Fuzzy Modeling and Identification Toolbox

For Use with MATLAB

Robert Babuška

Fuzzy Modeling and Identification Toolbox User's Guide (August 1998)

Disclaimer: This software is distributed without any warranty.

License agreement:

- You are allowed to use this software for non-commercial (academic) purposes free of charge.
- You are not allowed to commercialize the software. If you want to use the software for commercial projects, an additional agreement with the author must be made.

Toolbox version 2.0
This manual was printed on August 10, 1998 (first printing).

The development of this software was in part supported by the "FAMIMO" project (Fuzzy Algorithms for the Control of Multiple-Input, Multiple-Output Processes) funded by the European Commission (Esprit LTR 21911). More information about this project is available at `http://iridia.ulb.ac.be/FAMIMO/`

Dr. Robert Babuška
Control Engineering Laboratory
Faculty of Information Technology and Systems
Delft University of Technology
Mekelweg 4, P.O. Box 5031, 2600 GA Delft
The Netherlands

Phone: +31 15 2785117 Fax: +31 15 2626738
E-mail: `r.babuska@its.tudelft.nl`
Web: `http://lcewww.et.tudelft.nl/~babuska`

Contents

Introduction

This section first gives a brief introduction to fuzzy modeling. Then the structure of the toolbox and its installation are described.

Fuzzy Modeling and Identification

Since its introduction in 1965, fuzzy set theory has found applications in a wide variety of disciplines. Modeling and control of dynamic systems belong to the fields in which fuzzy set techniques have received considerable attention, not only from the scientific community but also from industry. Many systems are not amenable to conventional modeling approaches due to the lack of precise, formal knowledge about the system, due to strongly nonlinear behavior, due to the high degree of uncertainty, or due to the time varying characteristics. Fuzzy modeling along with other related techniques such as neural networks have been recognized as powerful tools which can facilitate the effective development of models.

Fuzzy models can be seen as logical models which use "if–then" rules to establish qualitative relationships among the variables in the model. Fuzzy sets serve as a smooth interface between the qualitative variables involved in the rules and the numerical data at the inputs and outputs of the model. The rule-based nature of fuzzy models allows the use of information expressed in the form of natural language statements and consequently makes the models transparent to interpretation and analysis. At the computational level, fuzzy models can be regarded as flexible mathematical structures, similar to neural networks, that can approximate a large class of complex nonlinear systems to a desired degree of accuracy.

Recently, a great deal of research activity has focused on the development of methods to build or update fuzzy models from numerical data. In order to automatically generate fuzzy models from measurements, a comprehensive methodology is implemented in this toolbox. It employs fuzzy clustering techniques to partition the available data into subsets characterized by a linear behavior. From the obtained fuzzy partitions a multivariable model of the Takagi–Sugeno type (Takagi and Sugeno, 1985) is constructed. A detailed description of this identification method can be found in (Babuška, 1998).

About the Toolbox

The Fuzzy Modeling and Identification (FMID) toolbox is a collection of MATLAB functions for the construction of Takagi–Sugeno (TS) fuzzy models from data. The toolbox provides five categories of tools:

- `fmclust` automatically generates a TS fuzzy model from given input–output data. The parameters of the obtained models are stored in a single `Matlab` structure. This allows the user to easily manipulate, store and document fuzzy models.
- `fmsim` simulates an available fuzzy model either from the input or in a one-step-ahead mode. This function also computes the variance accounted for (VAF) performance index for the model.
- `fm2tex` exports the parameters stored in the fuzzy model structure into a LaTeX file.
- `fmidemo` opens a menu with several demonstration of static and dynamic fuzzy models. The individual demos can serve as templates for the user's own applications.
- In addition, the toolbox contains several utilities which are of little importance to the user.

In addition to these tools developed specifically for fuzzy identification, a more general Fuzzy Toolbox is available from the author. This toolbox was originally developed for MATLAB 4 and is currently being converted to MATLAB 5. Eventually, both toolboxes will be merged in one.

Installation

The installation is straightforward and it does not require any changes to your system settings. Proceed along the following four steps:

1. Create a subdirectory under your `....\MATLAB\TOOLBOX` directory, call it whatever you like, for instance `FMID`. If you are updating an existing installation of the toolbox, remove all its files first.

2. Copy the toolbox files to this directory. If the toolbox was provided as a zip file, use pkunzip to unpack it in this directory. Use the '-d' option in order to extract also the PRIVATE subdirectory which contains some utility functions used by the main toolbox routines.

3. Modify or create the MATLAB `startup.m` file to include the `FMID` directory:

```
addpath c:\MATLAB\TOOLBOX\FMID
```

4. Start MATLAB and run `fmidemo`.

New Features in Version 2.0

The main differences with respect to the previous version (1.0) are:

- The synopsis for `fmclust` has been modified. Instead of passing a large number of numerical parameters, the function is now called with three parameters which are structures. Each of them groups parameters related to a certain aspect of the model (data, parameters, dynamics).
- The internal representation of the fuzzy model structure `FM` has been modified. This implies that models generated with the previous version cannot be used and have to be generated again from data.
- Function `plotmfs` has been added. This function plots the membership functions.
- A bug has been fixed in `fmsim`.

Examples

In this section, several examples for the identification of static and dynamic systems of varying complexity are given. For additional examples, see `fmidemo`.

Approximating a Static Function

This example is implemented in `statdemo`. Let us approximate a univariate static function by a TS fuzzy model. First, prepare the structure containing the input and output data:

```
u = (0:0.02:1)';
y = sin(7*u);
Dat.U = u;
Dat.Y = y;
```

Second, choose the number of clusters (5), and the type of the antecedent (projected membership functions). For the remaining parameters, default values will be used.

```
Par.c    = 5;
Par.ante = 2;
```

Now we can call the function `fmclust` which automatically constructs the fuzzy model and returns it in the structure `FM`:

```
[FM,Mu] = fmclust(Dat,Par);
```

The output of the model can be computed for the any input data by:

```
[ym,VAF] = fmsim(u,y,FM);
```

In this case, the same input was used as was used for identification. Usually, a different data set is used for model validation (see the following example). A plot is displayed on the screen and the output of the model is returned in `ym`. The second output argument, `VAF`, is the *variance accounted for* performance index (see `vaf`). Figure 1 shows the plot produced by the `fmsim` command.

In addition to the output and the VAF index, the degree of fulfillment and the outputs of the individual rules can be obtained from `fmsim`:

```
[ym,VAF,dof,yl,ylm] = fmsim(u,y,FM);
```


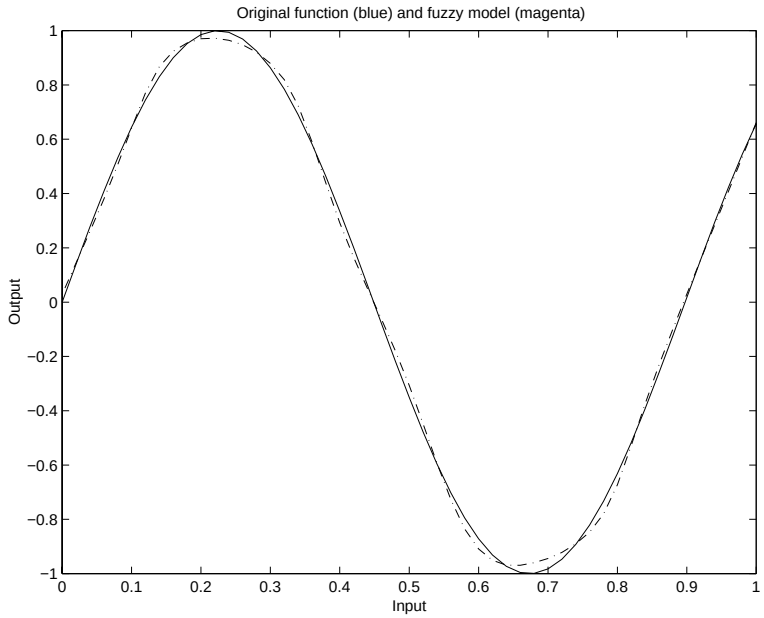

Figure 1. *Static function (solid line) and its approximation by a fuzzy model (dashed-dotted line). The fit of the model is VAF=99.67%.*

`dof` contains the degrees of fulfillment of the rules, `yl` are outputs of the individual rules. `ylm` is equal to `ym` with the exception that all outputs but the one corresponding to the largest `dof` are masked by NaN. This facilitates easy plotting of the local models. In our static SISO example, the `dof` matrix represents the membership functions evaluated for the input vector, see Fig. 2.

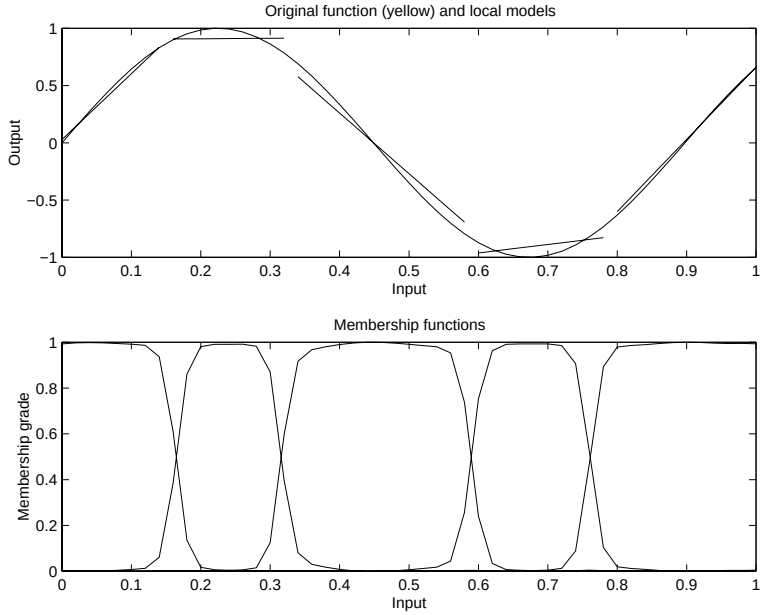

Figure 2. *The original function and the obtained local models (top), membership functions (bottom).*

Now you can investigate the contents of the `FM` structure. To display the consequent parameters, for instance, type:

```
FM.th{1}

ans =
   5.7961    0.0259
  -0.0023    0.8993
  -5.3350    2.3972
   0.5580   -1.2872
   6.1772   -5.5209
```


In a similar way, the cluster centers and other parameters can be extracted. The function `fm2tex` writes the information from `FM` to a LaTeXfile. A part of this file is included below, in order to see the rules and the cluster centers:

1. **If** u **is** A_1 **then** $y = 5.80 \cdot 10^{0} u + 2.59 \cdot 10^{-2}$
2. **If** u **is** A_2 **then** $y = -2.29 \cdot 10^{-3} u + 8.99 \cdot 10^{-1}$
3. **If** u **is** A_3 **then** $y = -5.33 \cdot 10^{0} u + 2.40 \cdot 10^{0}$
4. **If** u **is** A_4 **then** $y = 5.58 \cdot 10^{-1} u - 1.29 \cdot 10^{0}$
5. **If** u **is** A_5 **then** $y = 6.18 \cdot 10^{0} u - 5.52 \cdot 10^{0}$

rule	u
1	$7.28 \cdot 10^{-2}$
2	$2.40 \cdot 10^{-1}$
3	$4.53 \cdot 10^{-1}$
4	$6.74 \cdot 10^{-1}$
5	$8.89 \cdot 10^{-1}$

Modeling a SISO Dynamic System

In this example, we develop a simple dynamic model for the relationship between the throttle angle and the speed of an engine. Let us first define the `Par` (parameters) and `Dyn` (dynamics) structures:

```
Par.c = 3;              % number of clusters
Par.m = 2.2;            % fuzziness parameter
Par.tol = 0.01;         % termination criterion
Par.ante = 1;           % product-space MFs
Dyn.Ny = 1;             % number of lagged outputs
Dyn.Nu = 1;             % number of lagged inputs
Dyn.Nd = 1;             % number of transport delays
```

`Par.c` defines the number of required clusters, `Par.m`, and `Par.tol` are the fuzziness and the termination tolerance of the clustering algorithm, respectively. The `Par.ante` parameter specifies that product-space membership functions will be derived. `Dyn.Ny` and `Dyn.Nu` are the number of lags in the output and input, respectively, and `Dyn.Nd` is the number of pure delays from the input to the output. Thus, we will obtain a fuzzy model of the following structure: $y(k+1) = f(y(k), u(k))$. Now we load the data set, split it in halves and use the first half for identification and the second half for validation.

```
load data
N = size(engine_speed,1);
N2 = floor(size(engine_speed,1)/2);
Dat.U = throttle(1:N2);
Dat.Y = engine_speed(1:N2);
Dat.Ts = 0.1;
ue = throttle(N2+1:N);
ye = engine_speed(N2+1:N);
```

`Dat.Ts` is the sample time of the data. Now, the model can be constructed and validated by simulation:

```
FM = fmclust(Dat,Par,Dyn);
[ym,VAF] = fmsim(ue,ye,FM); VAF
```

The graphical output obtained from the `fmsim` function is shown in Fig. 3. Run `endemo1` to see this demo.

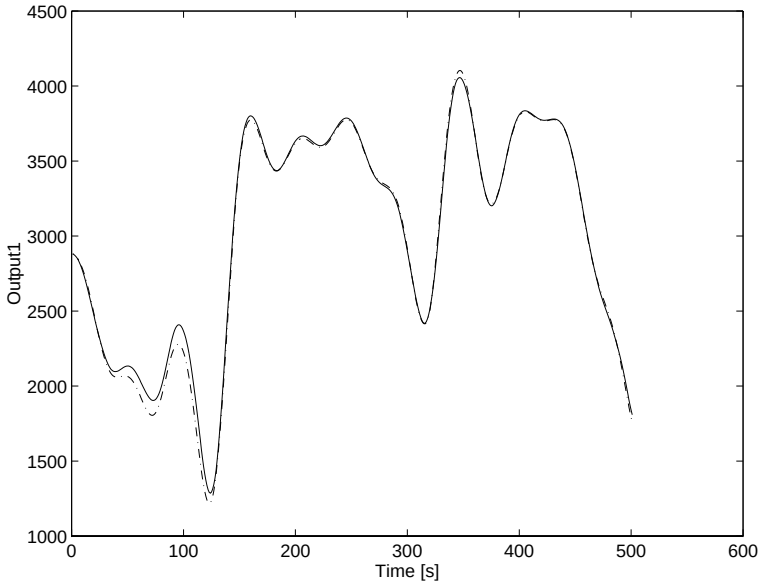

Figure 3. *Dynamic simulation of the engine (VAF=99.63%).*

Modeling a MIMO Dynamic System

Consider a MIMO process consisting of four cascaded tanks as shown in Fig. 4. The inputs are the two flow rates $\boldsymbol{u} = [Q_1, Q_2]^T$, and the outputs are the four levels $\boldsymbol{y} = [h_1, h_2, h_3, h_4]^T$.

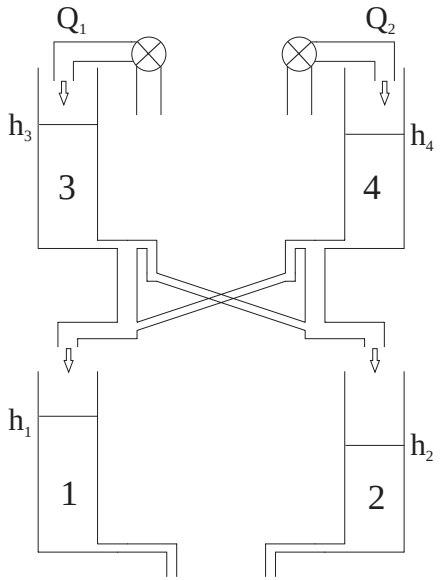

Figure 4. *Four cascaded tanks.*

A model of this system was simulated in Simulink in order to obtain input–output data sequences for identification. The input signal is a low-pass filtered normally distributed white noise to which white noise with a small amplitude is added, see Fig. 5. The low-frequency component signal drives the nonlinear system through the entire operating range, while the high-frequency component takes care for persistent local excitation.

The measured outputs are the levels in the four tanks. They are similar to signals given in Fig. 6. The number of samples available for identification is 1000 and the sample time is 10 s. The structure of the MIMO model is selected by using the

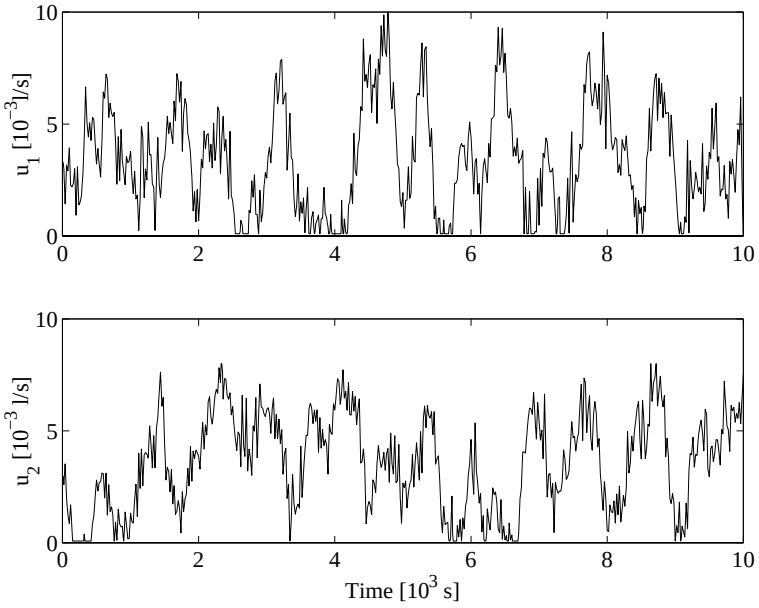

Figure 5. *Input data for identification.*

insight in the physical structure of the system as follows:

$$n_y = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; n_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \; n_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{1}$$

The first row of the n_y matrix states that level $h_1(k+1)$ depends on $h_1(k)$, $h_3(k)$ and $h_4(k)$, but not on $h_2(k)$, see Fig. 4. Similarly, the third row of this matrix states that $h_3(k+1)$ depends on $h_3(k)$, but not on the other variables. The meaning of n_u and n_d should be clear.

To construct the fuzzy model, the identification data set is first loaded and the structural parameters of the model are defined:

```
load ll4data          % load input-output data
Dat.U = u;            % input
Dat.Y = y;            % output
Dat.Ts = 10;          % sample time
Par.c = [3 3 3 3];    % number of clusters
Par.m = 2.2;          % fuzziness parameter
Par.ante = 1;         % product-space MFS
Dyn.Ny = [1 0 1 1;    % output lags
          0 1 1 1;
          0 0 1 0;
          0 0 0 1];
Dyn.Nu = [0 0;        % input lags
          0 0;
          1 0;
          0 1];
Dyn.Nd = [0 0;        % transport delays
          0 0;
          1 0;
          0 1];
```

Now the fuzzy model can be constructed and validated on a different data set (`ue`, `ye`):

```
FM = fmclust(Dat,Par,Dyn);
[ym,VAF] = fmsim(ue,ye,FM);
```

The validation data set was obtained by interchanging the two input signals. A comparison of the outputs of the fuzzy model with the process data is shown in Fig. 6.

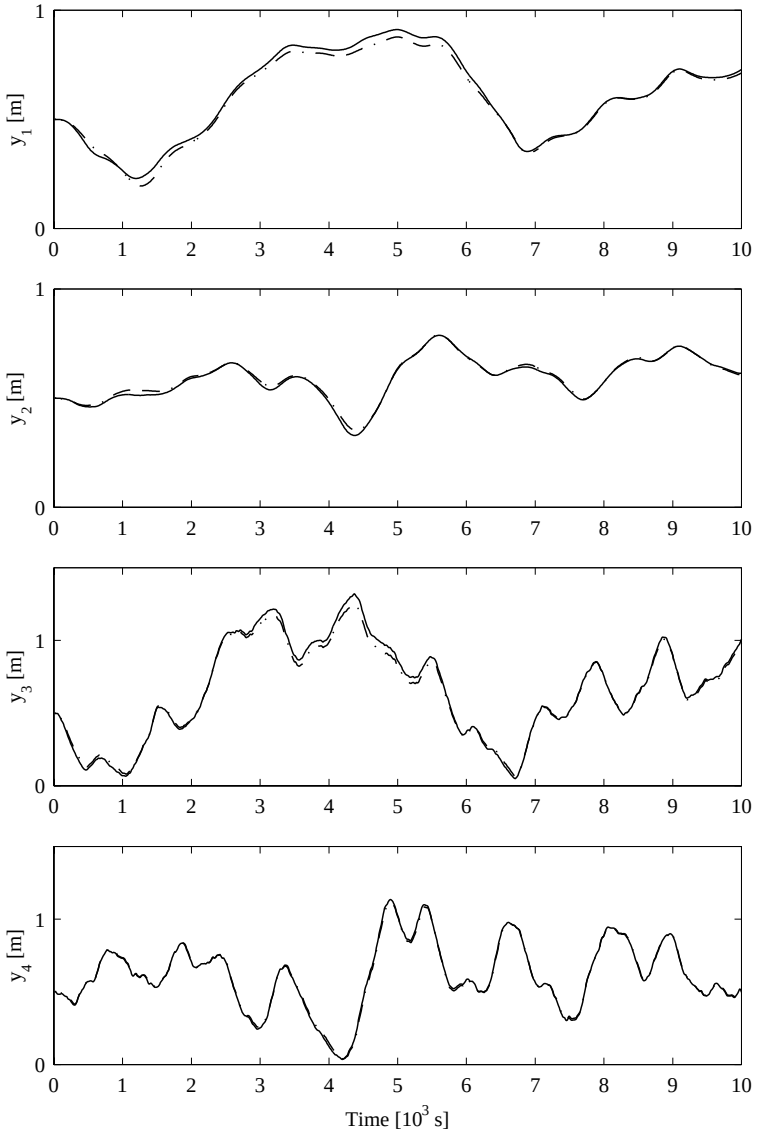

Figure 6. *Comparison of the process output (solid line) with the fuzzy model output (dashed-dotted line).*

For a comparison, a 4th-order linear state-space model was identified from the same data set by means of a subspace identification technique (Verhaegen and Dewilde, 1992). For the TS fuzzy model: VAF = $[99.41,\ 99.46,\ 99.47,\ 99.81]$, and for the linear state-space model: VAF = $[95.37,\ 85.11,\ 96.88,\ 89.64]$. It is clear that the fuzzy model is considerably more accurate than the linear one.

Reference

Construction and simulation of fuzzy models	
`fmclust`	build a MIMO NARX fuzzy model by product-space fuzzy clustering
`fmsim`	simulate a MIMO NARX fuzzy model

Utilities	
`fmstruct`	description of the fuzzy model structure `FM`
`fm2tex`	export a fuzzy model into a LaTEX file
`plotmfs`	plot membership functions of a fuzzy model
`rms`	root mean squared error
`vaf`	variance accounted for

Demonstrations	
`endemo1`	SISO model of throttle–speed relation
`endemo2`	MISO model of throttle–pressure relation
`endemo3`	SIMO model of throttle–(speed,pressure) relation
`fmidemo`	list of all toolbox demos
`lldemo1`	SISO model of a liquid level process
`ll4demo1`	MIMO model of a process with four cascaded tanks
`statdemo`	static SISO function (sine)
`wwdemo`	SIMO model of waste-water treatment process

fmclust

Purpose

Build a MIMO input–output static or dynamic fuzzy model from data by means of product-space fuzzy clustering (uses the Gustafson-Kessel algorithm).

Synopsis

```
[FM,Mu,Z] = fmclust(Dat,Par,Dyn)
```

Description

The `fmclust` function constructs a multivariable TS fuzzy model from input-output data. The data sequences and other data-related information is given in the `Dat` structure which has the following fields:

`Dat.U`	matrix containing input data
`Dat.Y`	matrix containing output data
`Dat.Ts`	sample time (optional, default 1)
`Dat.N`	number of data points per batch

The data sequences are stored in the *columns* of `Dat.U` and `Dat.Y`. In the sequel, the number of model inputs (columns of `Dat.U`) is denoted by `ni` and the number of model outputs (columns of `Dat.Y`) by `no`. `Dat.Ts` is the sample period of the data. It is only stored in the model structure `FM` and then used in simulation (`fmsim`) to get the right time scale. This parameter is optional (default `Ts` $= 1$). With `Dat.N`, one can specify that the input-output data consists of several separate batches concatenated in the `U` and `Y` matrices. `Dat.N` is a vector containing as many elements as there are batches. Each element then gives the number of data samples in the corresponding batch. For instance, `N=[100 250 200]` means that the `U` and `Y` matrices consist of three batches, the first one of 100 samples, the second one of 250 samples, and the third one of 200 samples.

User-supplied parameters related to clustering and model extraction are are given in the `Par` structure which has the following fields:

`Par.c`	number of clusters (thus also rules) per output
`Par.m`	the fuzziness exponent per output (default 2)
`Par.tol`	termination tolerance (default 0.01)

`Par.seed`	seed for random generator (default sum(100*clock))
`Par.ante`	type of the antecedent (default 1)

The number of required clusters is a scalar for MISO systems and a vector for MIMO systems (each MISO model may have a different number of clusters). All the remaining fields of `Par` are optional. `Par.m` is the fuzziness exponent (`Par.m` > 1) with the default value `Par.m` $= 2$. Larger values imply fuzzier (more overlapping) clusters. For MISO systems, it is a scalar, for MIMO systems a vector, i.e., each MISO model may have a different degree of fuzziness in clustering. The termination tolerance for the clustering algorithm can be given in `Par.tol` (default `Par.tol` $= 0.01$). In fuzzy clustering, a random initial partition is usually generated. In order to obtain reproducible results, the random generator may be seeded by supplying the `Par.seed` parameter. Its default value is `sum(100*clock))`. `Par.ante` specifies the type of the antecedent in the fuzzy model. Currently, two options are implemented, 1 for product-space membership functions (default), and 2 for projected membership functions. Product-space membership functions give faster but often less accurate models. For MISO systems, `Par.FMtype` is a scalar, for MIMO systems it is a $1 \times$ `no` vector (each MISO model can be of a different type).

The `Dyn` structure defines the dynamics of the input–output model. This structure is optional (if not supplied, a static MIMO model $y = f(u)$ is constructed) and has the following fields:

`Dyn.Ny`	number of delays in y (default 0)
`Dyn.Nu`	number of delays in u (default 1)
`Dyn.Nd`	number of transport delays (default 0)

`Dyn.Ny` is the number of delays in `y` (analogical to the order of the denominator polynomial of a linear transfer function). The default value is `Dyn.Ny=zeros(no,no)`, i.e., a static system. For MISO systems, `Dyn.Ny` is a scalar (there is one output only), for MIMO systems it is an `no` $\times$ `no` matrix. Each row corresponds to one MISO model and specifies which delays of which outputs are included in that model. `Dyn.Nu` defines the delays in `u` (analogical to the order of the numerator polynomial of a linear transfer function). The default value is `Dyn.Nu=ones(no,ni)` (static system). For MISO systems, `Dyn.Nu` is a $1 \times$ `ni` vector, for MIMO systems it is an `no` $\times$ `ni` matrix. Each row corresponds to one MISO model and specifies which delays of which inputs are included in that model. `Dyn.Nd`

defines the number of pure transport delays in `u`. The default value is `Dyn.Nd=zeros(no,ni)` (static system, thus no delay). For MISO systems, `Dyn.Nd` is a $1 \times$ `ni` vector, for MIMO systems it is an `no` $\times$ `ni` matrix. Each row corresponds to one MISO model and specifies which the transport delays in all the inputs of that model. To obtain a causal model $y(k+1) = f(y(k), ..., u(k), ...)$, `Dyn.Nd` must be set to one.

The output of `fmclust` is the `FM` structure which contains all the parameters of the obtained fuzzy model. See `fmstruct` for details. The fuzzy partition matrices are returned in the `Mu` cell array, where each cell corresponds to one output. Similarly, `Z` is a cell array containing the data matrix that has been clustered. To visualize the partition, the cells of `Mu` can be plotted against the columns of the cells in `Z`.

Algorithm

`fmclust` uses fuzzy clustering in the product space of the regressors and the regressand in order to approximate a nonlinear system by a collection of local linear models. Each local model then corresponds to one fuzzy rule of the Takagi-Sugeno type. MIMO systems are identified (and simulated) as a set of coupled MISO systems. See (Babuška, 1998) for details.

Example

Approximate a sinusoidal function by a TS fuzzy model with five rules:

```
Dat.U = (0:0.02:1)';
Dat.Y = sin(7*u);
Par.c = 5;
[FM,Mu] = fmclust(Dat,Par);
[ym,VAF] = fmsim(u,y,FM); VAF
```

See Also

`fmsim, fmstruct, fm2tex`

fmsim

Purpose

Simulate a MIMO input–output fuzzy model.

Synopsis

```
[Ym,q,DOF,Yl,Ylm] = FMSIM(U,Y,FM, ...
                          Ymin,Ymax,show,H)
```

Description

The `fmsim` function simulates a fuzzy model `FM` from the input data `U` and compares the simulated output `Ym` with the true output `Y`. The first several values of `Y` are used to initialize `Ym`. The number of these values depends on the number of lags defined in `Dyn`. The `Y` parameter is optional, if an empty matrix is supplied, zero initial conditions are used. The format of data in `U` and `Y` is the same as in `fmclust`. `Ymin` and `Ymax` are the lower and upper bounds on `Y`. During the simulation, the outputs are constrained between these bounds. This parameter is optional, default bounds are `Ymin=-inf` and `Ymax=inf`. The `show` parameter determines what graphical output is shown on the screen. Set this parameter to 1 for on-line plot during the simulation, to 2 for a plot at the end of simulation, and to 0 for no plot at all (optional, default 1). The `H` parameter (optional) specifies the prediction horizon. In this version, it only can be set to 1, which means one-step-ahead prediction. If not supplied, simulation from input is used.

The output argument `q` is a performance index of the model, computed as *variance accounted for* (VAF). See `vaf` for details. `DOF` is a matrix containing the degrees of fulfillment of the rules. For multiple-output systems, the individual models are concatenated in one matrix: `[DOF_1, DOF_2, ..., DOF_no]`. The contributions of the consequents of the individual rules are returned in the matrix `Yl`. `Ylm` is identical to `Yl` except for that all outputs but the one corresponding to the largest `DOF` are masked by NaN. This format is suitable for plotting the local models. The same holds for MIMO systems as with `DOF`.

Note: `fmsim` currently only works properly for static dynamic systems with inputs, some minor adjustments are needed for autoregressive systems. The possibility to include a prediction horizon and a "single-step" mode will be added as well.

See Also

`fmstruct, fmclust, vaf`

fmstruct

Purpose

Help on the structure of FM.

Synopsis

`fmstruct` or `help fmstruct`

Description

The parameters of a fuzzy model are stored in a MATLAB 5 structure named FM (fuzzy model) which has the following fields:

`Ts`	sample time
`ni`	number of inputs
`no`	number of outputs
`N`	number of data samples used for identification
`tol`	termination tolerance for clustering
`seed`	seed for random initialization of fuzzy partition
`date`	date of model construction
`ny`	number of output lags
`nu`	number of input lags
`nd`	number of pure delays
`ante`	type of fuzzy model
`m`	fuzziness exponent
`Alist`	list of indices of used antecedent variables
`Clist`	list of indices of used consequent variables
`rls`	rule matrix
`mfs`	membership function matrix
`th`	consequent parameters
`s`	standard deviation for `th`
`V`	cluster centers
`P`	norm-inducing matrices
`zmin`	minima of each column of the data matrix `Z`
`zmax`	maxima of each column of the data matrix `Z`
`InputName`	names of input variables (cell array)
`OutputName`	names of output variables (cell array)

Each element of the FM array corresponds to one output of the model.

See Also

`fmclust, fmsim, fm2tex`

fm2tex

Purpose

Export a fuzzy model into a LaTEX file.

Synopsis

`fm2tex(FM,filename)`

Description

This utility function writes some of the information contained in FM into a LaTEX file. The created file contains an introductory description of the model and its structure. For each output, the rule base, the consequent parameters and the cluster centers are included. FM is the fuzzy model parameter matrix and `filename` specifies the name of the LaTEX file to be created. If a file with the specified name already exists, it is overwritten without a warning.

See Also

`fmstruct, plotmfs`

plotmfs

Purpose

Plot membership functions.

Synopsis

```
plotmfs(FM,opt)
```

Description

This utility plots the membership functions contained in `FM` on the screen. Projected membership functions are plotted directly. Product-space membership functions, however, cannot be visualized in general. An approximate idea about their shape is obtained by plotting their projections.

See Also

`fmstruct, fm2tex`

rms

Purpose

Root-mean-squared error between two signals.

Synopsis

```
rms(y1,y2)
```

Description

Function `rms` computes the root-mean-squared error between two signals. The RMS index is often used to assess the quality of a model, by comparing the true output with the output of the model.

See Also

`vaf`

vaf

Purpose

Percentile variance accounted for (VAF) between two signals.

Synopsis

```
vaf(y1,y2)
```

Description

Function `vaf` computes the percentile *variance accounted for* (VAF) between two signals as follows:

$$\text{VAF} = 100\% \cdot \left[1 - \frac{\text{var}(\texttt{y1} - \texttt{y2})}{\text{var}(\texttt{y1})}\right]$$

The VAF of two equal signals is 100%. If the signals differ, VAF is lower. When `y1` and `y2` are matrices, VAF is calculated for each column. The VAF index is often used to assess the quality of a model, by comparing the true output with the output of the model.

See Also

`rms`

Literature

Babuška, R. (1998). *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Boston.

Takagi, T. and M. Sugeno (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. Systems, Man and Cybernetics 15*(1), 116–132.

Verhaegen, M. and P. Dewilde (1992). Subspace model identification. Part I: the output-error state space model identification class of algorithms. *International Journal of Control 56*, 1187–1210.

Notes