Quantum Computing Optimizer (QCOptimizer)

Version 1

User Guide

1 Contents

1 CONTENTS 2

2 BEFORE YOU BEGIN 3

2.1 STRUCTURE OF THIS GUIDE. 3

3 INTRODUCTION TO QUANTUM COMPUTING OPTIMIZER 4

3.1 LEARNING SIGNAL FOR QCOPTIMIZER 4

4 USING QUANTUM COMPUTING OPTIMIZER 5

4.1 STARTING WITH QUANTUM COMPUTING OPTIMIZER 5
4.1.1 SCOptimizer program window 5
4.1.2 QCOptimizer toolbar 6
4.1.3 QCOptimizer menu 6
4.1.3.1 FILE Menu 6
4.1.3.2 VIEW Menu 6
4.1.3.3 HELP Menu 6
4.1.4 Dialog boxes 6
4.1.5 QCOptimizer files 7
4.2 WORKING WITH QCOPTIMIZER 7
4.2.1 Creating new model 7
4.2.2 Loading model from file 7
4.2.3 Saving model files 7
4.2.4 Importing SCOptimizer files 7
4.2.5 Changing locale settings 8
4.2.6 Working with model 9
4.2.7 Learning Signal Manager 10
4.2.8 Available blocks 11
4.2.8.1 Model Input blocks 11
4.2.8.2 Model Output blocks 12
4.2.8.3 Fuzzification Blocks 13
4.2.8.4 Inference Blocks 14
4.2.8.5 Constant Blocks 17
4.2.8.6 Quantum Generalization Blocks 17
4.2.8.7 File Blocks 20
4.2.8.8 Sum Blocks 20
4.2.8.9 Scale Blocks 21
4.2.9 External link 21
4.2.10 Viewing data passing through the model 22
4.2.11 Testing model on input signal from file 22
4.2.12 Optimization 22
4.2.12.1 Selecting optimization options 22
4.2.12.2 Starting optimization 23
4.2.12.3 Optimization progress/evaluation 24

5 SCLIB INTERFACE FOR MATLAB FITNESS FUNCTION CALCULATION 26

6 APPENDIX 1 – SUPPORTED MF DISTRIBUTIONS 27

2 Before you begin

This user guide assumes that you are familiar with fuzzy control theory and genetic optimization algorithms.
`Courier font` indicates program interface objects, like menu items, windows, pages and so on.
`BOLD Courier` indicates words or characters you type.
Important terms are selected with **bold** font.

2.1 Structure of this guide.

Section **3 Introduction to Quantum Computing Optimizer** tells you about the process of model creation with SCOptimizer. It will show you which steps you should perform and what is required for these steps.

Section **4 Using Quantum Computing Optimizer** is devoted to work with QCOptimizer interface. First part of this section **Starting with Quantum Computing Optimizer** describes basic interface principles**. Working with QCOptimizer** tells you how to use SCOptimizer to perform different optimization tasks and how to view and edit your model.

Section Ошибка! Источник ссылки не найден. **Using SCLib interface for Matlab fitness calculation** describe how to connect to QCOptimizer external link.

Reference information is available in Appendix.

3 Introduction to Quantum Computing Optimizer

Quantum Computing Optimizer (QCOptimizer) is a software tool designed for modeling, creation and optimization of complex control structures, including fuzzy and quantum control modules.

Working with QCOptimizer one can:

1. Create a model of complex control system, consisting from different blocks connected at any fashion;
2. Analyze performance of modeled control system and optimize it using different teaching signals.

3.1 Learning signal for QCOptimizer

In order to perform different optimization algorithms QCOptimizer requires **learning signal**, which presents samples of input values and corresponding output values. QCOptimizer is able to read signal data from Matlab v.4 and v.5 files and from text files.

Text files are processed based on **locale data**, which defines symbols for decimal point, thousands separators and so on. By default QCOptimizer uses windows settings for these parameters. If those settings do not match signal file format they can be changed at any moment. Once changed, locale parameters are saved in model and will be used for future processing of data. Locale setting affects reading and writing of text data files and model files.

4 Using Quantum Computing Optimizer

4.1 Starting with Quantum Computing Optimizer

Main module of Quantum Computing Optimizer is a Win32 application, called qcowin.exe. It can be used in Windows 95, 98, NT, Me, 2000 and XP environments.
In order to start the program you should type “QCOptimizer” in command prompt or double-click QCOptimizer icon in windows explorer.
During process of model optimization QCOptimizer creates temporary files in current directory. Those files have “.mms” extensions. Before starting QCOptimizer check that current directory does not contain files with same extension.

4.1.1 SCOptimizer program window

Main program window of QCOptimizer is shown on the figure 4.1.1.1.

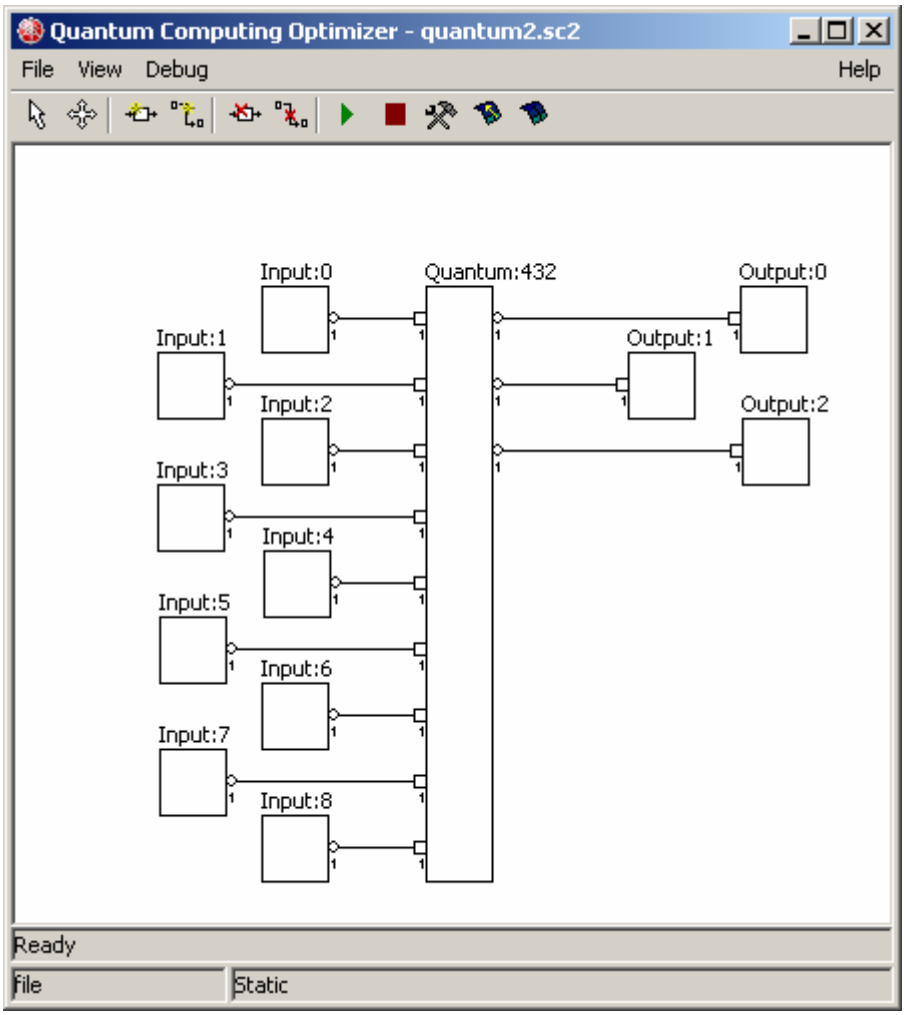

Figure 4.1.1.1. QCOptimizer program window.

QCOptimizer window is mostly occupied with model display. It displays current control system model as a set of connected blocks. Each block correspond to some data-processing module in control system. Lines show how inputs and outputs of modules are connected. Model display allows access to each module configuration and allows manipulation of modules and connections.

4.1.2 QCOptimizer toolbar

QCOptimizer toolbar is located at the upper part of the main window. Buttons on the toolbar provide access for model-editing and optimization functions. They will be described in detail in next sections.

4.1.3 QCOptimizer menu

QCOptimizer also has standard window menu, displayed on the top of program window. To choose a command with the mouse or keyboard, first select the menu and then choose command you want. Each underlined character in menu and command names corresponds to the key you can press to select a menu or to choose a command.

To choose a command by using the mouse you should first click the menu name containing the command you want and then click the command name in the pop-up menu.

To choose a command by using the keyboard first press the **ALT** key. The File menu appears selected. Use left and right arrow keys to select required menu and then press down arrow or **ENTER** to open pop-up menu. Use up and down arrows to select desired command and press **ENTER** to activate it. You can also press one of the underlined characters to quickly select desired menu or command.

4.1.3.1 FILE Menu

QCOptimizer allows project files to be managed by the means of commands in the FILE Menu.

The following items are available in this menu:

New: closes actual model and starts creation of new model.
Open: open existing model from disk file.
Import: import SCOptimizer file to the current model.
Save: save actual model to the file.
Close: close current model without saving.
Signal Manager: activate learning signal manager.
Number Format: set number format conventions for teaching signal and model files.
Exit: closes QCOptimizer.

4.1.3.2 VIEW Menu

View menu can be used to open signal view windows:

Open signal view: displays new signal window.

4.1.3.3 HELP Menu

Help menu currently have only one command – About, which displays version information of QCOptimizer.

4.1.4 Dialog boxes

When you choose a command having options, QCOptimizer shows a dialog box. A dialog box may contain fields in which you can enter text, numbers or select some items. Typical dialog box will have OK button, which you should press after filling all fields in order to activate the command, and CANCEL button, which will abort command execution. You can also use ENTER key on the keyboard to activate the same function as OK button and ESC as CANCEL.

If an operation require many parameters Wizard-style dialog box will be used. Wizard box consists of a sequence of dialog boxes that will guide you through the steps of an operation. Three buttons will be available for navigation of the wizard box. NEXT>> button tells wizard that you have successfully completed filling of current page and want to switch to the next page or perform the command if no

additional parameters are required. <<BACK button should be used to return to one of the previous pages, if you want to change some parameters. CANCEL button aborts command execution.

4.1.5 QCOptimizer files

SCOptimizer saves project in special files: **model files** (with .sc2 extension). **Model file** is used to store all model data, including model blocks, their properties, connections and positions, as well as most optimization and other settings.

QCOptimizer can save genetic algorithm optimization state to disk file, so you can continue optimization process interrupted some time ago. Those files have .st extension and will be called **state file**.

If you wish to copy QCOptimizer files to new location (for example to another computer) you should copy **model file, teaching signal** files and **state files**, if it is required. If your **teaching signal** file was originally located in directory other than one with project file, than QCOptimizer may be unable to load it automatically in new location. In this situation use Teaching Signal Manager to point QCOptimizer to new location of **teaching signal** file.

4.2 Working with QCOptimizer

4.2.1 Creating new model

Select File/New from menu to create new model. An empty model will be created and you can than add various blocks to it.

4.2.2 Loading model from file

If you have a file with previously saved model you should select File/Open command. Standard windows File Open dialog box will appear. Select file with you model and press Open button.

4.2.3 Saving model files

You can use File/Save command to save you model files. Standard windows File Save dialog box will appear. Select file with you model and press Save button.

We recommend to save you model regularly, in order to avoid loss of data due to possible program/computer failure or undesired model change. Note that most operations of QCOptimizer do not have “undo” option, so the only way to restore to previous state is to load model from saved file.

4.2.4 Importing SCOptimizer files

You can import SCOptimizer model files (.sco extension) in order to use fuzzy control systems developed in SCOptimzier in QCOptimizer projects. Select File/Import command to start model input. Standard windows File Open dialog box will appear. Select file with SCOptimizer model and press Open button.

Import options dialog will than appear:

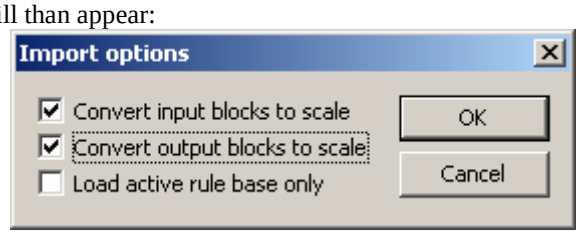

Figure 4.2.4.1. Import options dialog.

Check option checkboxes as required and press OK to import model. The options are following:

`Convert input blocks to scale`: by default imported SCOptimizer model includes new input blocks that are used to feed input data to fuzzy control system. If this option is selected the model will be imported with scale blocks instead, which will use input normalization parameters to scale signal. This option can be useful when creating models which includes several fuzzy control modules using the same input.

`Convert output blocks to scale`: same as before, but applied to fuzzy control output.

`Load active database only`: allow import of only one fuzzy rule database from SCOptimizer model, the active one. By default all databases are imported and active database is connected to output blocks.

4.2.5 Changing locale settings

This operation is available from `File/Number Format` menu and from New Model creation Wizard. After selection of this operation QCOptimizer will show you dialog box, containing current settings, like those:

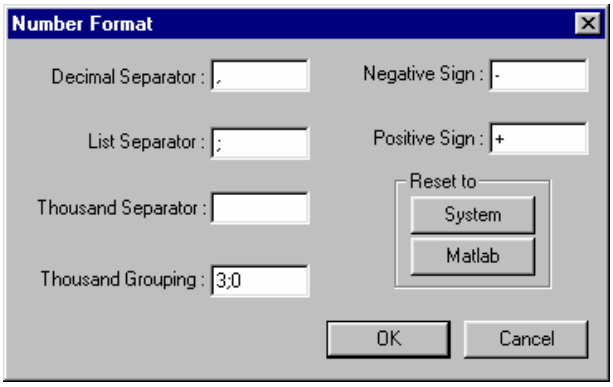

Figure 4.2.5.1. Number Format dialog box.

Every string in this list shows one of the locale parameters. Those parameters have following meaning:

List separator

List separator is a symbol or string used to separate several successive numbers in list.

Decimal separator

Separates integer and fractional part of numbers.

Negative sign

Defines negative numbers.

Positive sign

Shows that number is positive. SCOptimizer won't print this string before positive numbers, but will accept numbers with this sign in data files.

Grouping method

Define how digits of integer part of numbers will be grouped. This is a semicolon-separated list of numbers, defining number of digits in each group, from right to left. Trailing 0 means "use previous value for all following groups". SCOptimizer will insert the thousand separators in every place defined by grouping method, but will accept thousand separators in any position in input files.

Thousand's separator

Separates groups of digits, defined by **Grouping method**.

You can change any parameter by entering desired symbol(s) to the corresponding field. There are also two buttons, which you can use to switch to one of the predefined configurations:

`System`: switch to system defaults, as defined by Windows locale settings.

`Matlab`: switch to format, used by Matlab for text files. Following settings are default for the Matlab:

```
List separator: ';'
Decimal separator: '.'
Negative sign: '-'
Positive sign: '+'
Grouping method: '3;'
Thousand's separator: ' '
```

When entering locale parameters be careful not to:

- set different parameters to the same symbol
- make list separator, decimal separator or negative sign an empty string
- do not set any parameter to one of the following symbols: **(**,**)**,**{**,**}**,:,**'**.

Breaking these rules may cause program not to save and/or load files correctly.

After you set all options to desired characters press `OK` to apply changes or `CANCEL` if you want to reset you changes.

4.2.6 Working with model

You can edit model using model view in the main QCOptimizer window. It displays model as a set of processing modules, called blocks, and connections between them. You can add new blocks, delete existing, create and destroy links between blocks and access block properties.

Each block is displayed as a rectangle with block name written above of it. All blocks have some number of input and output ports, which allow signals to be passed to and from blocks. Input ports are drawn as squares on the left edge of the block, and output ports are drawn as circles on the right edge. The size of the block depends on number of input and output blocks.

Small numbers near input and output ports show port dimensions. For regular signals it is '1', for fuzzified signal it is number of membership functions used for fuzzification+1. Only ports with equal dimensions can be connected. Dimension of the port depend on each module type and configuration.

To access various edit functions QCOptimizer toolbar buttons are used:

- `Select tool`  : use this button to enable select mode (default). Click on the block to select it. Selected block is displayed with altered colors (depend ending on windows color settings). You can select multiple blocks by holding `CTRL` key and clicking on blocks to be selected. To deselect all blocks click on a free space.
  Selected block[s] can be moved across display. To move, click and hold left mouse on selected block, and drag it to desired location. Release mouse button to stop moving.
  You can also access block properties in this mode. Double click left mouse button over block to display corresponding option dialog box.
- `Move window tool`  : select this button to move display window if the model does not fit inside screen. Click toolbar button and then click and hold left mouse button on model view and drag mouse to change display position. Release mouse to stop moving.
- `Create block`  : click this button to create new block. After you click it menu with list of available block types will be displayed. Select required block type from this menu. Depending on block type dialog box may appear, where you should enter initial block settings. After finishing with block settings you will see new block displayed in model view. Drag block to it's desired location and click mouse to finish block creation.
- `Create link tool`  : click this button to create link between blocks. Then click to ports (one input and one output, in any order) to create link between ports. Any output port can have as many connections as required. Input ports can have only one connection. If you select input port which already have connection it will be reconnected as instructed.
- `Delete block tool`  : To delete block click this button and then block to be deleted.

- `Delete link tool`  : To delete link click this button and then input port, which should be disconnected. You cannot delete links by clicking on output ports.

4.2.7 Learning Signal Manager

If you want to use **learning signal** for optimization it should be loaded and assigned for model variables before use.

QCOptimizer may load teaching signal from Matlab v.4 and v.5 files and from text files. File type is autodetected. If the file does not look like Matlab file than it assumed to be a text file.

Matlab file should contain an array of real numbers with number of columns equal to sum of number of input and output variables of the model. If version file contains several variables then first one will be loaded.

Text files are processed using **locale settings**. Please set **locale settings** as described in section 4.2.5 before loading text files. If **locale settings** do not match the file format than signal will be loaded incorrectly and no error message will typically be displayed. Text file should contain **learning signal** data separated by any separators. This will include text files produced by Matlab, CSV files and practically any other file of this kind.

To manage learning signals select `File/Signal Manager` command from QCOptimizer menu. `Signal manager window` will be displayed:

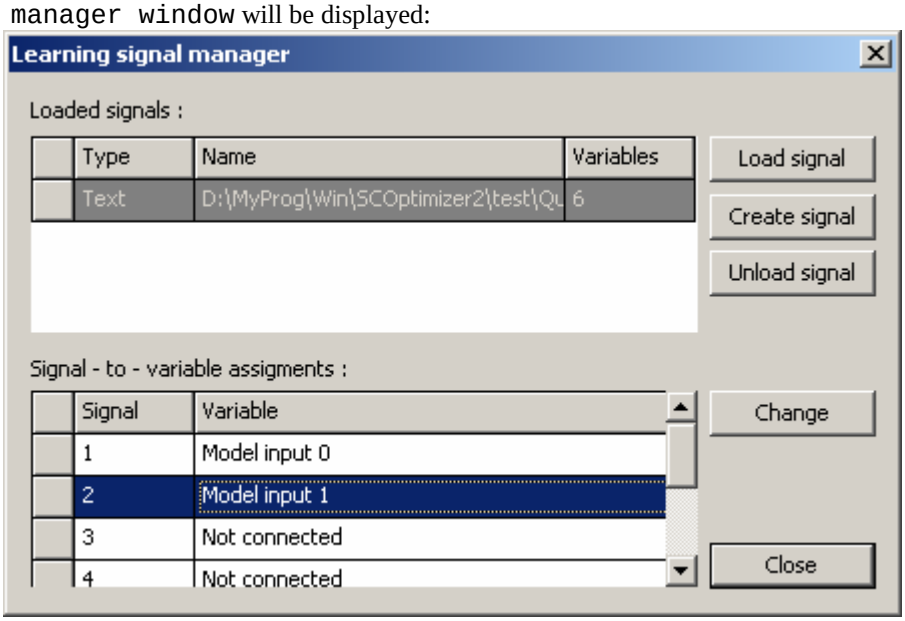

Figure 4.2.7.1. Signal manager dialog box.

To load new signal file press `Load signal` button. Standard windows file open dialog will be displayed. Select file with signal and click `Open` to load it. Loaded files will appear in list box labeled `Loaded signals`.

Click on a file name in this list box and it signal to variable assignments will be displayed below. Those define how loaded signal and model variables are related. By default table show “not connected” to indicate that signal variable is not assigned to model variable. To set the assignment (or change it) click `Change` button. Another dialog will appear:

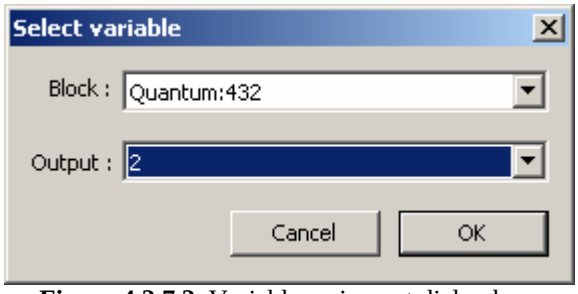

Figure 4.2.7.2. Variable assigment dialog box.

In this dialog select block and **output** port name corresponding to loaded teaching signal variable. Press `OK` to save assignment. To assign signal to model input select "Model Input/Output" as block name, than select corresponding input or output in second list.

Loaded and assigned signals can be used during optimization, if teaching signal is selected as a source of fitness function in `optimization manager` parameters. In this case for model evaluation loaded signals will be presented on model input and block outputs will be compared to assigned signals.

File with learning signal can be unloaded by selecting it in the list and clicking `Unload` button. Unloaded signals will not be used during optimization.

4.2.8 Available blocks

Blocks are building parts of a control system models. Following block types are supported:

- **Model input**: feeds input data to model;
- **Model output**: used to pass calculated output from model to external world;
- **Fuzzification**: performs fuzzification of signal;
- **Inference**: fuzzy rule database, performing Sugeno inference;
- **Constant**: generates constant signal;
- **Quantum generalization**: perform quantum generalization algorithm;
- **File**: injects data from external file to model;
- **Sum**: calculates weighted sum of it's inputs;
- **Scale**: perform scale and offset on signal.

Blocks are described in the following sections.

4.2.8.1 Model Input blocks

Input blocks are used to pass incoming data to model. They do not have input ports and receive data from external connection or assigned learning signal file.

Input blocks are numbered from 0, and the index of block is shown after block name. Vector signals from external connection are separated to components and send to input blocks according to their index. Block index depend on the order in which blocks where created and can only be changed by deleting blocks with lower indexes.

Input blocks also support signal normalization. To alter signal normalization parameters double click block. Following dialog will be displayed:

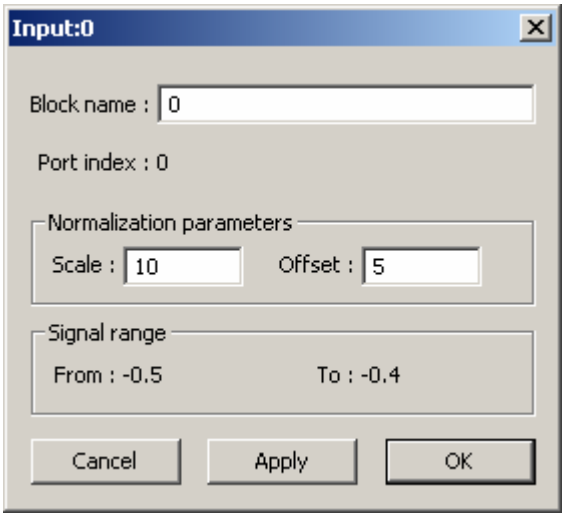

Figure 4.2.8.1.1. Input block parameter dialog box.

In this dialog you can change block name (initially equals block index) and normalization parameters. Signal range corresponding to current not-normalized signal are shown below.

Press `OK` to accept changes and close window. Press `Apply` to change block parameters without closing window. Press `Cancel` to close window without applying modifications.

4.2.8.2 Model Output blocks

Output blocks are used to pass results of calculation to external world. Data arriving at output blocks is then sent over external link or compared to assigned learning signals.

Output blocks are numbered from 0, and the index of block is shown after block name. Output signal sent over external connection are compiled from signals of output blocks in order of their indexes. Block index depend on the order in which blocks where created and can only be changed by deleting blocks with lower indexes.

Output blocks support signal denormalization. To alter signal normalization parameters double click block. Following dialog will be displayed:

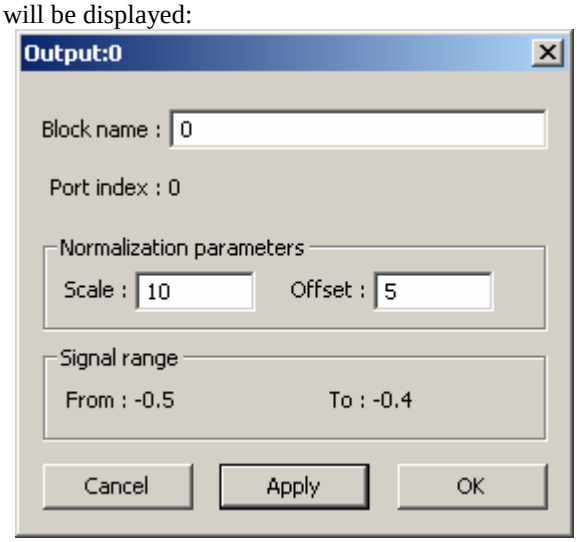

Figure 4.2.8.1.1. Input block parameter dialog box.

In this dialog you can change block name (initially equals block index) and normalization parameters. Signal range corresponding to current not-normalized signal are shown below.

Note that output block perform demoralization on the signal, so that directly connected input and output blocks with equal parameters will leave signal intact. Input block offset signal to offset parameter and than scale it, **output block perform reverse operation**.

Press `OK` to accept changes and close window. Press `Apply` to change block parameters without closing window. Press `Cancel` to close window without applying modifications.

4.2.8.3 Fuzzification Blocks

Fuzzification blocks perform fuzzufication of input signal. During block creation dialog box will appear where you should enter number of fuzzy membership functions and block name:

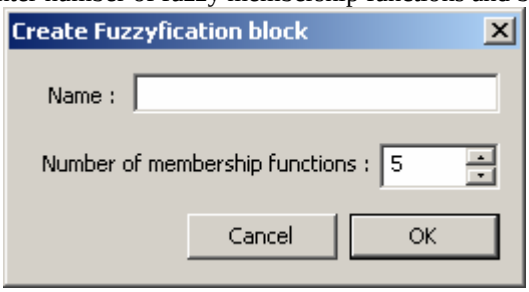

Figure 4.2.8.3.1. Fuzzification block creation dialog.

Once created fuzzification block parameters can be set using properties dialog.

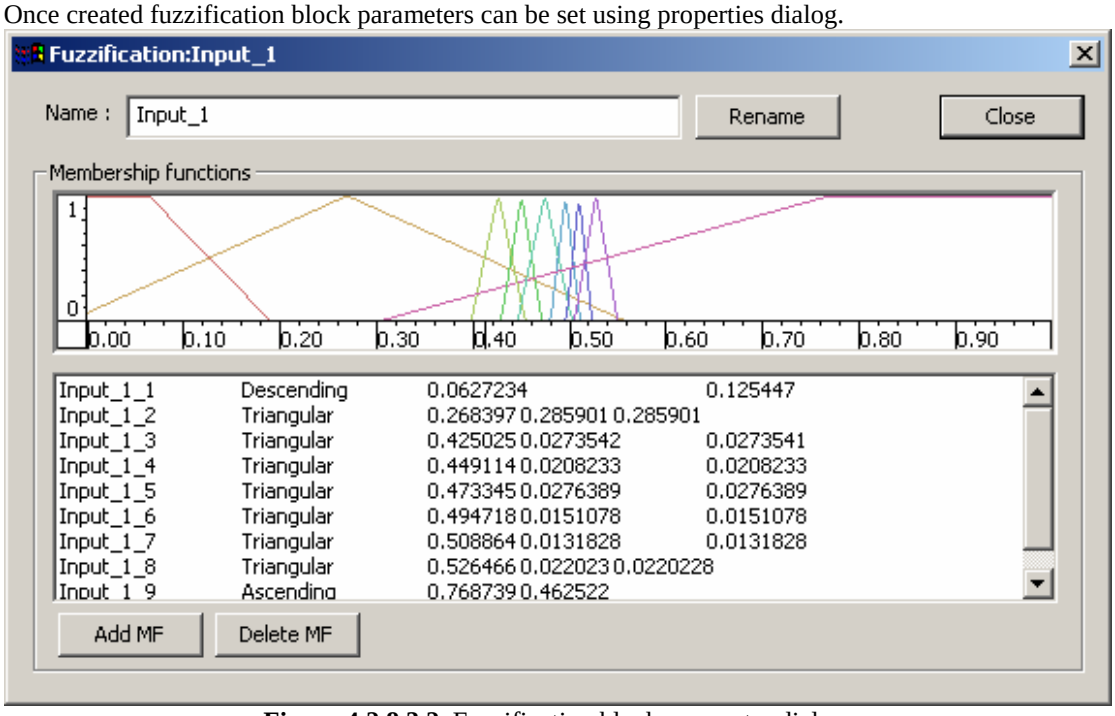

Figure 4.2.8.3.2. Fuzzification block parameter dialog.

There is the name of the block atop of the dialog.

Then there is display **membership functions** (MF's) of the variable.

Graphical window display **distribution functions** of MF's. You can change appearance of this window by the pop-up menu, activated by the right-click of the mouse in the window. Menu items are the following:

`Track cursor`: Use this feature to see the margins of alpha-levels. When mouse cursor is on the y-axis then lines representing alpha level will be drawn, as well as color lines which will show margins of this level for all MF's. When cursor is somewhere else on the window then vertical lines at the position of cursor and horizontal lines from intersections of this line with MF's will be drawn.

`Display MF supports`: Display supports of MF's using colored vertical lines.

`Display signal interval`: Display margins of signal change interval with vertical lines.

`Color shapes`: Draw functions using filled color figures (default).

`Color lines`: Draw functions using color lines.

`B&W lines`: Draw functions using black lines.

`Save Image`: Save current image to file (Windows BMP format).

You can use this window to change MF distribution parameters. Move mouse to the x coordinate of the modal value or support margin of one of the MF's. Colored line will appear showing selected parameter. Press and hold left mouse button. Move mouse left or right to change the parameter. New shape of the MF will be drawn using outline method. Release left mouse button when you are satisfied with the shape of MF.

The list in the very bottom of the page display **membership functions** and their parameters. First column of the list is MF name, next – **distribution type** followed by **distribution parameters**.

You can change those parameters by double clicking list items. If you do, the following dialog box will appear:

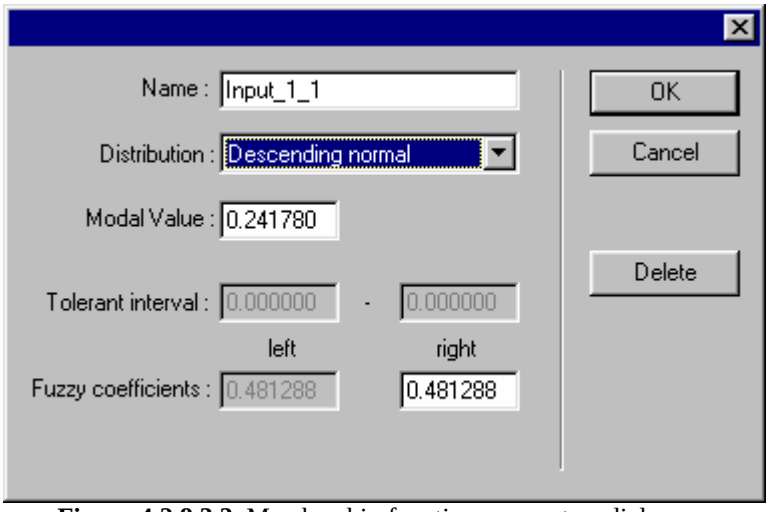

Figure 4.2.8.3.3. Membership function parameters dialog.

This dialog box display all parameters of the membership function. Parameters can be changed by entering new data into corresponding fields. Please note that some parameters may not be available for different distributions. For example **Descending normal** distribution require only **Modal Value** parameter and **Right Fuzzy Coefficient** parameter. Fields, corresponding to the unused parameters, are grayed and can not be changed.

When you are done with this dialog press `OK` to apply you changes or `Cancel` to return without modifying the variable.

You can also use this dialog to delete **membership functions** of input variables. If you wish to do it press `Delete` button.

`Add MF` button allow you to manually add new membership function to the current variable. After you press this button the same dialog as was used for editing new variable will appear and you will be able to enter parameters of newly added membership function. If you press `OK` MF with those parameters will be added, if you press `CANCEL` it won't. Since rule database structure is highly dependant on number of MF's, rule database will be cleared and recreated if you add a MF.

4.2.8.4 Inference Blocks

Inference block support fuzzy inference with Sugeno model. When the block is being created following dialog will appear, where you should enter initial block parameters:

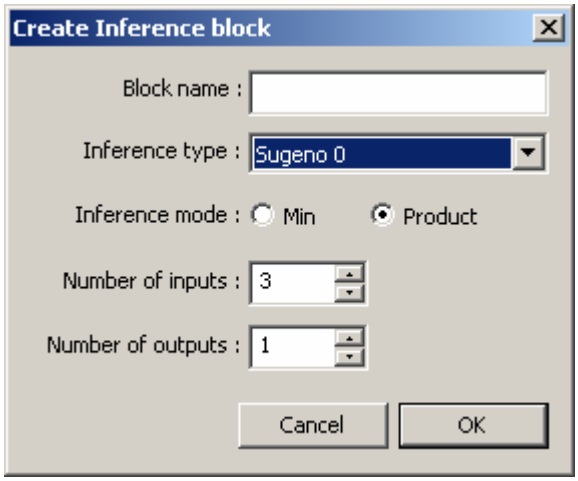

Figure 4.2.8.4.1. Inference block creation page.

To view or edit inference block properties double click the block. Following dialog box will appear:

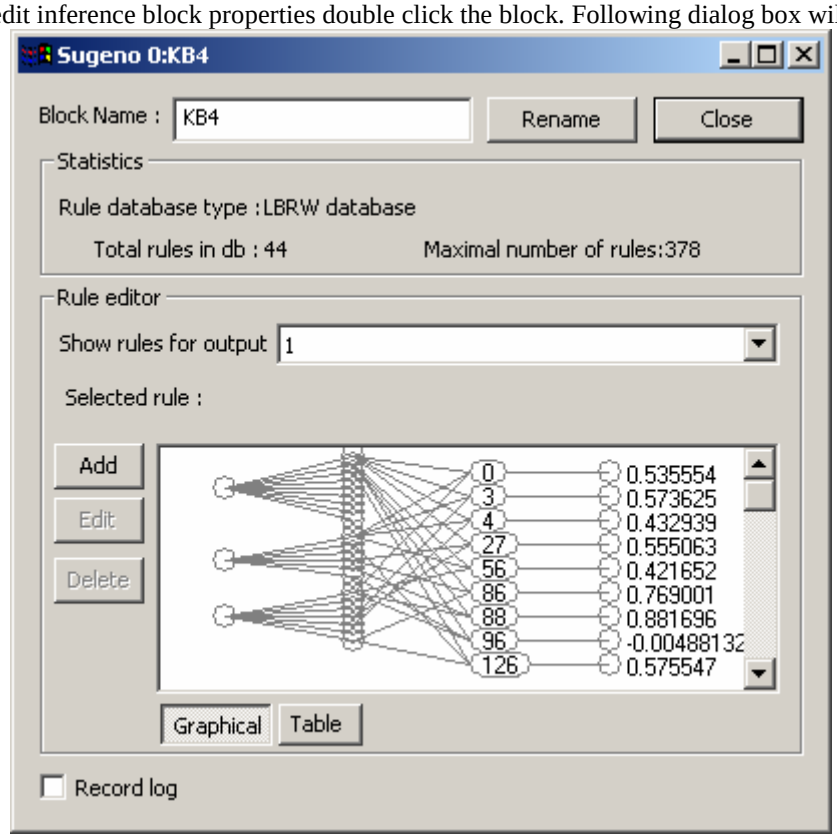

Figure 4.2.8.4.1. Inference block parameter dialog.

This dialog displays following parameters:

`Rule database type`: type of the database used in model. Can be **Complete rule database** or **LBRW Rule database**.

`Maximal number of rules`: Maximal number of rules for current model.

`Total rules in db`: Number of rules stored in the database. For **complete database** this is always equal to maximal number of rules. For **LBRW database** it can be less then maximal.

`Show rules for output`: Rule database is displayed for one of the outputs. This list selects output variable for which database will be displayed.

`Selected rule`: Displays textual representation of selected rule, if any.

Rule Database editor display database as network with four layers. First layer `is input variable layer`. Each circle in this layer represents **input variable**. Second layer is `input MF layer`.

Circles of this layer represent **membership functions** of variables. Circles in the third layer represent **rules** of the database. Number written inside this circle is a **rule number** of the rule in the database. Last layer is the `output layer`. For **Mamdani model** `output layer` is composed of circles, corresponding **to membership functions** of selected output variable. For **Sugeno models** `output layer` displays numerical parameters of the rule.

Database structure is shown with lines, connecting different layers. Each node in the `rule level` is linked with those MF's in `input MF layer`, which are included in the **if-part** of the rule. It is also linked with output MF or numerical parameter of the **then-part**.

Since all rules of the model won't fit on display, scroll bar is implemented which scrolls nodes of the `rule level`.

You can select a **rule** from the database by clicking on the node of the `rule level`. Textual representation of the rule will be shown in the `Selected rule` field and you will be able to edit or delete this rule. When you select a rule in the database it activation level is displayed as red lines on the `Rule Activation Level` page.

If you wish to delete the rule, select it and press `delete` button. Rule will be removed from the database. Note that rules from complete database can not be deleted.

By pressing `Edit` button you can change selected rule. The following dialog will appear:

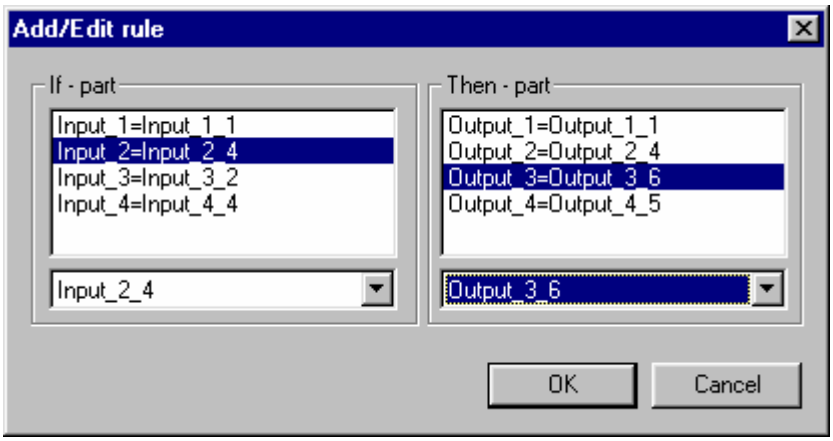

Figure Ошибка! Источник ссылки не найден.**.2**. Add/Edit Rule dialog for Mamdani model.

The left part of the dialog represents **if-part** of the rule, and the right part corresponds to **the then-part**. You can change parameters of any part by selecting items from the list and changing values in the drop-down box below the list.

For **Sugeno 0** model this dialog will have a slightly different appearance:

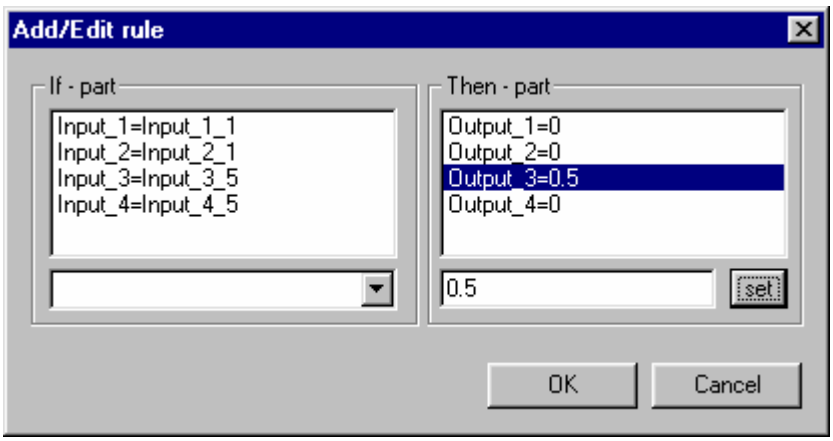

Figure Ошибка! Источник ссылки не найден.**.3**. Add/Edit Rule dialog for Sugeno 0 model.

To change output parameter for **Sugeno 0** model select corresponding line from the list, enter new value in the text field below and press `set`.

You can add rule to the database by pressing `Add` button. The same dialog as used for rule editing will appear. Change values in the **if-part** and **then-part** as desired and press `OK` to add rule. If the rule with selected **if-part** already exists in the database it will be replaced with new rule.

4.2.8.5 Constant Blocks

Constant blocks can be used to create constant signals in the model. During block creation the following dialog box will appear:

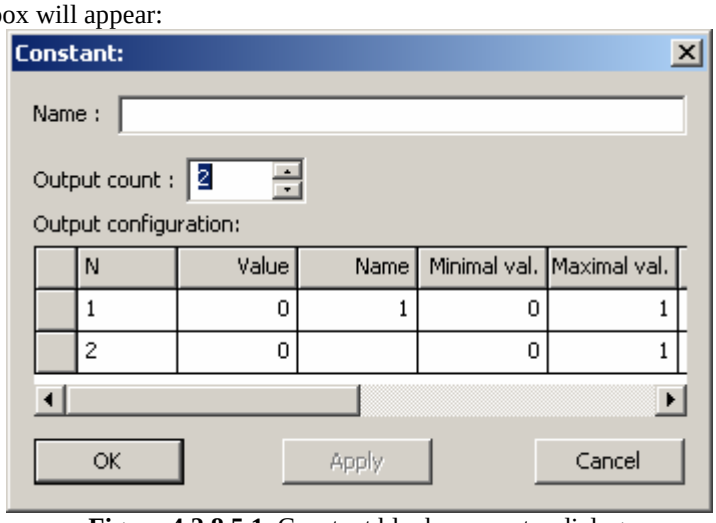

Figure 4.2.8.5.1. Constant block parameter dialog.

Enter number of block outputs desired and configuration of each output, as follows:

Value – constant value
Minimal and Maximal values – range of value search for block optimization
Cyclic – indicate that value have cyclic property which is used during optimization

Number of block outputs cannot be changed after the block is created. Other parameters can be changed by activating parameter dialog by double clicking on the block.

4.2.8.6 Quantum Generalization Blocks

During creation of quantum generalization block you should enter initial parameters in the following dialog:

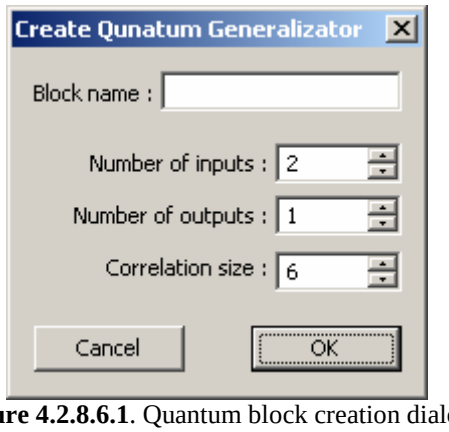

Figure 4.2.8.6.1. Quantum block creation dialog.

Enter block name, number of inputs and outputs, size of correlation vector. Correlation vector size can be changed later, while number of inputs and outputs cannot. When done press `OK` to create block.

Configuration of block can be checked and changed by double clicking on the block. Following dialog box will appear:

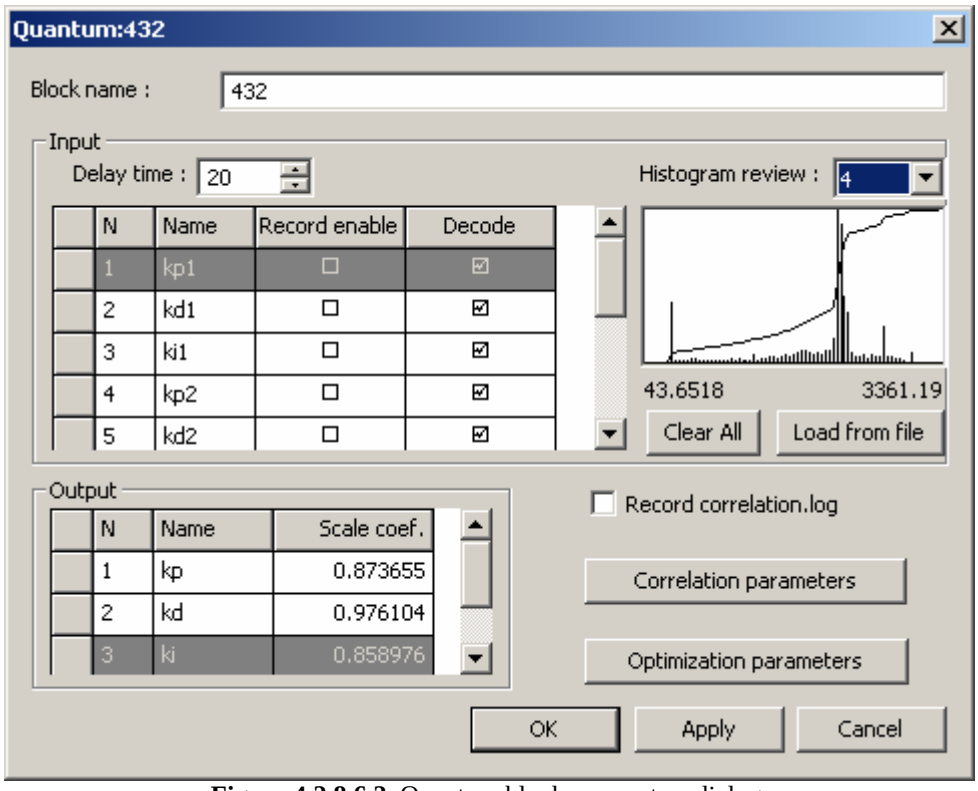

Figure 4.2.8.6.2. Quantum block parameters dialog.

Main page of parameter dialog show configuration of inputs and outputs. Table in the input section lists following configurable parameters for each input:

`Name` – input name, to be used in correlation configuration;
`Record enable` – check to enable histogram recording for selected channel;
`Decode` – uncheck to skip this input during decoding (required by QPID algorithm).

To the left of table there is a histogram review window. It shows histogram for the channel selected in the drop-down list above it. `Clear All` button can be used to erase all recorded histograms for all channels.

`Delay time` control in the top part of the section determine amount of delay introduced for temporal correlation.

In the output section there is a table with output parameters:

`Name` – output name, to be used in correlation configuration;
`Scale coef.` – output is scaled to this constant.

Output scale coefficients can be optimized. Optimization settings can be accessed in dialog box activated by `Optimization Parameters` button:

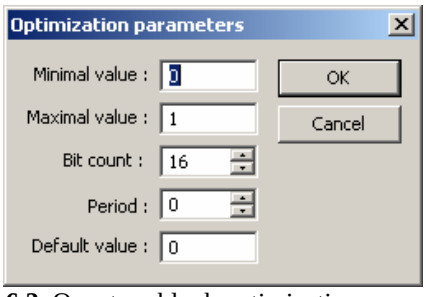

Figure 4.2.8.6.3. Quantum block optimization parameters dialog.

This dialog allow selection of parameter search range, default value, number of bits for binary GA chromosomes and periodic property, if any.

Quantum correlation can be configured by pressing `Correlation parameters` button:

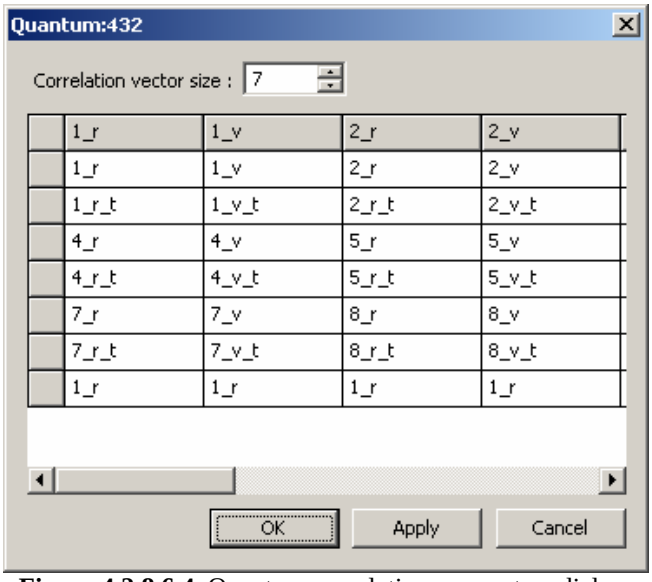

Figure 4.2.8.6.4. Quantum correlation parameters dialog.

Input field at the top part of window allow changing correlation vector size. Table below lists current correlation vectors. Each to columns of the table correspond to real (one with "_r" appended) and virtual ("_v") parts of a vector for each output.
Entries in the table show which value will be used at which position of a vector. Those values can be changed by clicking on them and selecting new value from menu. Name of the component is composed of input name (or input number if name is not specified), and modifier, as follows:

_r – real component is used;
_r_t – real delayed component used;
_v – virtual component used;
_v_d – virtual delayed.

Quantum algorithm requires the histogram to be loaded for each channel. Histograms can be recorded during block operation or by processing file from disk.

In the first case check `Record enable` checkbox in input table of corresponding inputs and run signal through the model. Data appeared on inputs corresponding to checked table lines will be recorded in histogram.

To load histogram press "Load from file button".

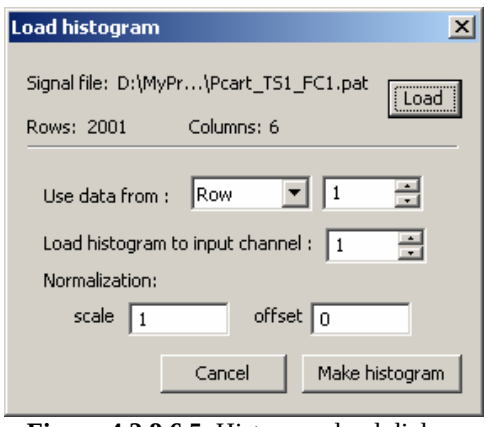

Figure 4.2.8.6.5. Histogram load dialog.

In the dialog appear first press `Load` button and select to file to be used. File name and number of rows and columns will be displayed. Then select if the data used will be taken from `row` or `column` (depending on file dimensions and format). Select file row/column index and index of input where histogram should be loaded. If required enter normalization parameters. Press `Make histogram` to load specified data. You can than select another indexes and load histograms to other inputs by pressing `Make histogram` again. Press `Cancel` when done with loading.

4.2.8.7 File Blocks

File block can be used to insert data from external file to the model. Block support reading of Matlab v.5 and v.6 and text fles. File type is detected automatically.
During creation of the block following dialog box will appear:

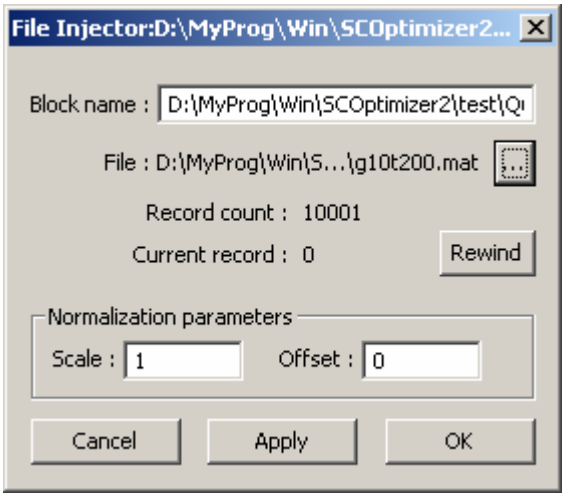

Figure 4.2.8.7.1. File block parameter dialog.

Select file you wish to use by clicking on "…" button and selecting file name using standard windows file open dialog. Data from file can be normalized before sending to block output. Normalization parameters should be entered to the corresponding fields of the dialog. Press `OK` to create block.

During operation file block output one record from the file for every model calculation step. Current record index is shown in block properties dialog and can be reset to zero by pressing `Rewind` button. Block automatically restarts from first record when all records from file are read.

4.2.8.8 Sum Blocks

Sum block calculate weighted average of its inputs. During creation of a block dialog window will appear where you should select required number of inputs and initial weights:

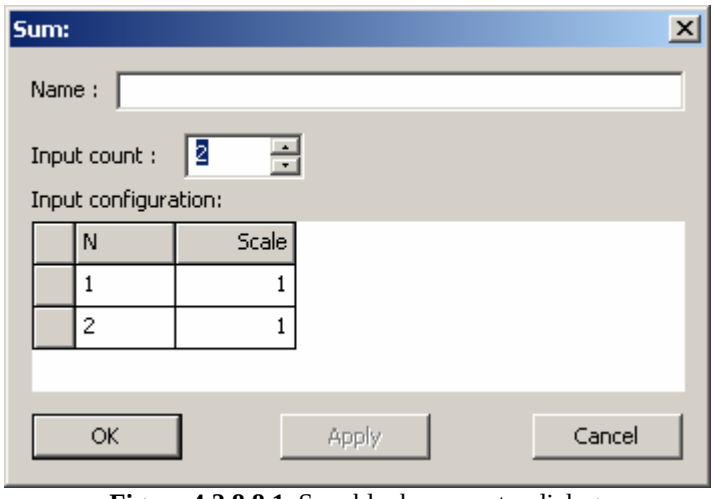

Figure 4.2.8.8.1. Sum block parameter dialog.

Enter desired number of inputs in input count field, weight factors in scale column in input configuration table and press `OK` to create the block.

You cannot change number of inputs of sum block, but input weights can be changed. To do it double click the block. The same dialog window will appear, where you can set weight factors as desired.

4.2.8.9 Scale Blocks

Scale blocks transform input signal by scaling and offsetting it. To set scale and offset parameters double click the block. Following dialog will appear:

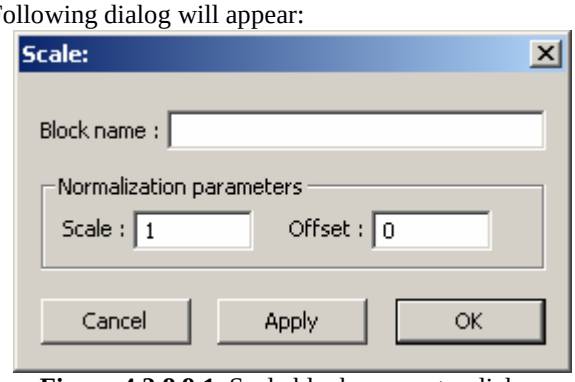

Figure 4.2.8.9.1. Scale block parameter dialog.

Set desired parameters in dialog and press `OK` or `Apply` to change block parameters.

4.2.9 External link

QCOptimizer support data exchange with external programs, which can be used to pass data from external program as input to QCOptimizer model and return resulting model output. This link uses the same interface as one available for external inference in SCOptimizer, and can be accessed from SCLib library functions. Link interface is described in section 5.

If current fitness calculation mode is set to “teaching signal” external link is not used and cannot be activated. Change fitness calculation mode to Matlab/external signal to enable external link access.

To connect via external link QCOptimizer and another program should use the same link name. Default link name is SC2. Link name in SCOptimizer can be changed by pressing  toolbar button. Dialog box will appear where you can enter desired link name:

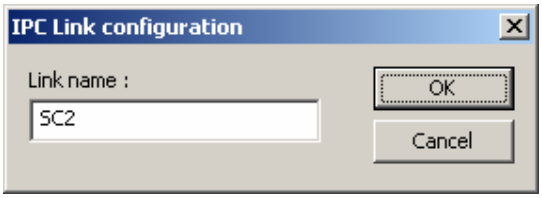

Figure 4.2.9.1. Link name configuration dialog.

Current link name is also displayed in the status line in the left-bottom corner of main QCOptimizer window preceding with “IPC:”.

To enable external link press  toolbar button. To indicate that QCOptimizer is ready for external connection an ‘*’ symbol will be shown after link name in status line. To disable external connection press  , ‘*” will disappear.

Always enable external link first from QCOptimizer and than in external program. Close link in reverse order – first in external program, than in QCOptimizer.

4.2.10 Viewing data passing through the model

You can view signals passing through the model in the graphical representation. To view signal select `View/Open signal view` menu item. Signal view window will appear, showing signals on model input. To change signal shown in window click right mouse button above window. In the menu appear select block (or model input/output) and than block output.
You can open as many data view windows as you like, by selecting the same menu item several times.

4.2.11 Testing model on input signal from file

You can make QCOptimizer perform calculations using input signal recorded in file. To do so you should:

1. Load signal using teaching signal manager and associate it with model inputs.
2. Select “teaching signal” fitness mode calculation in optimization configuration.

Once those two actions are done, you should see “file” displayed in the status line typically used to show external link name. Press  toolbar button to perform a full run through data available in loaded files. You can view resulting signals using signal view windows, or you can use this mode to record histograms in quantum block.

4.2.12 Optimization

QCOptimizer support various optimization algorithms and modes. It can perform optimization of single block, or several blocks at a time, including optimization with different algorithms, running at the same time and evaluated over same link.

4.2.12.1 Selecting optimization options

To set optimization settings press  toolbar button. Following dialog box will appear:

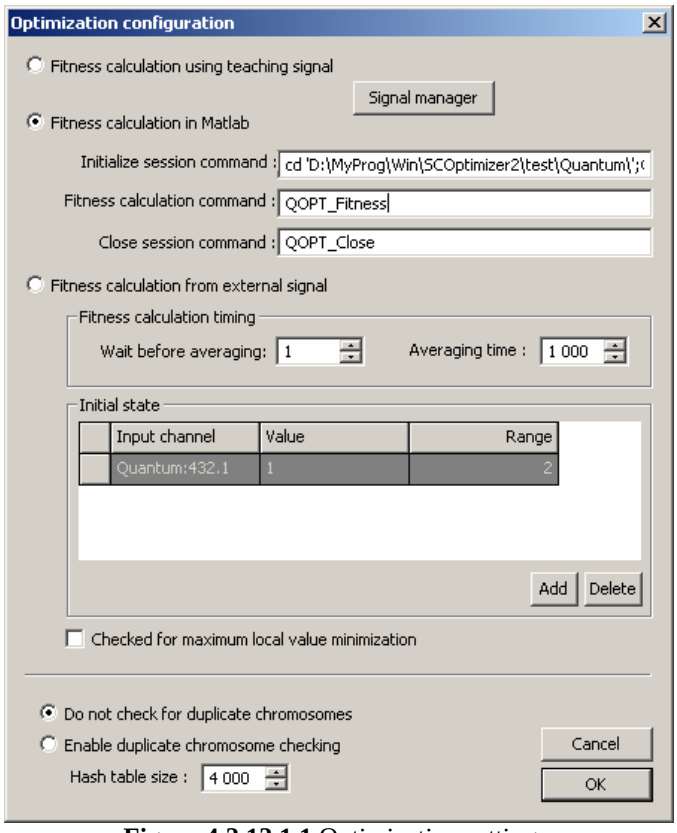

Figure 4.2.12.1.1 Optimization settings.

First select desired fitness calculation mode:

- Teaching signal: model is evaluated using teaching signal loaded in teaching signal manager and fitness is computed by comparing actual data from model with data selected as teaching signal.
- Fitness calculation in matlab: during optimization QCOptimizer set model to the state being evaluated and calls matlab function specified in configuration. Function should evaluate model quality (typically using external link to do some calculations) and return fitness value. To work in this mode matlab should be available and one should create three functions and set it names to following parameters:
    - Initialize session command: called by QCOptimizer when optimization is started. Should perform initialization, for example load some simulink model.
    - Fitness calculation command: called by QCOptimizer to calculate fitness for current state.
    - Close session command: called when optimization is finished. May free resources allocated for calculations.
- Fitness calculation from external signal. Signal coming from external link is used for model input and fitness calculation. Concrete fitness function is configured during optimization start, but here you can enter initial conditions and calculation time. Initial conditions are set in `Initial state` table where you can enter desired initial value for some variables and possible delta. In example shown on figure above optimization will start when output 1 of quantum block 432 will equal 1±2. Once initial condition is met QCOptimizer will wait for delay time set in wait before averaging field and than calculate fitness function as average over averaging time selected.

In this window you can also select what QCOptimizer will do if it detects that the same model state evaluation is requested during single optimization process. If `do not check for duplicate chromosomes` is selected than duplicate detection will be disabled and all states will be evaluated. You can greatly improve optimization time by selecting single evaluation mode. In this case you may also want to set hash table size used to store processed states.

4.2.12.2 Starting optimization

To start optimization press  button. Optimization start dialog will appear:

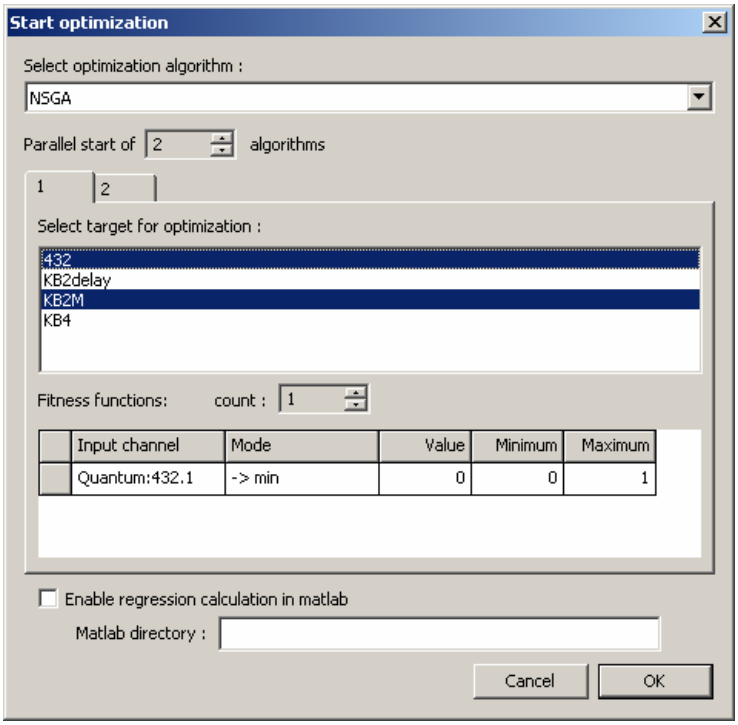

Figure 4.2.12.2.1 Optimization start dialog.

First select optimization algorithm from list. Blocks available for optimization are shown in optimization target lists. Select blocks to be optimized by clicking on them.

Below optimization target list there is a fitness function configuration table. In “fitness calculation in Matlab” mode those settings are not used. For NSGA algorithm you can use more than one fitness function. In the table select desired singal, optimization mode (for minimum, for maximum, or to match a value).

You can start several instances of the same algorithm operating over different blocks by setting parallel start option to desired number. Several pages will appear where you can select targets for all algorithms.

Once you press OK you will be prompted to enter algorithm parameters (depending on algorithm selection). When you confirm algorithm settings optimization will start.

4.2.12.3 Optimization progress/evaluation

When optimization is active optimization status dialog is shown:

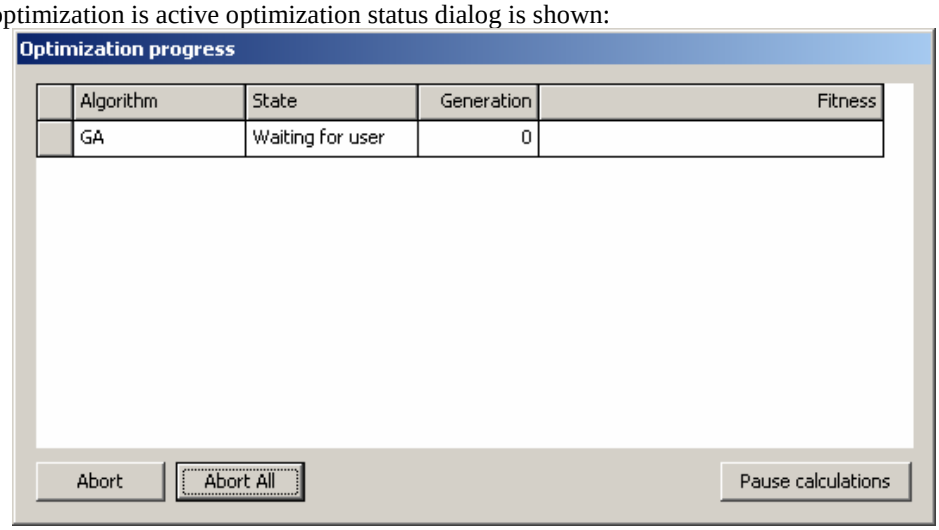

Figure 4.2.12.3.1 Optimization progress dialog.

For each optimization currently active it displays a line, which shows current status, generation and fitness achieved. You can abort one of algorithms by selecting it and pressing `Abort` button. Pressing `Abort All` will abort all active algorithms. Algorithms may be aborted with some delay, if there is a calculation which cannot be stopped immediately.

Press `Pause calculations` to pause all calculations. Current evaluations will be restarted from beginning after calculation is resumed.

When calculation is over (or after each generation if you select corresponding option in algorithm configuration) the dialog will appear showing state of current population:

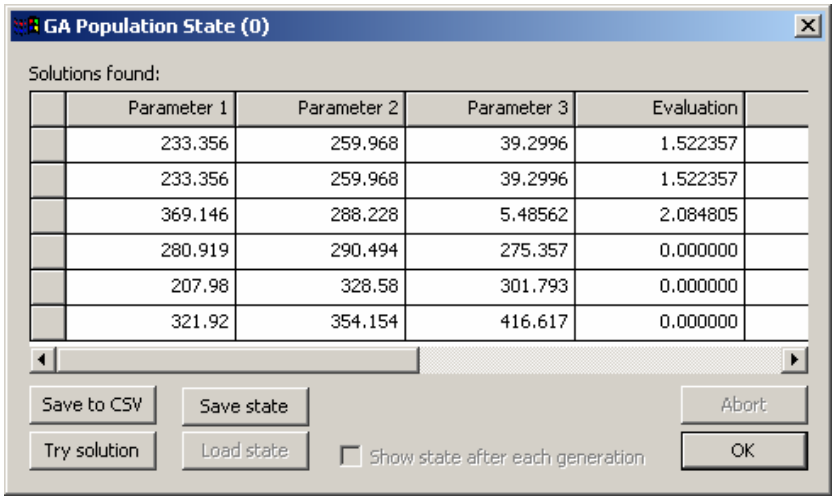

Figure 4.2.12.3.2 Optimization progress dialog.

The table lists all parameters for all chromosomes, as well as evaluation, fitness and other algorithm-dependent parameters. Top line shows best chromosome available, next lists chromosomes of current generation. You can save this data to text (CSV) file by pressing `Save to CSV` button. Select chromosme and press `Try solution` to set model state to chromosme selected, so that you can evaluate it manually.

If this dialog is shown after optimization is over, you should select a chromosome which will be used as optimization result (by default best one shown in top line is used). When you are finished this chromosome will be applied to model.

5 SCLib interface for Matlab fitness function calculation

QCOptimizer support the same external link as SCOptimizer. It is typically used when you use Matlab as source of teaching data for genetic algorithms. Following SCLib functions should be used to interface with QCOptimizer:

- BOOL GAConnect(LPCTSTR param) – called to establish connection with QCOptimizer process. Param is a text string passed by QCOptimizer as parameter of Initialize Session command. This function should be called before first call to GAInfer().
- FloatVector & GAInfer(FloatVector &in) – performs inference using current QCOptimizer state.
- void GADisconnect() – frees resources associated with QCOptimizer link. You should call this function once for each call of GAConnect().

Simulink versions of those functions are also available:

- BOOL SimGAConnect(LPCTSTR param, SimStruct *S);
- FloatVector & SimGAInfer(FloatVector &in, SimStruct *S, bool all);
- void SimGADisconnect(SimStruct *S);

Other calls are not supported by QCOptimizer, see SCOptimizer documentation for description of other functions.

6 Appendix 1 – Supported MF distributions

SCOptimizer supports following fuzzy membership function shapes:

- Exact numbers
  MF of exact numbers equals 1 at some value and 0 in all other cases.
  Exact numbers are written like they are: 1 (exact number “one”).
- Triangular
  MF of triangular distribution equals to 1 at modal value and linearly decreases to 0 at modal value – left_fuzzy and modal value + right_fuzzy points.
  Triangular distribution is written in the following form: tr(modal value; left_fuzzy;right_fuzzy). Example: tr(0;1.5;1.3).
- Trapezium
  MF of trapezium distribution equals 1 at interval [left_tolerant,right_tolerant] and linearly decreases to 0 at left_tolerant-left_fuzzy and right_tolerant+right_fuzzy points.
  Trapezium distribution is written as tp(left_tolerant;right_tolerant;left_fuzzy;right_fuzzy). Example: tp(0;1;0.5;0.5).
- Descending
  MF of descending distribution equals to 1 at all points less then or equal to modal value, then it linearly decrease to 0 at modal value+fuzzy point and remains 0 for greater values.
  Descending distribution format is: ds(modal_value;fuzzy). Example: ds(1;0.5).
- Ascending
  MF of ascending distribution equals 0 at points less than modal_value-fuzzy, then it linearly increase and reach 1 at modal_value. At points greater than modal_value it equals 1.
  Ascending distribution format is: as(modal_value;fuzzy). Example: as(0;0.3).
- Normal (Gaussian)
  MF of normal distribution is changed according to Gaussian function: $\exp(-9*(x-\text{modal\_value})^2/(2*\text{fuzzy}^2)$
  Normal distribution format is n(modal_value;fuzzy). Example: n(0;1).
- Asymmetrical normal
  Asymmetrical normal distribution has a shape of Gaussian function with different scale parameters to the left and to the right from modal value.
  Format for asymmetrical normal distribution is: asymn(modal_value;left;right).
  Example: asymn(0;1.1;0.9).
- Normal descending
  MF of descending normal distribution equals to 1 at all points less than or equal to modal value, then it have shape of gaussian function.
  Descending normal distribution format is: descn(modal_value;fuzzy). Example: descn(1;0.5).
- Normal ascending
  MF of ascending normal distribution equals to 1 at all points greater than or equal to modal value, at lower values it have a shape of Gaussian function.
  Ascending normal distribution format is: ascn(modal_value;fuzzy). Example: ascn(1;0.5).

Please note, that numbers should be written in the format, specified for model. Examples are given for the case when decimal separator is dot (.) and list separator is semicolon (;). See **changing locale settings** section for more details.