the corresponding direction.

Setting blocks passive

Blocks can be set passive. During the simulation process passive blocks are handled as if they were not existent.

Deleting blocks

A single block or a group of blocks selected before can be deleted in two ways:

- By the pop-up menu (right mouse button)
- By pressing the corresponding button of the control toolbar (more recommendable).

All input and output connections of the deleted blocks are deleted automatically with their blocks.

How to rotate blocks

The orientation of a block can be turned by 180° so that the input(s) of the block are on the right side (for example for blocks in a feedback). For this purpose use the *Edit / Turn block* command. All connections of the block are re-routed automatically.

Copy and paste

The copy and paste function of BORIS is controlled by a temporary file named BOCOPY.TMP. All selected blocks, their parameters and the connections between the selected blocks are copied resp. pasted. Two different cases are possible:

If you want to copy a part of the structure *within the same system*, proceed as follows:

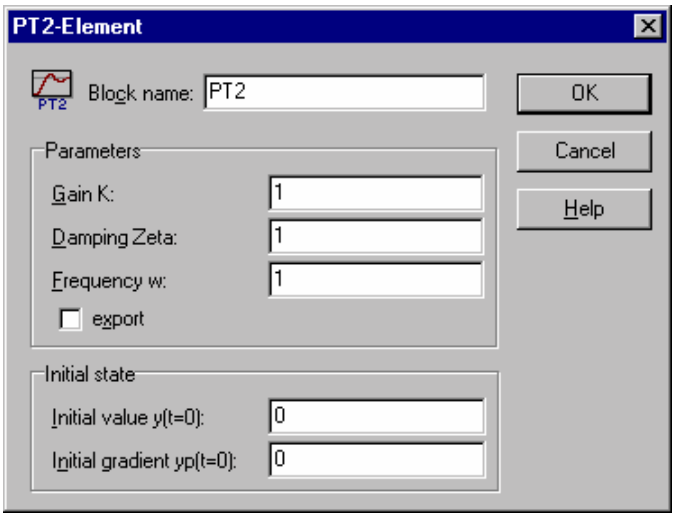
- Select the blocks to be copied and then select the *Edit / Copy* menu option or press the corresponding button of the control toolbar. To insert the blocks again use the *Edit / Paste* command or the corresponding toolbar button.

To copy blocks *between different systems* proceed as follows:

- First load the system where you want to copy from, select the blocks and use the *Edit / Copy* command. Now load the system you want to copy to (do not close BORIS before, because BORIS deletes the BOCOPY.TMP file if you quit the program!) and use the *Edit / Paste* command. Alternatively of course it is possible to start BORIS twice, load the two systems simultaneously and copy between both BORIS applications.

How blocks get their parameters

The most simple way to modify the parameters of a block is to doubleclick it with the left mouse button. Alternatively the block can be selected first and then the *Edit / Edit block* command be executed. Both ways lead to a block specific parameter dialog where the parameter modifications can be made. The following screen shot as an example shows the parameter dialog of an oscillating PT_2-Element.



Parameter dialog for oscillating PT_2-Element

For all block types the parameter dialog contains an edit field at its top where the name of the block can be modified. This name is set into the caption of the block window within the paint window. By default this name is identical to the block type name. The maximum length of the name is 25 characters. In addition to that all parameter dialogs contain a *Help* button by which a block specific help information can be requested.

Changing the block size

All system blocks can be displayed in different sizes by BORIS. A new inserted block is normally displayed as big as possible (100%). This default setting can be changed by the display dialog (*Options/Adjust...*, entry field *Standard block size*).

To change the block-size of a single or of several blocks follow these steps:

1. Click on the menu-item *Display/Block size* window to open the block size window.
2. Select the blocks you want to modify.
3. Enter the desired size in the entry field *Relative size* of the block size window and click on the *Apply* button.

Block size window

Connecting blocks

How to connect two blocks

All connections between blocks are drawn by mouse actions. Because an integrated autorouter automatically routes the connections in a right-angled way and - as far as possible - free of cross points, only the two blocks to be connected have to be selected. In this connection the following principle is to observe:

Connections are drawn from input to output!

So to draw a connection proceed as follows:

1. First select the output square of that block where the connection shall start by clicking at it with the left mouse button. The mouse cursor now changes its symbol to a stylized soldering-iron.
2. Now the input square of the target block can be selected in the same way. If the border of the paint window is reached during the drawing process the window is scrolled automatically. To cancel a connection that has already been begun to draw, simply click at any free position within the paint window.

After a connection was completed correctly, it is drawn automatically by the integrated autorouter including the arrow at its end. The route is calculated in such a way that as far as possible no blocks or other connections are crossed. If nevertheless a crossing occurs, in most cases a small movement of the block can take this back. Connections that belong to the same block output are combined by the autorouter automatically.

Any block output can contain as much connections as you like. Any block input however can naturally have only one input signal. The connections itself can be displayed in different ways (refer to the *Options / Customize...* dialog).

Deleting connections

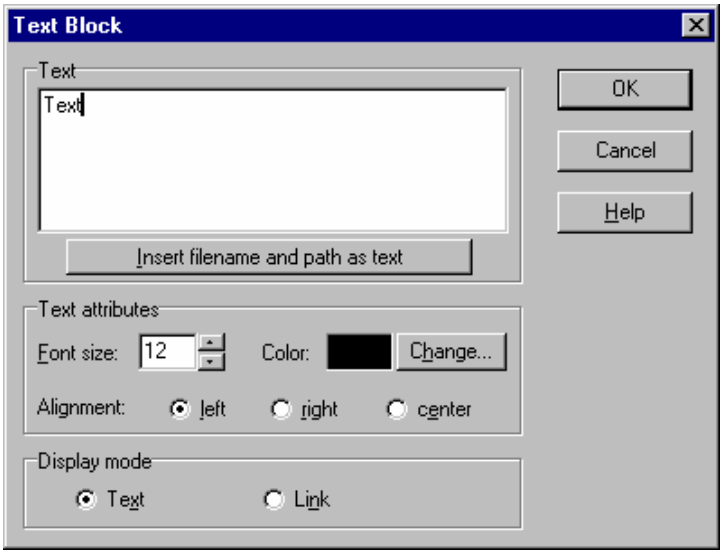
To delete an existing connection, first select the start or target block of the connections. In the following you have two alternatives:

- Output connections of a block are deleted by the *Edit / Delete output connections* or - more comfortable - the corresponding button of the control toolbar, input connections by the *Edit / Delete input connections* resp. the toolbar button. In both cases *all* connections of the blocks are deleted simultaneously.
- The second way is to use the popup menu (right mouse button).

Text blocks and frames

Using text blocks

Beside the "normal" system blocks BORIS offers text blocks as an opportunity to insert comments into the system structure. These text lines can be of different colours and sizes and may be moved as any other system block. To insert a text block use the *Edit / Insert text* command or the corresponding button of the control toolbar. Subsequently the default text "Text" appears in the upper left corner of the paint window. By a double click with the left mouse button this text can be modified (see next screen shot). By a single click and pressed left mouse button a text block can be moved in the same way as other blocks.



Parameter dialog for text blocks

Using frames

Another important feature for the improved documentation of system structures are *frames*. They represent a way for the graphical combination of system blocks that belong together in any way. Of course frames do not have any influence on the simulation itself.

To insert a frame...

- first select the blocks to be grouped
- then choose the *Edit / Insert frame* menu option or press the corresponding button of the control toolbar.

Attention: Frames are *static* constructs without any link to the blocks within the frame! So they can neither be moved nor are they deleted automatically if the blocks within are deleted.

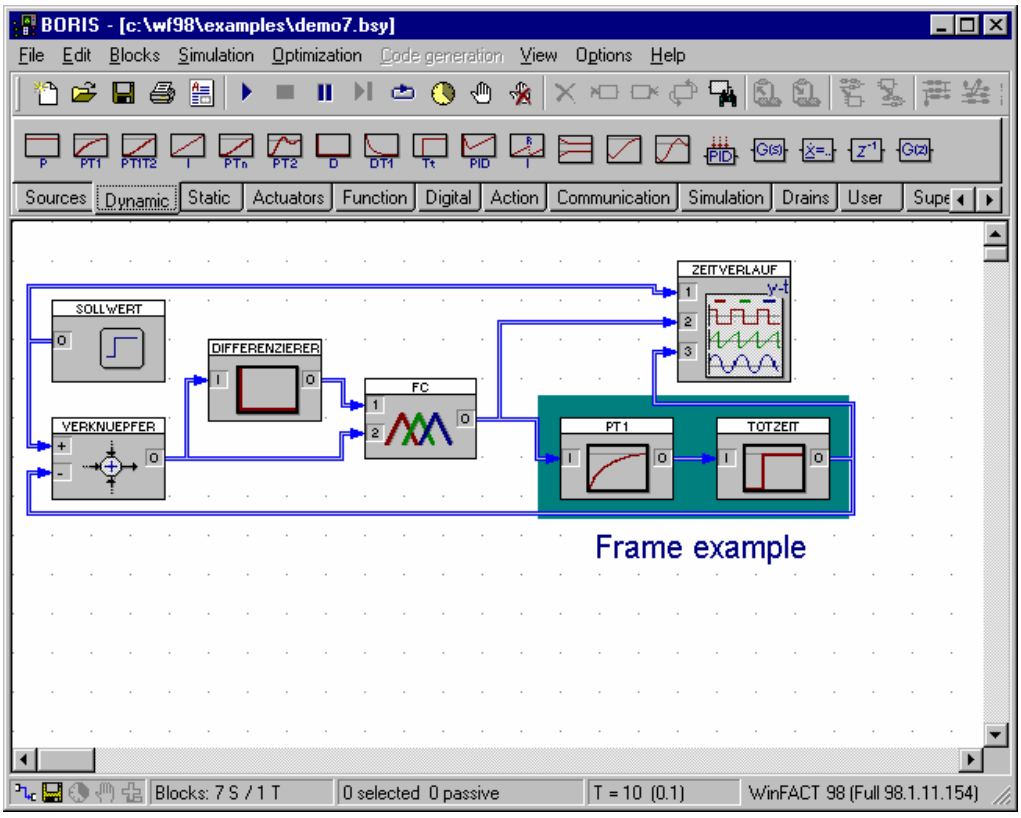
The inserted frame by default has a distance of eight points to the block surrounding rectangle, a black border with a full line of one point width and no inner colour. These parameters can be modified later (see next screen shot). For this purpose proceed as follows:

1. Select at least one block within the frame.
2. Choose the *Edit / Edit frame* command of the menu bar.

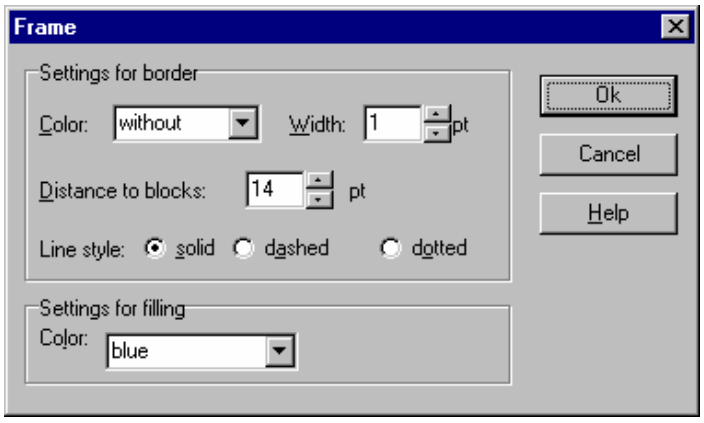
This command leads to the frames parameter dialog.

To delete a frame proceed as follows:

1. Select at least one block within the frame.
2. Choose the *Edit / Delete frame* command of the menu bar.



A little example how to use frames



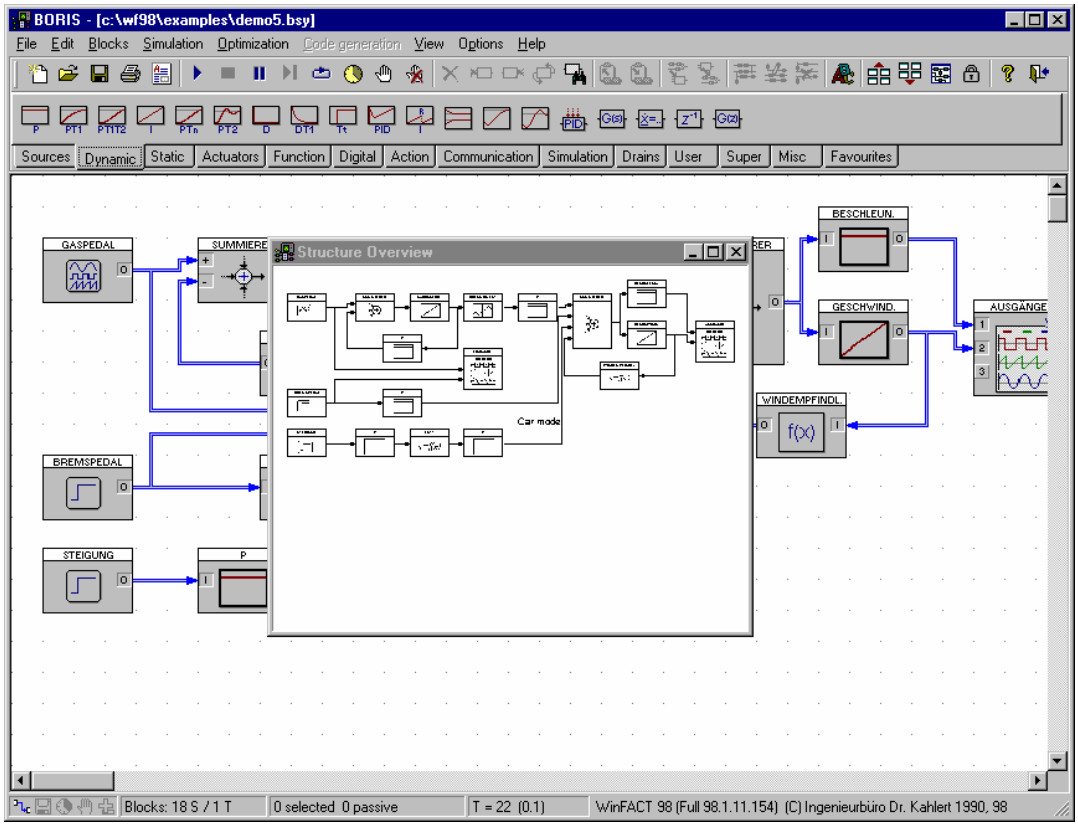
Parameter dialog for frames

Structure overview

When designing complex systems normally not all system blocks are visible within the paint window simultaneously. To get an overview of the designed system without scrolling many times a seperate window can be opened that shows all blocks in a zoomed mode so that all blocks are visible. This window is opened by the *View / Structure overview* command or the corresponding button of the control toolbar. The window can be moved and its size can be reduced or enlarged. As long as the overview window is visible it is actualized after each user action.

Within the overview window the system blocks are drawn in a more simple form without their specific bitmaps. Therefore this window is especially recommendable for the printer

output. This output can be realized by the *File / Print* command. Additionally the system structure can be exported to a WMF- or BMP-file by the *Edit / Export* command (not available in demo version).



Structure overview of a complex system

Simulation control

Simulation parameters

Before the simulation can be started, the simulation parameters have to be defined. These are:

- The simulation time T_{Simu}

  This value determines the time up to which the simulation is executed if it is not cancelled before. The default value is 10.

- The simulation step size ΔT

  This value determines the discretization step size for the simulation and thus the precision of the simulation results. If ΔT is choosen too large, discretization errors occur which in the worst case may lead to numerical instability. Normally ΔT should be not greater than 1/10 of the smallest time constant of the simulated system. The default value is 0.1

- The simulation algorithm

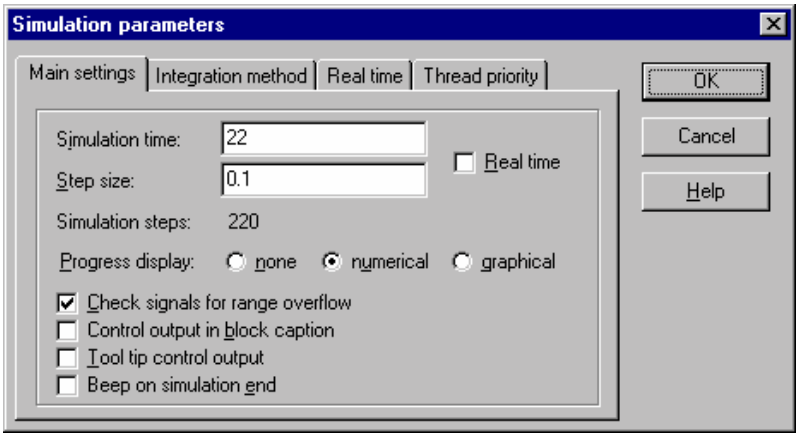
  Determines the choice of the numerical integration method for the simulation of the dynamic system components. Available are the Euler-method, the Runge-Kutta method of 4. order and the matrix exponential method.

- A range check during the simulation

If the option *Range check* is activated, all block outputs are checked after each simulation step. If at least one of the variables exceeds a value of 10^{20}, the simulation is a cancelled and a corresponding warning is displayed. Because this range check needs a certain calculation time, the simulation progresses a little bit slower if this option is activated (this is default).

- Time display in status line

To reach the simulation parameter dialog use the *Simulation / Parameters...* command or the corresponding toolbar button (see next screen shot).



Simulation parameter dialog

All settings concerning real time simulation are only of interest if BORIS is used in combination with a A/D-D/A-card.

The BORIS system block library

Types of system blocks

The system block library contains the following classes:

Inputs

To this class belong all those blocks that do only have outputs and thus generate input signals for the following blocks:

- Generator
- File input
- Constant
- Driving curve
- Controllable sinus generator (VCO)
- Simulation time
- Source

Dynamics

This class contains all types of linear and nonlinear dynamic blocks, beginning at a simple P-element up to transfer functions of higher order (time continuous or time discrete) and differential equation systems. The following types are available:

- P-Element
- PT_1-Element
- PT_2-Element
- PT_1T_2-Element
- PT_n-Element
- Limited integrator (I-Element)
- Limited integrator with reset
- Differentiator
- DT_1-Element
- PID-Controller
- Adaptive PID-Controller with controllable range
- All-pass element type I
- All-pass element type II
- Dead time
- Lead-/Lag-Element
- Transfer function *G(s)*
- User-defined differential equation system
- Unit delay z^{-1}
- *z*-Transfer function *H(z)*

Statics

This class mainly consists of static characteristic curves:

- Limiter
- Preload characteristic
- Dead zone
- Two point characteristic curve
- Three point characteristic curve
- Hysteresis
- Three point characteristic curve with hysteresis
- Fuzzy Controller
- User-defined characteristic curve

Final controlling elements

This class contains models of industrial actuators. Two different types are available:

- Actuator with limited control speed (Type I)
- Actuator with constant control speed (Type II)

Functionals

As functionals we define those system block types that can not be interpreted as transfer systems in the normal sense. The following blocks are available:

- Interconnecting element

- Function of one variable
- Function of two variables
- Function of several variables
- Extreme value determination
- Minimum/Maximum
- Statistics
- Sample and hold element
- Controllable S/H element
- Analog switch
- Analog change-over switch

Digitals

This class contains all blocks with binary outputs (high resp. low output). These are:

- Logical gate with one input
- Logical gate with two inputs
- RS-Flip-Flop
- D-Flip-Flop
- JK-Flip-Flop
- Mono-Flop
- Foreward/backward counter
- Discriminator
- Comparator
- Zero-axis crossing detector

Controlling elements

This class contains elements that concern the simulation control:

- Simulation cancel
- Simulation delay

Communication blocks

This class contains elements that concern the communication:

- DDE-In- and Output
- TCP/IP-blocks

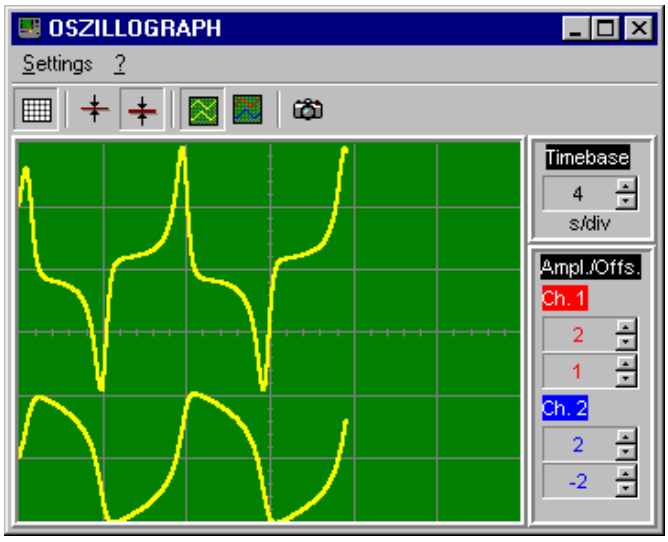
Interaction blocks

This class contains elements that allow an interactive control of the simulation process by the user:

- Push button
- Sliding potentiometer
- Rotation knob
- Industry PID-controller

Outputs

This type class contains all those block types that only have one or more inputs. Normally these are blocks for the visualization of simulation results or their further processing (e. g. oscilloscope or file-output). The first group - the so called "virtual instruments" - use a separate window - called output window - for the graphical representation of the input signals (see nect screen shot). This output window is automatically generated if the block itself is inserted. By the *View / Show all output windows* command all output windows can be enlarged from symbol to standard size, by the *View / Hide all output windows* they can be set to symbol size again. In most cases the size of output window can be modified by the user.



Output window example (here: Oscilloscope)

The following output blocks are available:

- Time response (y-t-plot)
- Oscilloscope
- Analog display
- Digital display
- Bar graph
- Trajectory display (x-y-plot)
- Status display
- FFT
- File output
- Table file output (Excel-format)
- Signal Drain

The specific block types shall not be explained in detail here. Each block offers a help function that can be called by the blocks parameter dialog. Only the fuzzy controller will be explained more extensive in the following chapter.

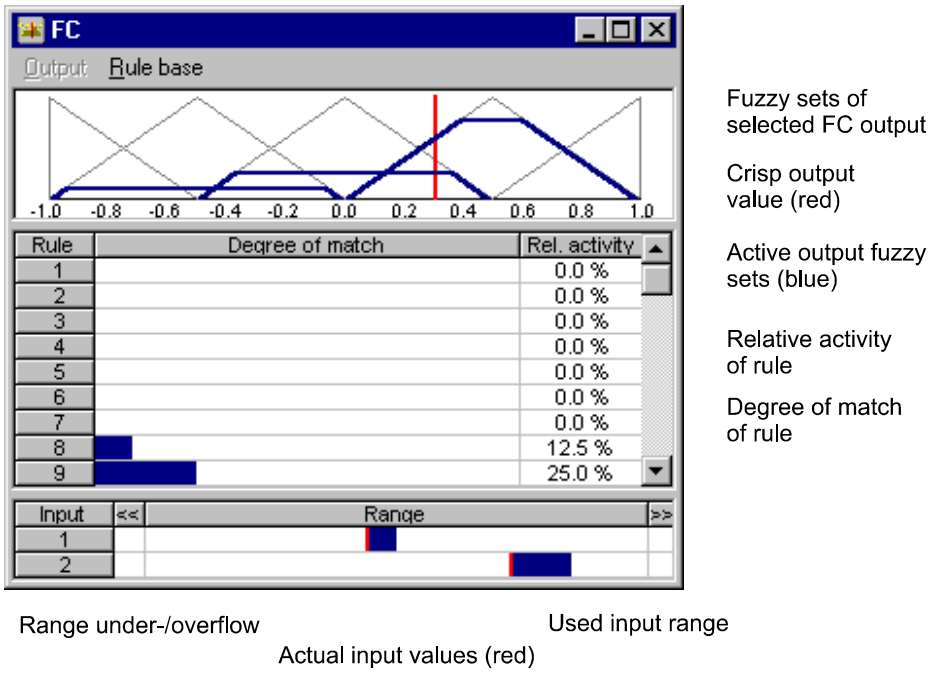
Fuzzy Controller and Fuzzy Debugger

The fuzzy controllers designed by the fuzzy shell FLOP can easily be integrated into a BORIS simulation structure. The fuzzy controller block has a control window called "fuzzy debugger" that displays all relevant informations about the fuzzy controllers inner states during the simulation. By this the fuzzy controller is an important help for the interactive design and analysis of fuzzy systems. The following screen shots show the fuzzy debugger and its parameter dialog.

During the simulation the following values are displayed within the fuzzy debugger window* :

- In the upper third of the window the fuzzy sets of the selected output variable (gray), the actual crisp output (red) and the resulting output fuzzy set (blue).
- In the middle third the actual degrees of match of all displayed rules (blue bars) and their relative activities. If, for example, a rule has a relative activity of 30%, the rule had an degree of match greater zero in 30% of all simulation steps.
- In the lower third the actual crisp input values (red) and the used input range for all input variables. If the input range of an input variable is left, this is marked by a yellow square at the left resp. right of the horizontal bar.

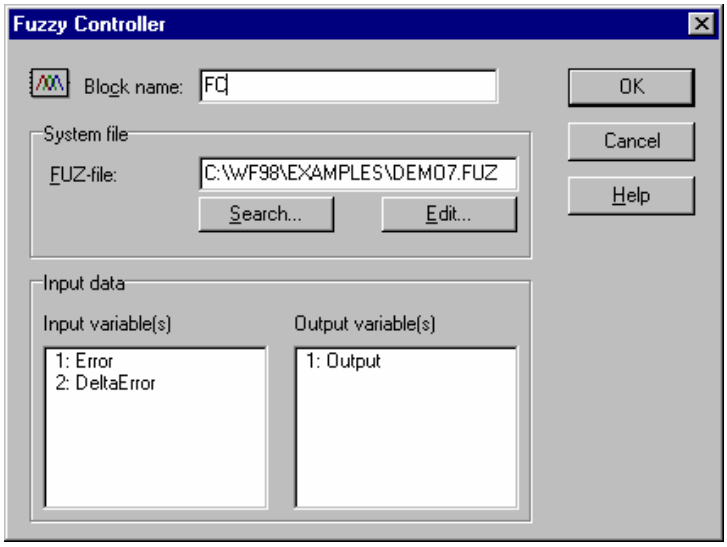
The output variable of the fuzzy controller that has to be displayed can be selected by the menu item *Output variable*. The menu option *Rule base* allows the output of the underlying rule base for controlling purposes.



Fuzzy debugger window

* For graphic adapters with more than 16 colours the colours may differ!

The parameter dialog of the fuzzy controller itself has the following design:



If no extension is defined for the *FUZ-file*, the extension FUZ is appended. By the *Search* button a file open dialog can be requested. The corresponding input and output variables are listed within the listboxes. After closing the dialog the number of block inputs and outputs is adjusted automatically.

Using superblocks

What is a superblock?

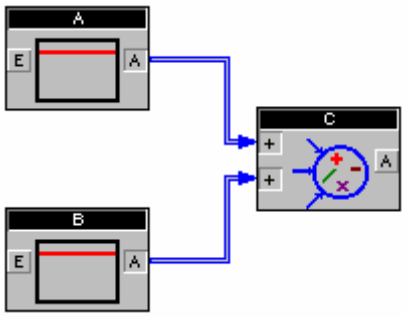
A superblock is a special system block type that results from the grouping of several system blocks and their connections. Thus a superblock is nothing else than a partial structure - a "black box" - being combined to a new block with in- and outputs. Therefore superblocks are especially recommendable for a clearly arranged structuring of complex systems and for the grouping of frequently used partial structures.

Superblocks are file referenced. All informations about the blocks and connections within the superblock are saved in files with the extension SBL. These files are - except an additional file header - identical with "normal" BORIS system files (BSY-files). Therefore superblock files can be saved as normal system files and vice versa, they also can be simulated.

Inputs and outputs of superblocks

The inputs and outputs of superblocks are automatically defined by BORIS basing upon the following rules:

- All open inputs and outputs of the selected system structure become inputs resp. outputs of the superblock. Example: The following structure has to be grouped to a superblock.



The resulting superblock has two inputs (the inputs of block A resp. B) and one output (the output of block C).

If block outputs that already contain a connection (e. g. a feedback) shall become superblock outputs, the use of label blocks is necessary. This special block type is availabe by the *Blocks / Label* command. By inserting a label and connecting its input to the other blocks output you easily create a new open output that in the following becomes a superblock output. If on the other hand an open input must not become a superblock input, connect it to a constant block with the value 0 (or another proper value).

- The inputs and outputs of the superblock are numbered in the sequence the corresponding blocks were included. If in- or outputs are to be changed, first delete the corresponding blocks and then insert them again in the correct order or - more comfortable - insert label blocks.

How to create a superblock

There are two ways to create a new superblock:

1. First create the superblock structure separately within the BORIS paint window. After the structure is complete, save it as a superblock file by the *File / Save superblock-file...* command. Later you can load the superblock simply by refering to this filename.
2. If a partial structure of an already existing system shall be grouped to a superblock, first select this partial structure and then create the superblock by the *Edit / Group to superblock* command, the Strg G-keys or the corresponding button of the control toolbar. In the following BORIS asks for the superblocks filename and then inserts the created superblock.

A superblock can be ungrouped at any time by the *Edit / Ungroup superblock* command, the Strg U-key or the corresponding button of the control toolbar. By this procedure all inner blocks and connections of the superblock appear again within the paint window; thus take care that there is enough free space in the superblocks neighbourhood!

Sample files:

The sample collection contains a lot of different sample files (Extension BSY).

Graphical presentation of results by INGO

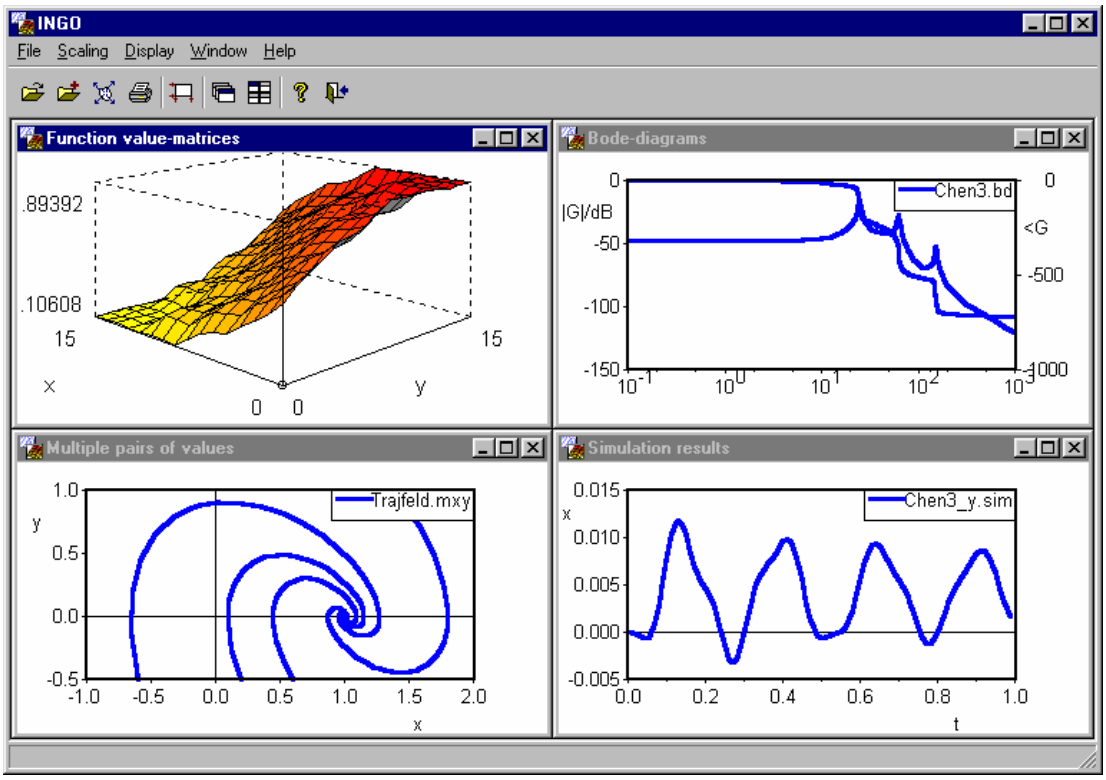
The WinFACT-module INGO allows the graphical presentation of all types of WinFACT-files:

- Simulation results (SIM-files)
- General data pairs (XY-files)
- Multiple data pairs (MXY-files)
- Bode-diagrams (BD-files)
- Nyquist-diagrams (OK-files)
- Function matrices (FWM-files, Presentation optional as 3D graphic or contour lines)

INGOs user interface is based on the WINDOWS multiple document interface and therefore allows the parallel presentation of several graphic windows (also of different types) whereby each graphic window can contain several curves. Each curve can be displayed as a line or as marker symbols.

A new graphic window is opened by the *File / New window* command, the Strg N-keys or the corresponding button of the toolbar. In the following a file open dialog appears; within this dialog the filename for the first input file has to be defined. The extension of the filename (e. g. SIM) automatically determines the type of the graphic window and thus the curves added later must fit this type.

To add a new curve to the active graphic window, use the *File / New curve* menu option, the Strg K-keys or the corresponding toolbar button. A graphic window of SIM, XY, MXY, BD or OK type can contain any number of curves. FWM-graphics which are presented in contour lines form can display a maximum of 30 contour lines per file. FWM-graphics in 3D-form however can display only one file per window.



INGO with some different graphic windows

The scaling of the axes is automatically adjusted in that way, that all curves are displayed completely. A modification of this automatic scaling can be performed by the *Scaling* menu option and the following parameter dialog.

Following the *Options / Miscellaneous...* command a dialog appears which offers some settings concerning line styles, diagram legend etc.

Sample files

The sample collection contains sample files for all types of WinFACT-graphic files.