

fzymodel

— Version 2.9 as of 29 July 2020 —

A neuro-fuzzy modeling tool

Manfred Männle, 2020

Contents

1	License	4
2	Introduction	5
3	Installation	6
4	Code description	7
4.1	main	8
4.2	Real	9
4.3	mode_type	9
4.4	Global constants	9
4.5	Fuzzy Set constants	14
4.6	Global variables	19
4.6.1	Options and settings for all modes	20
4.6.2	Options for mode == MODELING	24
4.6.3	Options for mode == PRINT_SETS	32
4.6.4	Options for mode == ESTIMATION or SIMULATION.	34
4.6.5	Options for mode == SIMULATION.	36
4.6.6	Options for mode == MAKE_DATA.	36
4.6.7	Options for mode == NORMALIZE.	38
4.7	Help functions	39
4.8	Error and exception handling	41
4.8.1	Error	41
4.8.2	FileOpenError	42
4.8.3	IncompleteCoverageError	43
4.8.4	ConsdimError	43
4.8.5	ParseError	44
4.8.6	SystemCallError	44
4.8.7	SVDError	44
4.8.8	FunctionUndefinedError	45
4.9	Main modules	45
4.10	Fuzzy Set funtions	47
4.11	FSet	49
4.14	FRule	50
4.17	FModel	53
4.20	Data	67
	Class Graph	75

Takagi/Sugeno/Kang Fuzzy Modeling

`fzmodel` is a collection of tools for fully automatic modeling of Takagi/Sugeno/Kang-type neuro-fuzzy models (systems identification), normalization of data sets, and systems simulation.

Here is a list of highlights:

- sigmoidal or trapezoidal fuzzy sets
- RPROP and SVD parameter optimization (deep learning), heuristic search of fuzzy rules (`fzmodel`)
- estimation, i.e. one-step prediction (`fzyestim`)
- simulation, i.e. multiple step prediction (`fzysimul`)
- LaTeX output of fuzzy rules, graphical visualization of fuzzy sets (`fzy2sets`)
- normalization of input data (`fzymkdat`)

For an introduction to the philosophy of `fzmodel` go on reading the Introduction.

License

MIT License

Copyright (c) 1999, 2020 Manfred Maennle

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Introduction

The `fzmodel` package provides a set of tools to build and evaluate TSK-type neuro-fuzzy model.

In order to make investigations and results easier to reproduce it is recommended to use `fzmodel` and its derivatives through shell scripts or similar. Therefore, all the tools are command line based and provide options to set relevant parameters.

For a usage description type `fzmodel -h`, `fzyestim -h`, `fzysimul -h`, `fzymkdat -h`, or `fzy2sets -h`.

The algorithm for structure and parameter optimization (deep learning) is described in my scientific papers and PhD thesis.

Installation

For installation go to the `src` directory, edit the `Makefile` (in particular the destination path), and type `make install`.

The installation is tested for Sun Solaris and Linux. You must use GNU `make`, `gcc` is recommended. You need to have the TNT (template numerical toolkit) library installed; set the include path `INCPATH` accordingly. You need to have `flex` and `bison` properly installed.

To remake the documentation you need `doc++` version 3.3 or later and `pdflatex`. For this, go to the `src` directory and type `make doc`, then go to the `doc/tex` directory and check `doc.pdf`.

If you encounter problems during installation or usage, contact the author at `fzymodel@maennle.org`.

The source code is split into the files:

- `global.{hh|cc}`: command line options and global variables and functions
- `main.{hh|cc}`: parse command line, print help, catch exceptions
- `fzymodel.{hh|cc}`: TSK modeling
- `fzyestim.{hh|cc}`: estimation, one-step prediction
- `fzysimul.{hh|cc}`: simulation, multiple step prediction
- `fzy2sets.{hh|cc}`: print tex description of a model
- `fzymkdat.{hh|cc}`: prepare (normalize) training/validation data
- `fmodel.{hh|cc}`: fuzzy model class
- `data.{hh|cc}`: training/validation data class
- `svd.{hh|cc}`: singular value decomposition
- `data_lex.{l|h}`: parse data files
- `fzy_prs.y`, `fzy_lex.{l|h}`: parse fuzzy model files (`.fzy`)

Code description

Names

4.1		main	8
4.2	typedef	Real	9
4.3	enum	mode_type	9
4.4		Global constants	9
4.5		Fuzzy Set constants	14
4.6		Global variables	19
4.7		Help functions	39
4.8		Error and exception handling	41
4.9		Main modules	45
4.10		Fuzzy Set funtions	47
4.11	class	FSet	49
4.12	typedef	Premise	50
4.13	typedef	Consequence	50
4.14	class	FRule	50
4.15	typedef	FSetContainer	52
4.16	typedef	FRuleContainer	53
4.17	class	FModel : public Function	53
4.18	#define	DATA_HH <i>we only have one function to minimize</i>	66
4.19	typedef	Uvector	67
4.20	class	Data	67
4.21	void	SVD (const Matrix<Real>& A, Matrix<Real>& Q, Matrix<Real>& U, Matrix<Real>& V) throw(SVDError)	74

4.1

main**Names**

4.1.1	int	main (int argc, char* argv[], char* env[])	8
4.1.2	void	print_options (ofstream& strm, int argc, char* argv[], char* env[])	8

4.1.1

```
int main (int argc, char* argv[], char* env[])
```

The main program.

Parses command line, checks command line arguments and then calls the appropriate module (e.g., fzymodel, fzyestim, etc.).

Return Value: The return status of the modeling/simulation (0 = ok, else error).

Parameters: **argc** The number of commandline parameters.

argv The commandline parameters.

Author: Manfred Männle

Version: *Revision : 2.9 Date : 2020 - 08 - 29 10 : 47 : 39 + 00 : 00 : 02*

4.1.2

```
void print_options (ofstream& strm, int argc, char* argv[],  
                     char* env[])
```

Print the command line arguments to a stream.

Parameters:

strm	output file stream
argc	no. of command line arguments
argv	command line argument vector
env	environment vector

4.2

`typedef Real`

The type `Real` is determined by the definition of either `LONGDOUBLE`, `DOUBLE`, or `FLOAT`. The constants `REAL_MAX`, `REAL_MIN`, `REAL_EPSILON`, and `REAL_NAME` are then set accordingly to the type of `Real`.

4.3

`enum mode_type`

Mode enumeration: `UNDEFD_MODE` == undefined mode; (initial value)
`MODELING` == build fuzzy model based on training and validation data
`ESTIMATION` == one-step prediction using data from file
`SIMULATION` == multiple step prediction using data from file
`PRINST_SETS` == print LaTeX and graphical description of a fuzzy model
`MAKE_DATA` == prepare data for fuzzy modeling

4.4

Global constants

Names

- 4.4.1 extern const char*
 model_prefix *prefix of fuzzy model output file* 10
- 4.4.2 extern const char*
 fzy2pixl_prefix *prefix of pixel output file* 11
- 4.4.3 extern const char*

	data_prefix	<i>prefix of data generation file</i>	11
4.4.4	extern const char*	estimation_prefix	<i>prefix of estimation output file</i>	.. 11
4.4.5	extern const char*	simulation_prefix	<i>prefix of simulation output file</i>	.. 11
4.4.6	extern const char*	model_suffix	<i>suffix of fuzzy model output file</i>	. 12
4.4.7	extern const char*	r2_suffix	<i>suffix of r2 output file</i> 12
4.4.8	extern const char*	error_suffix	<i>suffix of error output file</i> 12
4.4.9	extern const char*	log_suffix	<i>suffix of logfile</i> 12
4.4.10	extern const char*	output_suffix	<i>suffix of output file</i> 13
4.4.11	extern const char*	pixl_suffix	<i>suffix of pixel file (for graphical description of a fuzzy set)</i> 13
4.4.12	extern const char*	eps_suffix	<i>suffix of eps file (graphical eps description of a fuzzy set)</i> 13
4.4.13	extern const char*	tex_suffix	<i>suffix of LaTeX file (LaTeX description of a fuzzy model)</i> 13
4.4.14	extern const char*	release	<i>software release number</i> 14
4.4.15	extern const char*	release_date	<i>software release date</i> 14

4.4.1

extern const char* **model_prefix**

prefix of fuzzy model output file

prefix of fuzzy model output file

4.4.2

```
extern const char* fzy2pixl_prefix
```

prefix of pixel output file

prefix of pixel output file

4.4.3

```
extern const char* data_prefix
```

prefix of data generation file

prefix of data generation file

4.4.4

```
extern const char* estimation_prefix
```

prefix of estimation output file

prefix of estimation output file

4.4.5

```
extern const char* simulation_prefix
```

prefix of simulation output file

prefix of simulation output file

4.4.6

`extern const char* model_suffix`

suffix of fuzzy model output file

suffix of fuzzy model output file

4.4.7

`extern const char* r2_suffix`

suffix of r2 output file

suffix of r2 output file

4.4.8

`extern const char* error_suffix`

suffix of error output file

suffix of error output file

4.4.9

`extern const char* log_suffix`

suffix of logfile

suffix of logfile

4.4.10

`extern const char* output_suffix`

suffix of output file

suffix of output file

4.4.11

`extern const char* pixl_suffix`

suffix of pixel file (for graphical description of a fuzzy set)

suffix of pixel file (for graphical description of a fuzzy set)

4.4.12

`extern const char* eps_suffix`

suffix of eps file (graphical eps description of a fuzzy set)

suffix of eps file (graphical eps description of a fuzzy set)

4.4.13

`extern const char* tex_suffix`

suffix of LaTeX file (LaTeX description of a fuzzy model)

suffix of LaTeX file (LaTeX description of a fuzzy model)

4.4.14

```
extern const char* release
```

software release number

software release number

4.4.15

```
extern const char* release_date
```

software release date

software release date

4.5**Fuzzy Set constants****Names**

4.5.1	const Real	mu_left	<i>definition of right border of input space</i>	15
4.5.2	const Real	mu_right	<i>definition of left border of input space</i>	15
4.5.3	const Real	cons_0	<i>initialize consequence parameters c</i>	16
4.5.4	const Real	delta_mu_0	<i>initialize delta μ</i>	16
4.5.5	const Real	delta_cons_0	<i>initialize delta c</i>	16
4.5.6	const Real	sigma_0	<i>initialize σ</i>	16
4.5.7	const Real	delta_sigma_0	<i>initialize delta σ</i>	17
4.5.8	const Real	max_delta_mu	<i>upper border of delta μ</i>	17
4.5.9	const Real	max_delta_cons	<i>upper border of delta c</i>	17
4.5.10	const Real	max_delta_sigma		

			<i>upper border of delta σ</i>	17
4.5.11	const Real	min_delta_mu	<i>lower border of delta μ</i>	18
4.5.12	const Real	min_delta_cons	<i>lower border of delta c</i>	18
4.5.13	const Real	min_delta_sigma	<i>lower border of delta σ</i>	18
4.5.14	const Real	max_sigma_init	<i>upper border of σ</i>	18
4.5.15	const Real	min_sigma_init	<i>lower border of σ</i>	19
4.5.16	const Real	eta_plus	<i>RPROP step lengthening for successful step</i>	19
4.5.17	const Real	eta_minus	<i>RPROP step shortening for unsuccessful step</i>	19

Constants for Fuzzy Sets.

4.5.1

const Real **mu_left**

definition of right border of input space

definition of right border of input space

4.5.2

const Real **mu_right**

definition of left border of input space

definition of left border of input space

4.5.3

`const Real cons_0`

initialize consequence parameters c

initialize consequence parameters c

4.5.4

`const Real delta_mu_0`

initialize delta μ initialize delta μ **4.5.5**

`const Real delta_cons_0`

initialize delta c

initialize delta c

4.5.6

`const Real sigma_0`

initialize σ initialize σ

4.5.7

`const Real delta_sigma_0`

initialize delta σ initialize delta σ **4.5.8**

`const Real max_delta_mu`

upper border of delta μ upper border of delta μ **4.5.9**

`const Real max_delta_cons`

upper border of delta c upper border of delta c **4.5.10**

`const Real max_delta_sigma`

upper border of delta σ upper border of delta σ

4.5.11

`const Real min_delta_mu`

lower border of delta μ lower border of delta μ **4.5.12**

`const Real min_delta_cons`

lower border of delta c

lower border of delta c

4.5.13

`const Real min_delta_sigma`

lower border of delta σ lower border of delta σ **4.5.14**

`const Real max_sigma_init`

upper border of σ upper border of σ

4.5.15

const Real **min_sigma_init**

lower border of σ

lower border of σ

4.5.16

const Real **eta_plus**

RPROP step lengthening for successful step

RPROP step lengthening for successful step

4.5.17

const Real **eta_minus**

RPROP step shortening for unsuccessful step

RPROP step shortening for unsuccessful step

4.6

Global variables

Names

4.6.1	Options and settings for all modes	20
4.6.2	Options for mode == MODELING	24
4.6.3	Options for mode == PRINT_SETS	32
4.6.4	Options for mode == ESTIMATION or SIMULATION.	

	34
4.6.5	Options for mode == SIMULATION.	36
4.6.6	Options for mode == MAKE_DATA.	36
4.6.7	Options for mode == NORMALIZE.	38

4.6.1

Options and settings for all modes

Names

4.6.1.1	extern mode_type	mode	<i>the mode (modeling, estimation, simulation, etc)</i>	21
4.6.1.2	extern char*	prgname	<i>program name</i>	21
4.6.1.3	extern char*	binpath	<i>binary path</i>	21
4.6.1.4	extern int	tracelevel	<i>trace level</i>	21
4.6.1.5	extern int	verbose	<i>verbose level</i>	22
4.6.1.6	extern int	quiet	<i>inhibit verbose output</i>	22
4.6.1.7	extern string	basefilename	<i>basic filename of output files</i> ...	22
4.6.1.8	extern string	filename_extension	<i>output files' filename extension</i> .	22
4.6.1.9	extern string	tracefilename	<i>name of trace file</i>	23
4.6.1.10	extern ofstream	tracefile	<i>trace log stream</i>	23
4.6.1.11	extern string	logfilename	<i>name of verbose log file</i>	23
4.6.1.12	extern ofstream	logfile	<i>verbose log stream</i>	23
4.6.1.13	extern int	denormalize	<i>take back normalization</i>	24

4.6.1.1

extern mode_type **mode**

the mode (modeling, estimation, simulation, etc)

the mode (modeling, estimation, simulation, etc)

4.6.1.2

extern char* **prgname**

program name

program name

4.6.1.3

extern char* **binpath**

binary path

binary path

4.6.1.4

extern int **tracelevel**

trace level

trace level

4.6.1.5

extern int **verbose**

verbose level

verbose level

4.6.1.6

extern int **quiet**

inhibit verbose output

inhibit verbose output

4.6.1.7

extern string **basefilename**

basic filename of output files

basic filename of output files

4.6.1.8

extern string **filename_extension**

output files' filename extension

output files' filename extension

4.6.1.9

extern string **tracefilename**

name of trace file

name of trace file

4.6.1.10

extern ofstream **logfile**

trace log stream

trace log stream

4.6.1.11

extern string **logfilename**

name of verbose log file

name of verbose log file

4.6.1.12

extern ofstream **logfile**

verbose log stream

verbose log stream

4.6.1.13

extern int **denormalize**

take back normalization

take back normalization

4.6.2

Options for mode == MODELING

Names

4.6.2.1	extern algo_type	optimization	<i>optimization algorithm</i>	26
4.6.2.2	extern size_t	consequence_dimension	<i>dimension of fuzzy rules' consequences</i>	26
4.6.2.3	extern int	norm	<i>error norm (eg, 1 == L1-norm, 2 == L2-norm)</i>	26
4.6.2.4	extern int	local_cons_optimization	<i>local consequence optimization</i> ..	26
4.6.2.5	extern int	shortcut	<i>use shortcut for Kang's heuristic search</i>	27
4.6.2.6	extern int	reset_cons	<i>reset consequence parameters</i> ..	27
4.6.2.7	extern int	update_premise	<i>update premise when optimizing</i>	27
4.6.2.8	extern size_t	max_n_rules	<i>limit no.</i>	27
4.6.2.9	extern size_t	min_n_rules	<i>minimal no.</i>	28
4.6.2.10	extern int	adjacent_equal_mu	<i>same mu of adjacent fuzzy sets (ie, input space divisions)</i>	28
4.6.2.11	extern size_t			

	steps_per_validation	<i>take mean of several optimization iterations before validation</i>	28
4.6.2.12	extern int consequence_optimize_SVD	<i>optimize the consequence of each new candidate model by SVD</i> ...	28
4.6.2.13	extern size_t max_opt_iterations_parallel	<i>maximal no.</i>	29
4.6.2.14	extern size_t min_opt_iterations	<i>minimal no.</i>	29
4.6.2.15	extern size_t max_opt_iterations	<i>maximal no.</i>	29
4.6.2.16	extern Real max_sigma	<i>maximal sigma of fuzzy sets</i>	29
4.6.2.17	extern Real min_sigma	<i>minimal sigma of fuzzy sets</i>	30
4.6.2.18	extern Real R2_improvement	<i>terminate modeling when R2 improvement is below</i>	30
4.6.2.19	extern Real best_R2_improvement	<i>terminate modeling when best model's R2 improvement is below</i>	30
4.6.2.20	extern int optimize_epoch_best	<i>no.</i>	30
4.6.2.21	extern int optimize_global_best	<i>no.</i>	31
4.6.2.22	extern Real alpha	<i>gradient descent learning rate</i> ..	31
4.6.2.23	extern Real beta	<i>gradient descent momentum</i>	31
4.6.2.24	extern algo_type parallel_optimization	<i>algorithm for parameter optimization (fine tuning) using parallel model</i>	31

4.6.2.1extern algo_type **optimization***optimization algorithm*

optimization algorithm

4.6.2.2extern size_t **consequence_dimension***dimension of fuzzy rules' consequences*

dimension of fuzzy rules' consequences

4.6.2.3extern int **norm***error norm (eg, 1 == L1-norm, 2 == L2-norm)*

error norm (eg, 1 == L1-norm, 2 == L2-norm)

4.6.2.4extern int **local_cons_optimization***local consequence optimization*

local consequence optimization

4.6.2.5

extern int **shortcut**

use shortcut for Kang's heuristic search

use shortcut for Kang's heuristic search

4.6.2.6

extern int **reset_cons**

reset consequence parameters

reset consequence parameters

4.6.2.7

extern int **update_premise**

update premise when optimizing

update premise when optimizing

4.6.2.8

extern size_t **max_n_rules**

limit no.

limit no. of fuzzy rules

4.6.2.9

```
extern size_t min_n_rules
```

minimal no.

minimal no. of fuzzy rules

4.6.2.10

```
extern int adjacent_equal_mu
```

same mu of adjacent fuzzy sets (ie, input space divisions)

same mu of adjacent fuzzy sets (ie, input space divisions)

4.6.2.11

```
extern size_t steps_per_validation
```

take mean of several optimization iterations before validation

take mean of several optimization iterations before validation

4.6.2.12

```
extern int consequence_optimize_SVD
```

optimize the consequence of each new candidate model by SVD

optimize the consequence of each new candidate model by SVD

4.6.2.13extern size_t **max_opt_iterations_parallel***maximal no.*

maximal no. of optimization iterations for parallel tuning

4.6.2.14extern size_t **min_opt_iterations***minimal no.*

minimal no. of optimization iterations

4.6.2.15extern size_t **max_opt_iterations***maximal no.*

maximal no. of optimization iterations

4.6.2.16extern Real **max_sigma***maximal sigma of fuzzy sets*

maximal sigma of fuzzy sets

4.6.2.17extern Real **min_sigma***minimal sigma of fuzzy sets*

minimal sigma of fuzzy sets

4.6.2.18extern Real **R2_improvement***terminate modeling when R2 improvement is below*

terminate modeling when R2 improvement is below

4.6.2.19extern Real **best_R2_improvement***terminate modeling when best model's R2 improvement is below*

terminate modeling when best model's R2 improvement is below

4.6.2.20extern int **optimize_epoch_best***no.*

no. of additional optimization iterations at each epoch's best model

4.6.2.21

```
extern int optimize_global_best
```

no.

no. of additional optimization iterations at the globally best model

4.6.2.22

```
extern Real alpha
```

gradient descent learning rate

gradient descent learning rate

4.6.2.23

```
extern Real beta
```

gradient descent momentum

gradient descent momentum

4.6.2.24

```
extern algo_type parallel_optimization
```

algorithm for parameter optimization (fine tuning) using parallel model

algorithm for parameter optimization (fine tuning) using parallel model

4.6.3

Options for mode == PRINT_SETS

Names

4.6.3.1	extern size_t	n_pixels	<i>no.</i>	32
4.6.3.2	extern int	global_fset_value	<i>normalize each fset value as $F(x)/(\sum F_i(x))$</i>	32
4.6.3.3	extern int	scale	<i>scale of y-axis</i>	33
4.6.3.4	extern int	tex_precision	<i>precision of numbers to beeing output in the TeX file</i>	33
4.6.3.5	extern int	execute_gnuplot_system_call	<i>execute gnuplot system call; 0==no, 1==yes</i>	33
4.6.3.6	extern string	gnuplot_file_extension	<i>gnuplot script file extension, usually "gpl"</i>	33

4.6.3.1

extern size_t **n_pixels**

no.

no. of interpolation pixels for fuzzy sets' graphical output

4.6.3.2

extern int **global_fset_value**

normalize each fset value as $F(x)/(\sum F_i(x))$

normalize each fset value as $F(x)/(\sum F_i(x))$

4.6.3.3

extern int **scale**

scale of y-axis

scale of y-axis

4.6.3.4

extern int **tex_precision**

precision of numbers to beeing output in the TeX file

precision of numbers to beeing output in the TeX file

4.6.3.5

extern int **execute_gnuplot_system_call**

execute gnuplot system call; 0==no, 1==yes

execute gnuplot system call; 0==no, 1==yes

4.6.3.6

extern string **gnuplot_file_extension**

gnuplot script file extension, usually "gpl"

gnuplot script file extension, usually "gpl"

4.6.4

Options for mode == ESTIMATION or SIMULATION.

Names

4.6.4.1	extern Real	error_offset	<i>error offset (only added when printing the RMS into a file) ...</i>	34
4.6.4.2	extern int	page_hinkley	<i>perform Page-Hinkley test: first column increase, second decrease</i>	34
4.6.4.3	extern Real	page_hinkley_mu_0	<i>mean value</i>	35
4.6.4.4	extern Real	page_hinkley_nu_inc	<i>increase step to detect</i>	35
4.6.4.5	extern Real	page_hinkley_nu_dec	<i>decrease step to detect</i>	35
4.6.4.6	extern Real	page_hinkley_lambda	<i>sensitivity lambda</i>	35

4.6.4.1

extern Real **error_offset**

error offset (only added when printing the RMS into a file)

error offset (only added when printing the RMS into a file)

4.6.4.2

extern int **page_hinkley**

perform Page-Hinkley test: first column increase, second decrease

perform Page-Hinkley test: first column increase, second decrease

4.6.4.3

extern Real **page_hinkley_mu_0**

mean value

mean value

4.6.4.4

extern Real **page_hinkley_nu_inc**

increase step to detect

increase step to detect

4.6.4.5

extern Real **page_hinkley_nu_dec**

decrease step to detect

decrease step to detect

4.6.4.6

extern Real **page_hinkley_lambda**

sensitivity lambda

sensitivity lambda

4.6.5

Options for mode == SIMULATION.

Names

4.6.5.1 extern int **order** *order of y (y-1, y-2, , y-order)* 36

4.6.5.1

extern int **order**

order of y (y-1, y-2, , y-order)

order of y (y-1, y-2, , y-order)

4.6.6

Options for mode == MAKE_DATA.

Names

4.6.6.1 extern int **n_training_data** *no.* 37
 4.6.6.2 extern int **n_skip_data** *no.* 37
 4.6.6.3 extern vector<int>
 dynamic_orders *dynamic orders of u and y* 37
 4.6.6.4 extern int **ut_regressor** *start with u(t) as regressor (in-
 stead of u(t-1))* 37

4.6.6.1

```
extern int n_training_data
```

no.

no. of learning/training patterns

4.6.6.2

```
extern int n_skip_data
```

no.

no. patterns to skip at the beginning

4.6.6.3

```
extern vector<int> dynamic_orders
```

dynamic orders of u and y

dynamic orders of u and y

4.6.6.4

```
extern int ut_regressor
```

*start with $u(t)$ as regressor (instead of $u(t-1)$)*start with $u(t)$ as regressor (instead of $u(t-1)$)

4.6.7

Options for mode == NORMALIZE.

Names

4.6.7.1	extern int	scale_only_used_data	<i>compute scale by using only some data, instead of by the whole set</i>	38
4.6.7.2	extern int	scale_by_standard_deviation	<i>scale by standard deviation (default)</i>	38
4.6.7.3	extern int	scale_by_range	<i>scale by range</i>	39
4.6.7.4	extern Real	sigma_factor	<i>scale by (sigma_factor * stddev)</i>	39

4.6.7.1

extern int **scale_only_used_data**

compute scale by using only some data, instead of by the whole set

compute scale by using only some data, instead of by the whole set

4.6.7.2

extern int **scale_by_standard_deviation**

scale by standard deviation (default)

scale by standard deviation (default)

4.6.7.3

extern int **scale_by_range**

scale by range

scale by range

4.6.7.4

extern Real **sigma_factor**

*scale by (sigma_factor * stddev)*

scale by (sigma_factor * stddev)

4.7

Help functions

Names

4.7.1	template<class T> void		
	tracemsg (int level, const string& name,		
	const T& object)	40
4.7.2	template<class T> void		
	verbose (int level, const string& name,		
	const T& object)	40
4.7.3	inline Real		
	sign (Real x)	40
4.7.4	string itos (int i)	40
4.7.5	string dtos (double r)	41
4.7.6	string dtos2 (double r)	41

4.7.1

```
template<class T> void tracemsg (int level, const string&  
                                name, const T& object)
```

print trace information

4.7.2

```
template<class T> void verbose (int level, const string&  
                                name, const T& object)
```

print verbose information

4.7.3

```
inline Real sign (Real x)
```

signum function

4.7.4

```
string itos (int i)
```

integer to string

4.7.5

string **dtos** (double r)

double to string

4.7.6

string **dtos2** (double r)

double to string

4.8

Error and exception handling

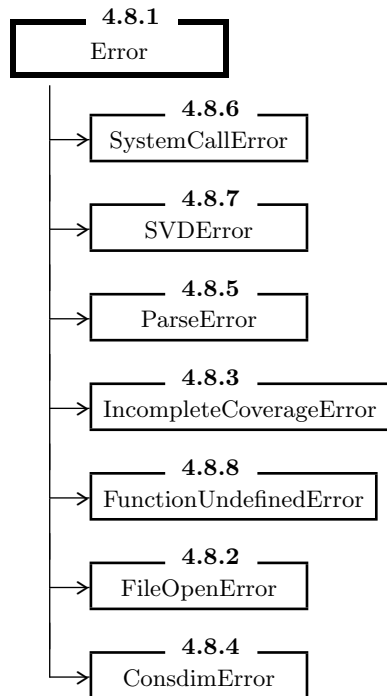
Names

4.8.1	class	Error	41
4.8.2	class	FileOpenError : public Error	42
4.8.3	class	IncompleteCoverageError : public Error	43
4.8.4	class	ConsdimError : public Error	43
4.8.5	class	ParseError : public Error	44
4.8.6	class	SystemCallError : public Error	44
4.8.7	class	SVDError : public Error	44
4.8.8	class	FunctionUndefinedError : public Error	45

4.8.1

class **Error**

Inheritance



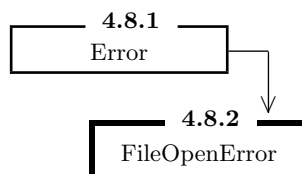
Error base class.

```

4.8.2
class FileOpenError : public Error

```

Inheritance



Error when opening a file or file not present.

4.8.3

```
class IncompleteCoverageError : public Error
```

Inheritance**4.8.1**

```
Error
```

**4.8.3**

```
IncompleteCoverageError
```

Input value found that is not covered by a model's rule.

4.8.4

```
class ConsdimError : public Error
```

Inheritance**4.8.1**

```
Error
```

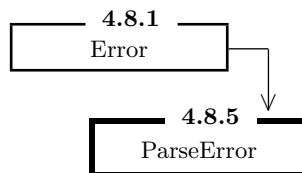
**4.8.4**

```
ConsdimError
```

Invalid consequence dimension.

4.8.5

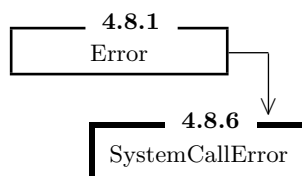
```
class ParseError : public Error
```

Inheritance

Parse error when loading a fuzzy model from file.

4.8.6

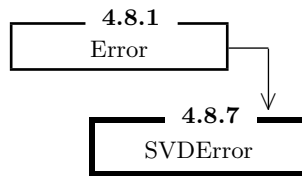
```
class SystemCallError : public Error
```

Inheritance

System call was unsuccessful.

4.8.7

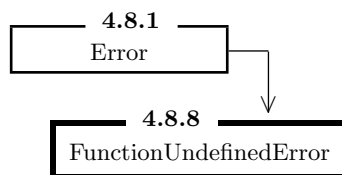
```
class SVDError : public Error
```

Inheritance

Singular Value Decomposition (SVD) unsuccessful.

4.8.8

```
class FunctionUndefinedError : public Error
```

Inheritance

Input value found where function is not defined

4.9

Main modules**Names**

4.9.1	void	fzymodel (char* learnfilename, char* validationfilename) throw(Error)	46
4.9.2	void	fzyestim (char* fzyfilename, char* datafilename) throw(Error)	

		46
4.9.3	void	fzy2sets (char* infilename, char* normfilename) throw(Error)	47
4.9.4	void	fzymkdat (char* normfilename, char* infilename, char* outfile) throw(Error)	47

4.9.1

```
void fzymodel (char* learnfilename, char* validationfile-
               name) throw(Error)
```

Build a fuzzy model, based on learning and validation data.

The output file names are created using the input file name as prefix: Writes estimation results to mod_[filename_extension]c[C]_r[R].out, best models to mod_[filename_extension]c[C]_r[R].mod, R2 to mod_[filename_extension]c[C].r2, and error to mod_[filename_extension]c[C].err

4.9.2

```
void fzyestim (char* fzyfilename, char* datafilename)
               throw(Error)
```

Write the one-step (estimation) or multiple step prediction (simulation) of a fuzzy model for a given data set

The output file names are created using the input file name as prefix: Writes estimation results to est_[filename].out or simulation results to sim_[filename].out, R2 to est_[filename_extension].r2 or sim_[filename_extension].r2, and error to est_[filename].err or sim_[filename].err.

4.9.3

```
void fzy2sets (char*  infilename,  char*  normfilename)
               throw(Error)
```

Write a LaTeX and a graphical description of a fuzzy model

Load a Fuzzy Model from file and write a LaTeX description and a graphical description (interpolation points) into files. The output file names are created using the input file name as prefix.

4.9.4

```
void fzymkdat (char*  normfilename,  char*  infilename,
               char*  outfilename) throw(Error)
```

Prepare data for fuzzy modeling (order shift and normalization).

4.10**Fuzzy Set funtions****Names**

4.10.1	inline	Real	trapezoid (Real sigma, Real u)	48
4.10.2	Real	sigmoid (Real x)	48	
4.10.3	inline	Real	t_norm (Real a, Real b)	48
4.10.4	inline	Real	inference (Real prem, Real cons)	48

4.10.1

```
inline Real trapezoid (Real sigma, Real u)
```

Membership value of trapezoidal fuzzy set.

Return Value: membership value
Parameters: sigma steepness
u position

4.10.2

```
Real sigmoid (Real x)
```

Membership value of sigmoidal fuzzy set.

Return Value: membership value
Parameters: x position

4.10.3

```
inline Real t_norm (Real a, Real b)
```

Product t-norm.

4.10.4

```
inline Real inference (Real prem, Real cons)
```

Product inference.

4.11

```
class FSet
```

Public Members

4.11.3 Real **F** (Real u) *compute the membership value* . 49

Protected Members

4.11.1 inline void **limit_delta_mu** (void)
check border of $\Delta\mu$ 49

4.11.2 inline void **limit_delta_sigma** (void)
check border of $\Delta\sigma$ 50

Fuzzy Set.

4.11.3

```
Real F (Real u)
```

compute the membership value

compute the membership value

4.11.1

```
inline void limit_delta_mu (void)
```

check border of $\Delta\mu$

check border of $\Delta\mu$

4.11.2

```
inline void limit_delta_sigma (void)
```

check border of $\Delta\sigma$

check border of $\Delta\sigma$

4.12

```
typedef Premise
```

Fuzzy Rule's Premise: a vector of pointers to Fuzzy Sets.

4.13

```
typedef Consequence
```

Fuzzy Rule's Consequence: a vector of its parameters.

4.14

```
class FRule
```

Public Members

- 4.14.2 void **copy** (const FRule& r, FSet* fset_begin,
 const FSet* r_fset_begin)
 copy a Fuzzy Rule by value 51
- 4.14.3 Real **consvalue** (const Uvector& u) const
 compute a Fuzzy Rule's conse-
 quence value for a given input vec-
 tor 51
- 4.14.4 Real **premvalue** (const Uvector& u) const

```

compute a Fuzzy Rule's premise
value for a given input vector .. 52

4.14.5 void      refine (FRule* r, size_t u, FSet* newleftfset,
                        FSet* newrightfset)
                        refine a Fuzzy Rule in uindex u;
                        save additional new rule under
                        FRule* r ..... 52

```

Protected Members

```

4.14.1 Premise    prem_ ..... 52

```

Fuzzy Rule.

4.14.2

```

void copy (const FRule& r, FSet* fset_begin, const FSet*
           r_fset_begin)

```

copy a Fuzzy Rule by value

copy a Fuzzy Rule by value

4.14.3

```

Real consvalue (const Uvector& u) const

```

compute a Fuzzy Rule's consequence value for a given input vector

compute a Fuzzy Rule's consequence value for a given input vector

4.14.4

Real **premvalue** (const Uvector& u) const

compute a Fuzzy Rule's premise value for a given input vector

compute a Fuzzy Rule's premise value for a given input vector

4.14.5

void **refine** (FRule* r, size_t u, FSet* newleftfset, FSet*
newrightfset)

refine a Fuzzy Rule in uindex u; save additional new rule under FRule r*

refine a Fuzzy Rule in uindex u; save additional new rule under FRule* r

4.14.1

Premise **prem_**

4.15

typedef **FSetContainer**

Fuzzy Set Container: a vector of Fuzzy Sets

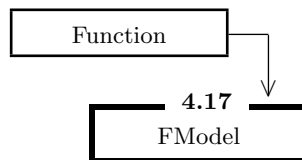
4.16

```
typedef FRuleContainer
```

Fuzzy Rule Container: a vector of Fuzzy Rules

4.17

```
class FModel : public Function
```

Inheritance**Public Members**

4.17.13	FModel ()	<i>empty model</i>	56
4.17.14	FModel (const Data& d, size_t cons_dimension)	<i>initial one-rule model</i>	56
4.17.15	void copy (const FModel& oldmodel)	<i>copy a whole model</i>	57
4.17.16	FModel (const FModel& oldmodel, size_t ruleindex, size_t uindex)	<i>refine a model's structure in rule rindex at index uindex</i>	57
4.17.17	void load (char* filename) throw(Error)	<i>load a model from file</i>	57
4.17.18	Real sum_w (const Uvector& u)	<i>w = sum(premise values)</i>	57
4.17.19	Real y_hat (const Uvector& u) throw(Error)	<i>feedforward step; returns \hat{y}</i>	58
4.17.20	void optimize_SVD (const Data& d) throw(Error)			

		<i>optimize consequence parameters using SVD</i>	58
4.17.21	void	reset_consequences (void) <i>reset consequence parameters</i> ...	58
4.17.22	void	RPROP_init (void) <i>initialize model for RPROP</i>	58
4.17.23	Real	RPROP (const Data& learndata) throw(Error) <i>one RPROP step; returns $\varepsilon = (y - \hat{y})^2$</i>	59
4.17.24	void	RPROP_backstep (void) <i>RPROP backstep: cancel last RPROP parameter update</i>	59
4.17.25	Real	optimize_RPROP (const Data& a, const Data& b, size_t min_iterations, size_t max_iterations) <i>optimize model by cross valida- tion; returns last error</i>	59
4.17.26	void	GRAD_DESCENT_init (void) <i>initialize model for GRAD_DESCENT</i>	59
4.17.27	Real	GRAD_DESCENT (const Data& learndata) throw(Error) <i>one GRAD_DESCENT step; re- turns $\varepsilon = (y - \hat{y})^2$</i>	60
4.17.28	void	GRAD_DESCENT_backstep (void) <i>GRAD_DESCENT backstep: can- cel last GRAD_DESCENT param- eter update</i>	60
4.17.29	Real	optimize_GRAD_DESCENT (const Data& a, const Data& b, size_t min_iterations, size_t max_iterations) <i>optimize model by cross valida- tion; returns last error</i>	60
4.17.30	Real	R2 (const Data& d) <i>estimation error; returns R2</i> ...	61
4.17.31	size_t	worst_rule_index (const Data& d) throw(Error)	

		<i>determine index of rule with biggest approximation error</i>	61
4.17.32	Real	estimation (const Data& d) throw(Error) <i>estimation error; returns $\varepsilon = (y - \hat{y})^2$</i>	61
4.17.33	Real	estimation (const Data& d, const char* outfilename) throw(Error) <i>write estimation into a file; re- turns estimation error</i>	61
4.17.34	Real	simulation (const Data& d) throw(Error) <i>simulation error; returns simula- tion error</i>	62
4.17.35	Real	simulation (const Data& d, const char* outfilename) throw(Error) <i>write simulation into a file; re- turns simulation error</i>	62
4.17.36	void	create_parameters (Data* learning_data, Data* validation_data) <i>prepare FModel for Minimizer</i> ..	62
4.17.37	void	erase_parameters () <i>erase parameters for Minimizer</i>	62
4.17.38	Real	calculate () <i>calculate function value (for learn- ing/training)</i>	63
4.17.39	Real	validate () <i>calculate function value (for vali- dation)</i>	63
4.17.40	void	derivate () <i>calculate derivation</i>	63
Protected Members			
4.17.1	size_t	rdim_ <i>number of rules</i>	63
4.17.2	size_t	udim_ <i>input space dimension</i>	64
4.17.3	size_t	cdim_ <i>consequence dimension</i>	64
4.17.4	size_t	sdim_ <i>number of fsets = 2 * (rdim-1)</i> .	64
4.17.5	size_t	worst_rule_ <i>index of rule with biggest error</i> .	64
4.17.6	vector<size_t>	history_ <i>history of refinements</i>	65
4.17.7	FSetContainer		

	fsets_	<i>the model's Fuzzy Sets</i>	65
4.17.8	FRuleContainer			
	frules_	<i>the models Fuzzy Rules</i>	65
4.17.9	string	learnfilename_	<i>name of the learning/training data file</i>
				65
4.17.10	string	validationfilename_	<i>name of the validation data file</i>	
				66
4.17.11	vector<ConsParam>	cons_	<i>for the usage of a Minimizer</i>	...
				66
4.17.12	inline void	limit_delta_cons	(Consequence::iterator& p)	
			<i>check border of Δc</i>
				66

Fuzzy Model.

4.17.13

FModel ()

empty model

empty model

4.17.14

FModel (const Data& d, size_t cons_dimension)

initial one-rule model

initial one-rule model

4.17.15

```
void copy (const FModel& oldmodel)
```

copy a whole model

copy a whole model

4.17.16

```
FModel (const FModel& oldmodel, size_t ruleindex, size_t  
uindex)
```

refine a model's structure in rule rindex at index uindex

refine a model's structure in rule rindex at index uindex

4.17.17

```
void load (char* filename) throw(Error)
```

load a model from file

load a model from file

4.17.18

```
Real sum_w (const Uvector& u)
```

w = sum(premise values)

w = sum(premise values)

4.17.19

`Real y_hat (const Uvector& u) throw(Error)`

feedforward step; returns \hat{y} feedforward step; returns \hat{y} **4.17.20**

`void optimize_SVD (const Data& d) throw(Error)`

optimize consequence parameters using SVD

optimize consequence parameters using SVD

4.17.21

`void reset_consequences (void)`

reset consequence parameters

reset consequence parameters

4.17.22

`void RPROP_init (void)`

initialize model for RPROP

initialize model for RPROP

4.17.23

Real **RPROP** (const Data& learndata) throw(Error)

one RPROP step; returns $\varepsilon = (y - \hat{y})^2$

one RPROP step; returns $\varepsilon = (y - \hat{y})^2$

4.17.24

void **RPROP_backstep** (void)

RPROP backstep: cancel last RPROP parameter update

RPROP backstep: cancel last RPROP parameter update

4.17.25

Real **optimize_RPROP** (const Data& a, const Data&
b, size_t min_iterations, size_t
max_iterations)

optimize model by cross validation; returns last error

optimize model by cross validation; returns last error

4.17.26

void **GRAD_DESCENT_init** (void)

initialize model for GRAD_DESCENT

initialize model for GRAD_DESCENT

4.17.27

```
Real GRAD_DESCENT (const Data& learndata)
                    throw(Error)
```

one GRAD_DESCENT step; returns $\varepsilon = (y - \hat{y})^2$

one GRAD_DESCENT step; returns $\varepsilon = (y - \hat{y})^2$

4.17.28

```
void GRAD_DESCENT_backstep (void)
```

GRAD_DESCENT backstep: cancel last GRAD_DESCENT parameter update

GRAD_DESCENT backstep: cancel last GRAD_DESCENT parameter update

4.17.29

```
Real optimize_GRAD_DESCENT (const Data& a,
                             const Data& b, size_t
                             min_iterations, size_t
                             max_iterations)
```

optimize model by cross validation; returns last error

optimize model by cross validation; returns last error

4.17.30

Real **R2** (const Data& d)

estimation error; returns R2

estimation error; returns R2

4.17.31

size_t **worst_rule_index** (const Data& d) throw(Error)

determine index of rule with biggest approximation error

determine index of rule with biggest approximation error

4.17.32

Real **estimation** (const Data& d) throw(Error)

estimation error; returns $\varepsilon = (y - \hat{y})^2$

estimation error; returns $\varepsilon = (y - \hat{y})^2$

4.17.33

Real **estimation** (const Data& d, const char* outfilename)
throw(Error)

write estimation into a file; returns estimation error

write estimation into a file; returns estimation error

4.17.34

Real **simulation** (const Data& d) throw(Error)

simulation error; returns simulation error

simulation error; returns simulation error

4.17.35

Real **simulation** (const Data& d, const char* outfilename)
throw(Error)

write simulation into a file; returns simulation error

write simulation into a file; returns simulation error

4.17.36

void **create_parameters** (Data* learning_data, Data* validation_data)

prepare FModel for Minimizer

prepare FModel for Minimizer

4.17.37

void **erase_parameters** ()

erase parameters for Minimizer

erase parameters for Minimizer

4.17.38Real **calculate** ()*calculate function value (for learning/training)*

calculate function value (for learning/training)

4.17.39Real **validate** ()*calculate function value (for validation)*

calculate function value (for validation)

4.17.40void **derivate** ()*calculate derivation*

calculate derivation

4.17.1size_t **rdim_***number of rules*

number of rules

4.17.2

size_t **udim_**

input space dimension

input space dimension

4.17.3

size_t **cdim_**

consequence dimension

consequence dimension

4.17.4

size_t **sdim_**

*number of fsets = $2 * (rdim-1)$* number of fsets = $2 * (rdim-1)$ **4.17.5**

size_t **worst_rule_**

index of rule with biggest error

index of rule with biggest error

4.17.6

vector<size_t> **history_**

history of refinements

history of refinements

4.17.7

FSetContainer **fsets_**

the model's Fuzzy Sets

the model's Fuzzy Sets

4.17.8

FRuleContainer **rules_**

the models Fuzzy Rules

the models Fuzzy Rules

4.17.9

string **learnfilename_**

name of the learning/training data file

name of the learning/training data file

4.17.10

```
string validationfilename_
```

name of the validation data file

name of the validation data file

4.17.11

```
vector<ConsParam> cons_
```

for the usage of a Minimizer

for the usage of a Minimizer

4.17.12

```
inline void limit_delta_cons (Consequence::iterator& p)
```

check border of Δc

check border of Δc

4.18

```
#define DATA_HH
```

we only have one function to minimize

we only have one function to minimize

4.19

```
typedef Uvector
```

Type of an input value vector u (u-part of one pattern).

4.20

```
class Data
```

Public Members

4.20.8	Data ()	<i>empty data</i>	68
4.20.9	Data (Subscript N, size_t M, Real x0)	<i>create data matrix filled with x0</i>	69
4.20.10	const string& filename () const	<i>filename of loaded data</i>	69
4.20.11	const vector<Uvector> & U () const	<i>u-parts of all patterns</i>	69
4.20.12	const vector<Real> & y () const	<i>y-parts of all patterns</i>	69
4.20.13	const vector<Real> & scale_factor () const	<i>scale factor</i>	70
4.20.14	const vector<Real> & scale_shift () const	<i>scale shift</i>	70
4.20.15	const size_t& blocksize () const	<i>blocksize (for twodimensional print)</i>	70
4.20.16	const Real& mean_y () const	<i>mean of all y values</i>	70
4.20.17	const Real& variance_y () const	<i>variance of all y values</i>	71
4.20.18	vector<Uvector> & U ()	<i>modify u-parts of all patterns</i>	71
4.20.19	vector<Real> &		

	y ()	<i>modify y-parts of all patterns</i>	... 71
4.20.20	void	load (char* filename) throw(Error) <i>load data matrix from file</i> 71
4.20.21	void	normalize (const vector<Real>& factor, const vector<Real>& shift) throw(Error) <i>normalize data matrix with given scale_factor and scale_shift</i> 72
Protected Members			
4.20.1	size_t	blocksize_ <i>block size for twodimensional plots</i> 72
4.20.2	vector<Uvector>	U_ <i>u-parts of the patterns</i> 72
4.20.3	vector<Real>	y_ <i>y-parts of the patterns</i> 72
4.20.4	Real	mean_y_ <i>mean of all y values</i> 73
4.20.5	Real	variance_y_ <i>variance of all y values</i> 73
4.20.6	vector<Real>	scale_factor_ 73
4.20.7	vector<Real>	scale_shift_ 73

Data container.

4.20.8

Data ()

empty data

empty data

4.20.9

Data (Subscript N, size_t M, Real x0)

create data matrix filled with x0

create data matrix filled with x0

4.20.10

const string& **filename** () const

filename of loaded data

filename of loaded data

4.20.11

const vector<Uvector> & **U** () const

u-parts of all patterns

u-parts of all patterns

4.20.12

const vector<Real> & **y** () const

y-parts of all patterns

y-parts of all patterns

4.20.13

```
const vector<Real> & scale_factor () const
```

scale factor

scale factor

4.20.14

```
const vector<Real> & scale_shift () const
```

scale shift

scale shift

4.20.15

```
const size_t& blocksize () const
```

blocksize (for twodimensional print)

blocksize (for twodimensional print)

4.20.16

```
const Real& mean_y () const
```

mean of all y values

mean of all y values

4.20.17

```
const Real& variance_y () const
```

variance of all y values

variance of all y values

4.20.18

```
vector<Uvector> & U ()
```

modify u-parts of all patterns

modify u-parts of all patterns

4.20.19

```
vector<Real> & y ()
```

modify y-parts of all patterns

modify y-parts of all patterns

4.20.20

```
void load (char* filename) throw(Error)
```

load data matrix from file

load data matrix from file

4.20.21

```
void normalize (const vector<Real>& factor, const vector<Real>& shift) throw(Error)
```

normalize data matrix with given scale_factor and scale_shift

normalize data matrix with given scale_factor and scale_shift

4.20.1

```
size_t blocksize_
```

block size for twodimensional plots

block size for twodimensional plots

4.20.2

```
vector<Uvector> U_
```

u-parts of the patterns

u-parts of the patterns

4.20.3

```
vector<Real> y_
```

y-parts of the patterns

y-parts of the patterns

4.20.4Real **mean_y_***mean of all y values*

mean of all y values

4.20.5Real **variance_y_***variance of all y values*

variance of all y values

4.20.6vector<Real> **scale_factor_**

scale factor Holds scale(u) in scale_factor_[0n_columns-1] and scale(y) in scale_factor_[n_columns].

4.20.7vector<Real> **scale_shift_**

scale shift Holds shift(u) in shift_factor_[0n_columns-1] and shift(y) in shift_factor_[n_columns].

4.21

```
void SVD (const Matrix<Real>& A, Matrix<Real>&
          Q, Matrix<Real>& U, Matrix<Real>& V)
        throw(SVDError)
```

Compute the Singular Value Decomposition (SVD) of a $m \times n$ matrix A .

This function computes the Singular Value Decomposition (SVD) of a given matrix A . Precondition: $m \geq n$. The SVD algorithm can be used to solve the least squares problem of overdetermined linear equation systems.

The algorithm is taken from: J.H. Wilkinson, C.Reinsch: Linear Algebra, Springer 1971, p. 134ff

The function computes the singular values Q and the complete orthogonal decomposition (U , V) of a real rectangular matrix A with $A = U * Q * V.transp()$; $U.transp() * U = V.transp() * V = I$.

With this, you can solve a linear least squares problem given by the m equations $y = A * p$ with $p = V * Q.inverse() * U.transp() * y$ where $Q.inverse()$ is easy to compute, since Q is a diagonal matrix.

Parameters:

- A** $m \times n$ input matrix to decompose; m rows, n columns. Precondition: $m \geq n$.
- Q** $n \times n$ diagonal matrix holding the singular values in its diagonal(return value).
- U** $m \times n$ matrix with orthonormalized columns (return value).
- V** $n \times n$ matrix with orthogonal columns (return value).

Exceptions: **SVDException** This exception will be thrown when an error occurs during the computation.

Author: Steffen Bloedt, (minor changes by Manfred Männle)

Class Graph

4.8.1 Error	41
→ 4.8.6 SystemCallError	44
→ 4.8.7 SVDError	44
→ 4.8.5 ParseError	44
→ 4.8.3 IncompleteCoverageError	43
→ 4.8.8 FunctionUndefinedError	45
→ 4.8.2 FileOpenError	42
→ 4.8.4 ConsdimError	43
4.11 FSet	49
4.14 FRule	50

Class Graph

<div>4.17</div> <div>FModel</div>	53
<div>4.20</div> <div>Data</div>	67