ANFIS
(Adaptive Neuro-Fuzzy Inference System)

By

Heikki Koivo[1]

[1] Heikki Koivo © 2000

A fuzzy system (FIS in MATLAB) can be considered to be a parameterized nonlinear map, called *f*. This point will be made clearer later on in the unified view of soft computing, but let's write here explicitly the expression of *f*.

$$f(\mathbf{x}) = \frac{\sum_{l=1}^{m} y^l \left(\prod_{i=1}^{n} m_{A_i^l}(x_i)\right)}{\sum_{l=1}^{m} \left(\prod_{i=1}^{n} m_{A_i^l}(x_i)\right)}$$

where y^l is a place of output singleton if Mamdani reasoning is applied or a constant if Sugeno reasoning is applied. The membership function $m_{A_i^l}(x_i)$ corresponds to the input $\mathbf{x} = [x_1, \cdots, x_n]$ of the rule l.
The *and* connective in the premise is carried out by a product and defuzzification by the center-of-gravity method.

This can be further written as

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i b_i(\mathbf{x})$$

where $w_i = y^i$ and

$$b_j(\mathbf{x}) = \frac{\prod_{i=1}^{n} m_{A_i^l}(x_i)}{\sum_{l=1}^{m} \left(\prod_{i=1}^{n} m_{A_i^l}(x_i) \right)}$$

If *F* is a continuous, nonlinear map on a compact set, then *f* can approximate *F* to any desired accuracy, i.e.

$$F \approx f_{FS} \qquad .$$

The reader can skip this part without loss of continuity, if the mathematical machinery of Hilbert spaces is not familiar.

Well-known theorems from Approximation theory for polynomials, can be extended to fuzzy systems (e.g. Wang: A course in fuzzy systems and control, Prentice Hall, 1997).

The following theorems are found from R.F. Curtain and A.J. Pritchard: Functional Analysis in Modern Applied Mathematics as Corollaries of Orthogonal Projection Theorem.

Theorem 1.
Let F be a bounded function on $[a,b]$ and $E=\{x_1,\cdots,x_k\}$ a set of points in $[a,b]$. Then there exists the least squares polynomial of degree $\leq n$, p_n^k which minimizes

$$\sum_{i=1}^{k}\left|F(x_i)-p(x_i)\right|^2$$

over all polynomials of degree $\leq n$.

Theorem 2.
If $F\in C[a,b]$, then for any $n\geq 0$, there exists a best approximating polynomial p_n of degree $\leq n$ such that

$$\left\|F-u_n\right\|_\infty \leq \left\|F-p\right\|_\infty$$

over all polynomials p of degree $\leq n$.

We can also consider the simpler problem of approximating at finitely many points.

Theorem 3.
If F is a bounded function on $[a,b]$ and $E=\{x_1,\cdots,x_k\}$ a set of points in $[a,b]$, then there exists a best approximating polynomial p_n^k of degree $\leq n$, p_n^k which minimizes

$$\max_{0\leq i\leq k}\left|F(x_i)-p(x_i)\right|$$

over all polynomials of degree $\leq n$.

The message of the Theorems can also be summarized by saying that polynomials are dense in the space of continuous functions.
The same can also be said of trigonometric functions.

ANFIS STRUCTURE

Consider a Sugeno type of fuzzy system having the rule base

1. If x is A_1 and y is B_1, then $f_1=p_1x+q_1y+r_1$
2. If x is A_2 and y is B_2, then $f_2=p_2x+q_2y+r_2$

Let the membership functions of fuzzy sets A_i, B_i, i=1,2, be m_{A_i}, m_{B_i}.

In evaluating the rules, choose *product* for T-norm (logical *and*).

1. Evaluating the rule premises results in

$$w_i = m_{A_i}(x)m_{B_i}(y),\ i=1,2.$$

2. Evaluating the implication and the rule consequences gives

$$f(x,y)=\frac{w_1(x,y)f_1(x,y)+w_2(x,y)f_2(x,y)}{w_1(x,y)+w_2(x,y)}.$$

Or leaving the arguments out

$$f=\frac{w_1f_1+w_2f_2}{w_1+w_2}.$$

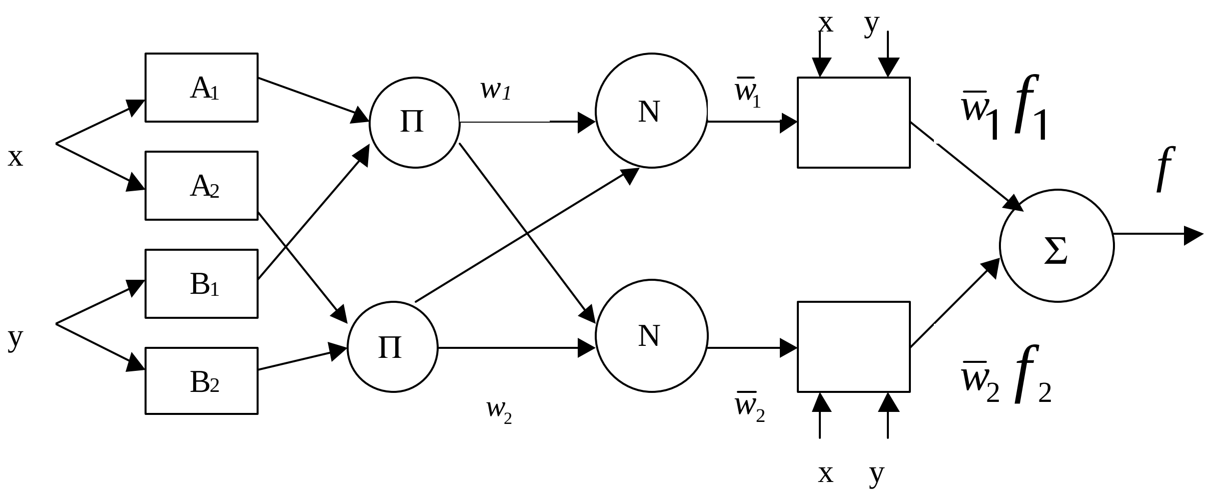
This can be separated to phases by first defining

$$\overline{w}_i=\frac{w_i}{w_1+w_2}.$$

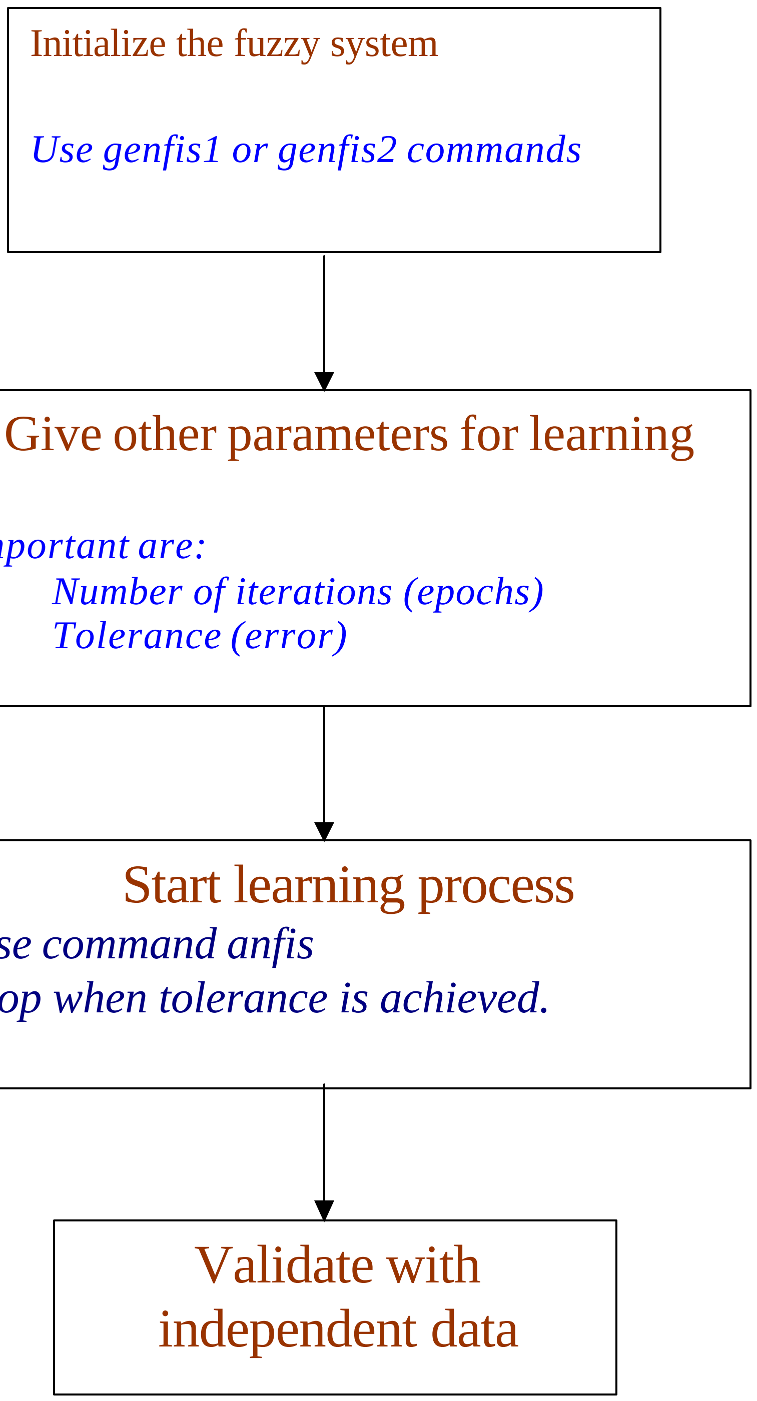
Then f can be written as

$$f=\overline{w}_1f_1+\overline{w}_2f_2.$$

All computations can be presented in a diagram form.



Basic flow diagram of computations in ANFIS.


Initialize the fuzzy system
Use genfis1 or genfis2 commands
Give other parameters for learning
Important are:
Number of iterations (epochs)
Tolerance (error)
Start learning process
Use command anfis
Stop when tolerance is achieved.
Validate with independent data

In Fuzzy Control Toolbox a useful command called ***anfis*** exists. This provides an optimization scheme to find the parameters in the fuzzy system that best fit the data. It is explained in the Toolbox manual that since most (not all) optimization algorithms require computation of the gradient, this is done with a neural network. Then, in principle, any of the optimization schemes, say those in the MATLAB Optimization Toolbox, can be used.

GENFIS1 and ANFIS Commands

It is not clear at the beginning, what the initial fuzzy system should be, that is, the type and number of membership functions, command ***genfis1*** may be used. This command will go over the data in a crude way and find a good starting system.

GENFIS1 *Generate FIS matrix using generic method.*

GENFIS1(DATA, MF_N, IN_MF_TYPE) *generates a fismatrix from training data DATA, using grid partition style. MF_N is a vector specifying the number of membership functions on all inputs. IN_MF_TYPE is a string array where each row specifies the MF type of an input variable.*

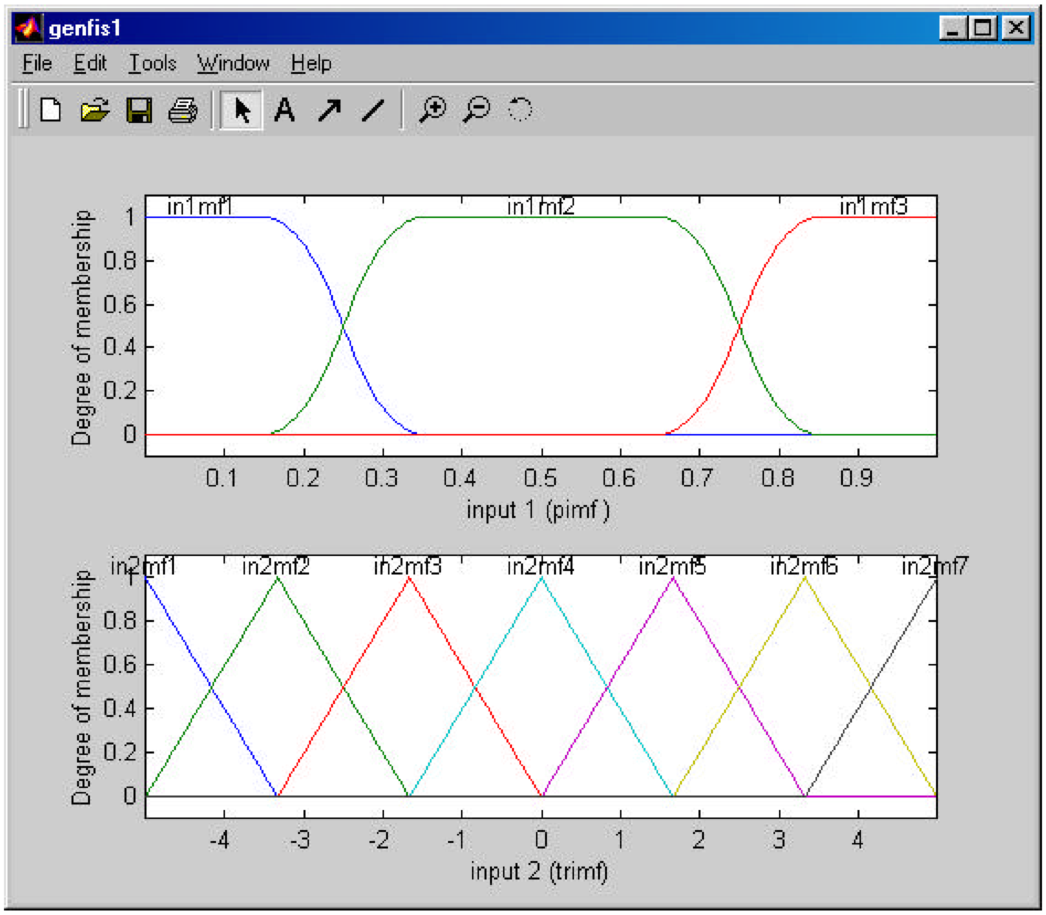
If MF_N is a number and/or IN_MF_TYPE is a single string, they will be used for all inputs. Note that MF_N and IN_MF_TYPE are pass directly to GENPARAM for generating input MF parameters.

Default number of membership function MF_N is 2; default input membership function type is 'gbellmf'.

For example:

```
 % Generate random data
        NumData = 1000;
        data = [rand(NumData,1) 10*rand(NumData,1)-5 rand(NumData,1)];

% Specify number and type of membership functions
        NumMf = [3 7];
        MfType = str2mat('pimf', 'trimf');

% Use GENFIS1 to generate initial membership functions
        FisMatrix = genfis1(data, NumMf, MfType);
```

```
% Plot the membership functions
        figure('name', 'genfis1', 'numbertitle', 'off');
        NumInput = size(data, 2) - 1;
        for i = 1:NumInput;
                subplot(NumInput, 1, i);
                plotmf(FisMatrix, 'input', i);
                xlabel(['input ' num2str(i) ' (' MfType(i, :) ')']);
        end

        See also GENPARAM, ANFIS.
```



SUMMARY: Use **genfis1** to generate **initial FIS system.** Use **anfis** to generate the best FIS system. **Anfis** uses the result from **genfis1** to start optimization.

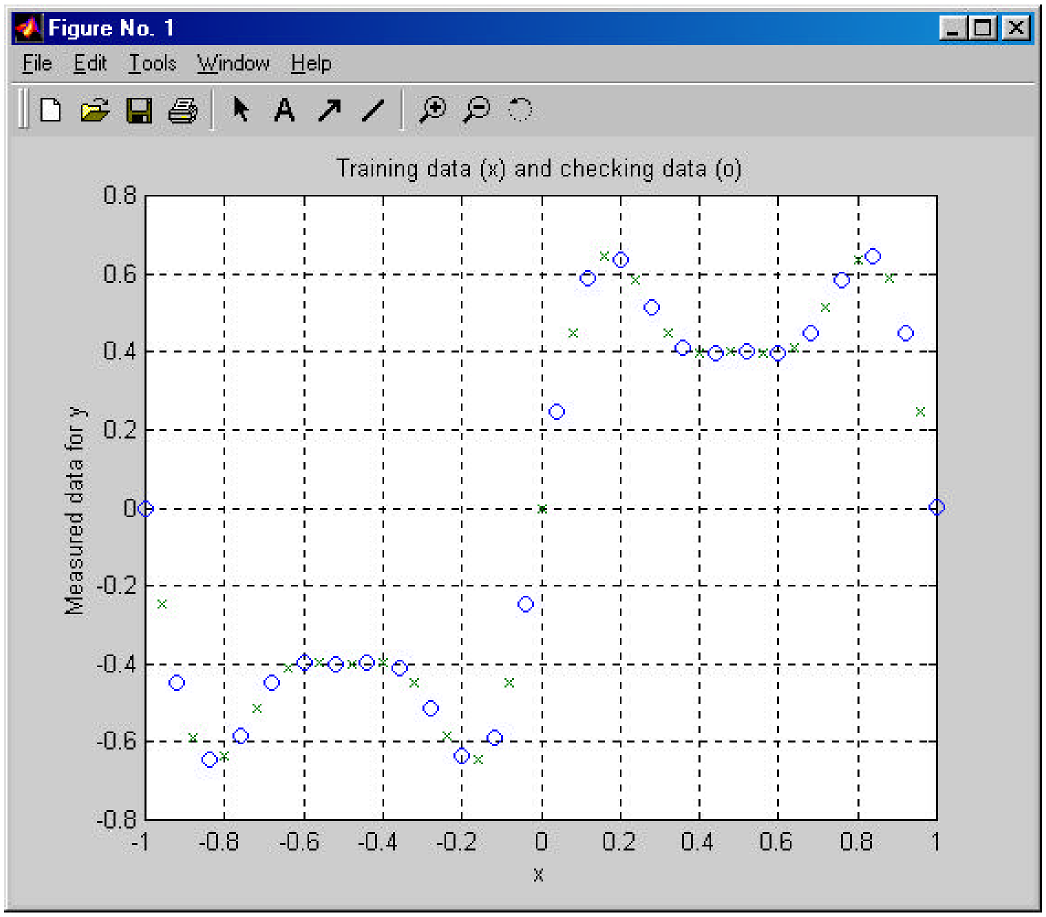
EXAMPLE 1.

Consider fitting a fuzzy system on a nonlinear function on [-1,1] consisting of sum of three sinusoidal functions

$$y = 0.6\sin(px) + 0.3\sin(3px) + 0.1\sin(5px).$$

SOLUTION:
The problem is solved using MATLAB and Fuzzy Logic Toolbox.
Since we are not provided data, we first generate an input-output data set

```
% Number of input data points numPts = 51 and input data x
numPts=51;x=linspace(-1,1,numPts)';

% Output data y
y=0.6*sin(pi*x)+0.3*sin(3*pi*x)+0.1*sin(5*pi*x);

% Store data in matrix called data; use part for training part for checking
data=[x y]; % total data set
trndata = data(1:2:numPts,:); % training data set
chkdata = data(2:2:numPts,:); % checking data set

% Plot training data (o) and checking data (x)
plot (trndata(:,1),trndata(:,2), 'o',chkdata(:,1),chkdata(:,2),'x')
grid
title('Training data (x) and checking data (o)')
xlabel('x');ylabel('Measured data for y');
```

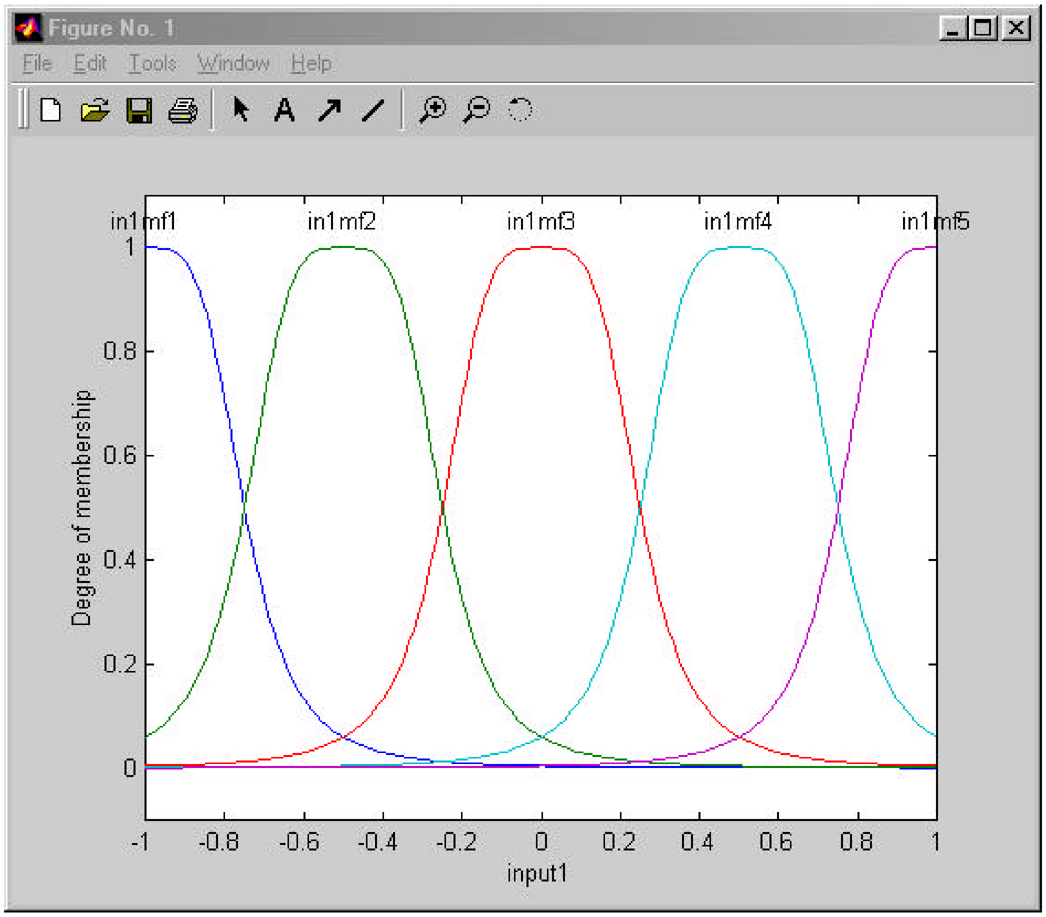


% Next apply genfis1 to generate initial set of membership functions.

> *nummfs=5;* ***% number of membership functions***
> *mftype='gbellmf';* ***% membership functions type is generalized bell***
> *fismat = genfis1(trndata,nummfs,mftype);*

You can see the plot of the initial membership functions by opening *anfis* under FIS editor and then looking at the input membership functions or plotting them with command

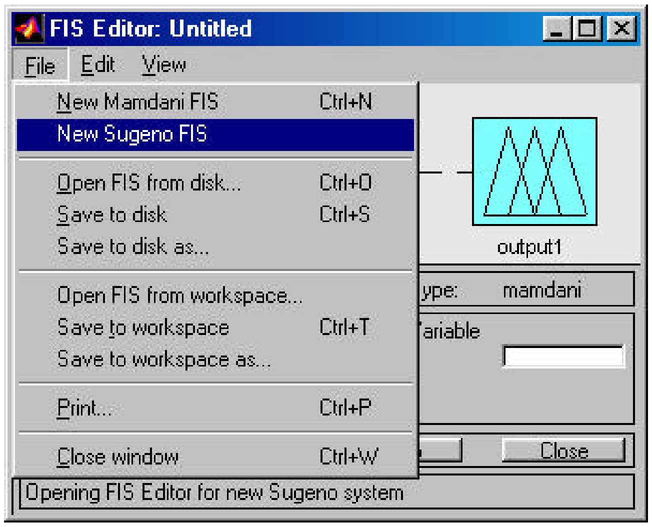
> *plotmf(fismat,'input',1);*



Another way to proceed is to use Anfis GUI. First open the GUI for Fuzzy Logic Toolbox by typing command

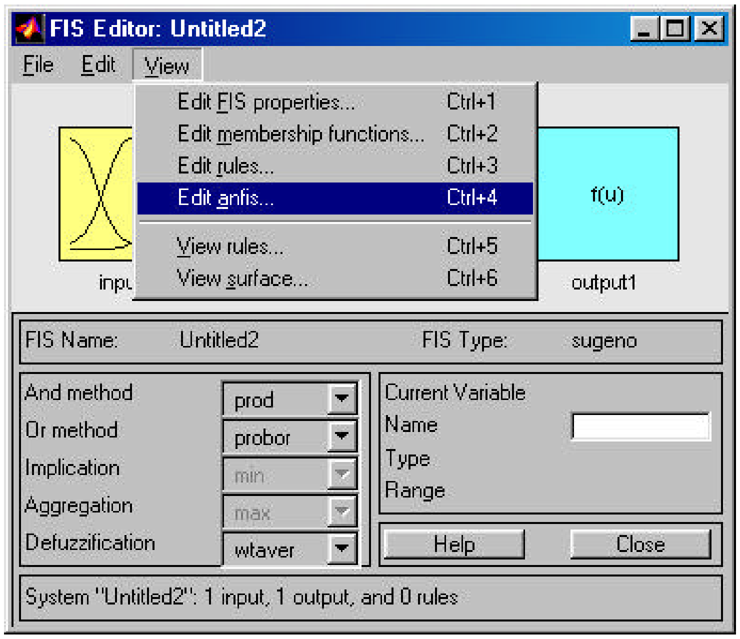
fuzzy

Then in the FIS Editor choose New Sugano system from File menu.



Opening FIS Editor for new Sugeno system

Then in View menu choose Edit Anfis



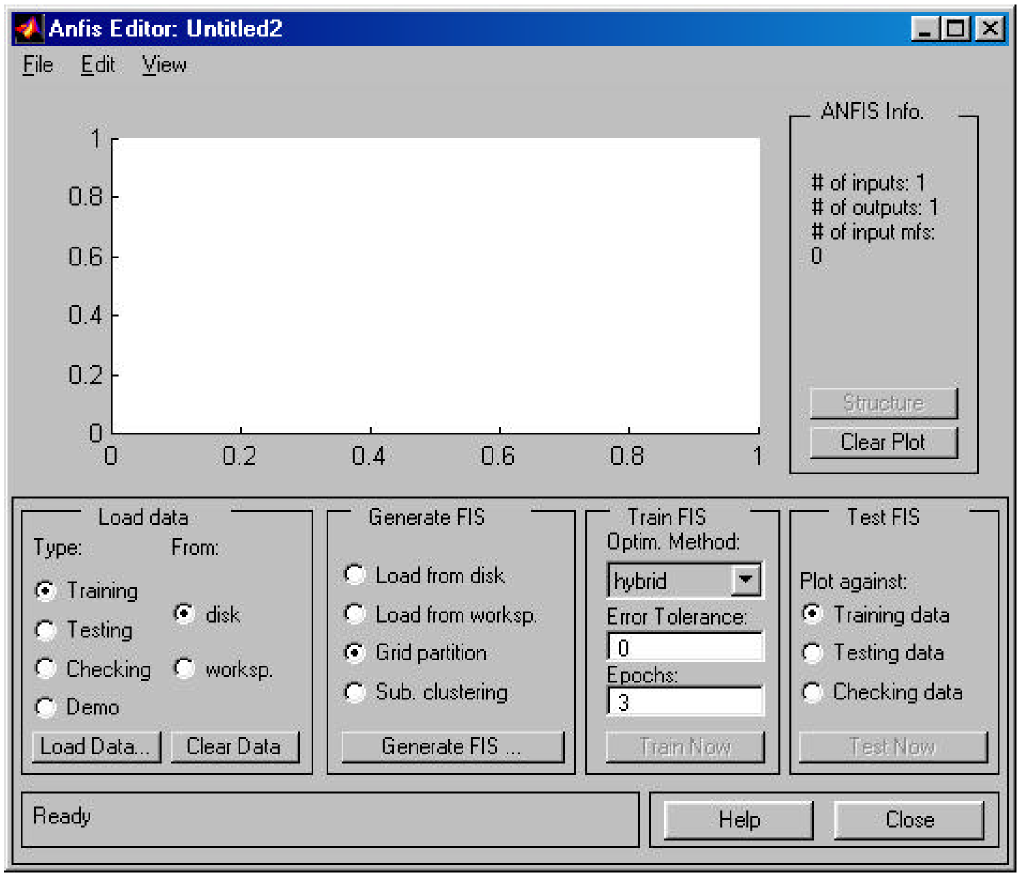
FIS Name: Untitled2 FIS Type: sugeno

And method: prod
Or method: probor
Implication: min
Aggregation: max
Defuzzification: wtaver

Current Variable
Name
Type
Range

System "Untitled2": 1 input, 1 output, and 0 rules

This will result in the following display.



ANFIS Info.

# of inputs: 1
# of outputs: 1
# of input mfs:
0

Load data

Type: [✓ Training] Testing Checking Demo

From: [✓ disk] worksp.

Generate FIS

Load from disk Load from worksp. [✓ Grid partition] Sub. clustering

Train FIS

Optim. Method: hybrid

Error Tolerance: 0

Epochs: 3

Test FIS

Plot against: [✓ Training data] Testing data Checking data

Ready

Anfis Editor display is divided into four main subdisplays:

1. Load data
2. General FIS
3. Train FIS
4. Test FIS

Let us consider them in more detail.

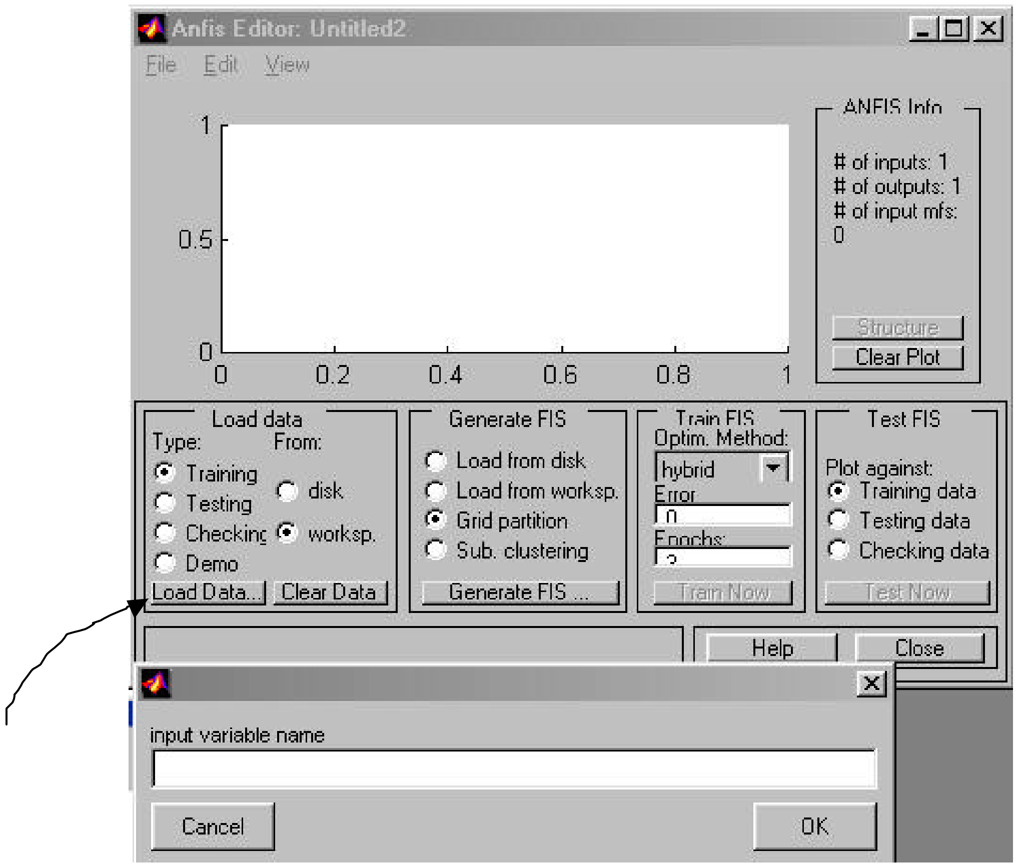
1. Load data
   Here you can choose Type of data according to the purpose: Training (default choice), Testing, Checking, or Demo.

   For your first effort stick with Training data.

   Suppose you have generated data on MATLAB command side. It should be in a matrix form, where the first columns consist of input data and the last of output data.

   Here you also have to be care of where the data are located. Data can be either stored on disk or it is in workspace only, because you just generated it. In our example it is in workspace so you have to change the default

value disk to workspace. Click Load data and you are asked to type Input variable name to the space provided.
Remember that the data are in matrix form.



ANFIS Info: # of inputs: 1; # of outputs: 1; # of input mfs: 0

Load data — Type: [✓ Training] Testing Checking Demo; From: disk [✓ worksp.]

Generate FIS: Load from disk; Load from worksp.; [✓ Grid partition]; Sub. clustering

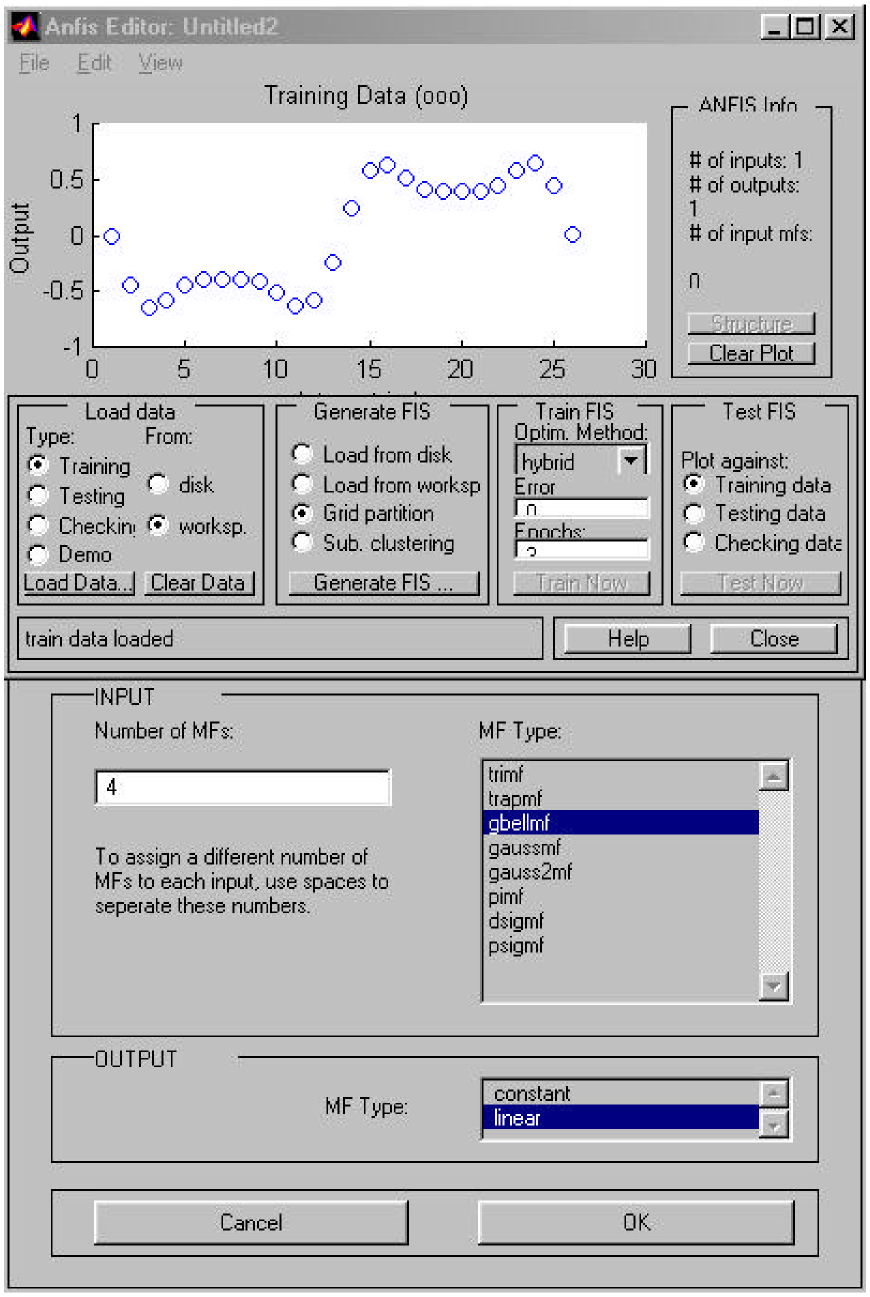
Train FIS — Optim. Method: hybrid; Error: 0; Epochs: 3

Test FIS — Plot against: [✓ Training data] Testing data Checking data

input variable name

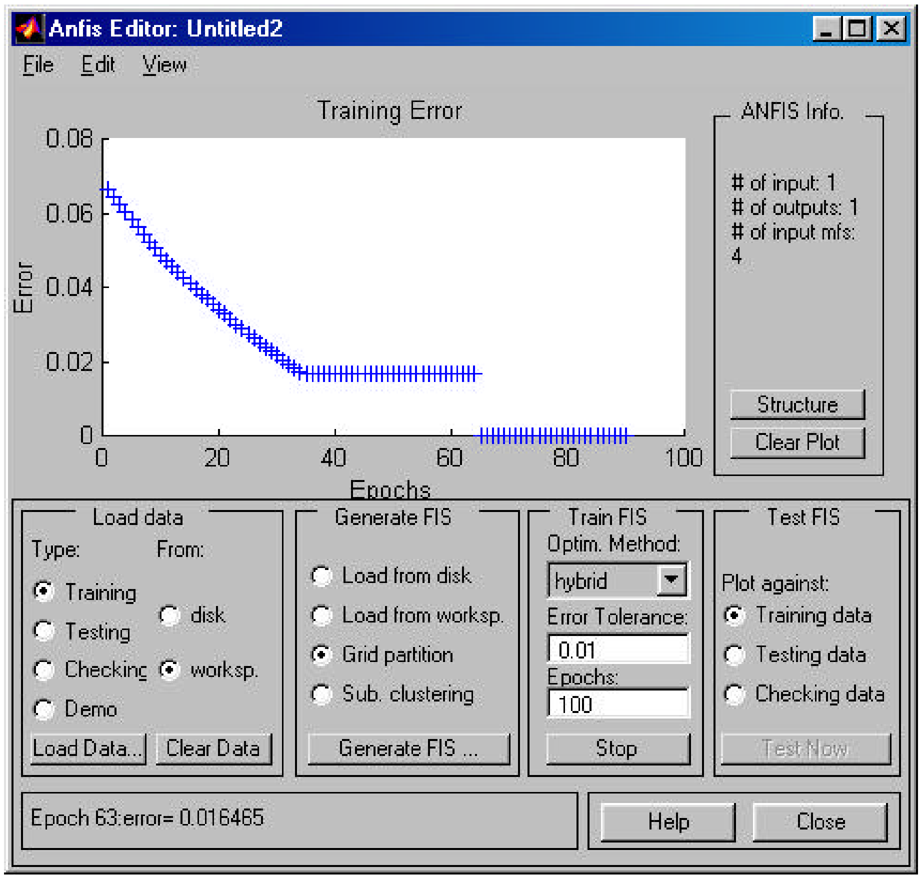
2. General FIS

   Once data have been loaded, an initial fuzzy system can be generated using Generate FIS. Easiest option is to leave the default choice, Grid partition on. Click Generate FIS and the following window opens.



As default 4 gbellmf-memebrship functions are chosen.

3. Train FIS
   The main computing occurs in the Training block. Here hybrid algorithm is used. The Error tolerance is 0. In practice, a complete fit is not achieved, so change this to, say 0.01. During training you can see how Training error develops. Epochs means number of iterations. The default value 3 is hardly ever enough. It is not unusual to have hundreds or even thousands of iterations. Pick 100. Now you are ready for training. Click Train now.



# of input: 1
# of outputs: 1
# of input mfs:
4

Type: [✓ Training] Testing Checking Demo
From: disk [✓ worksp.]

Load from disk / Load from worksp. / [✓ Grid partition] / Sub. clustering

Optim. Method: hybrid
Error Tolerance: 0.01
Epochs: 100

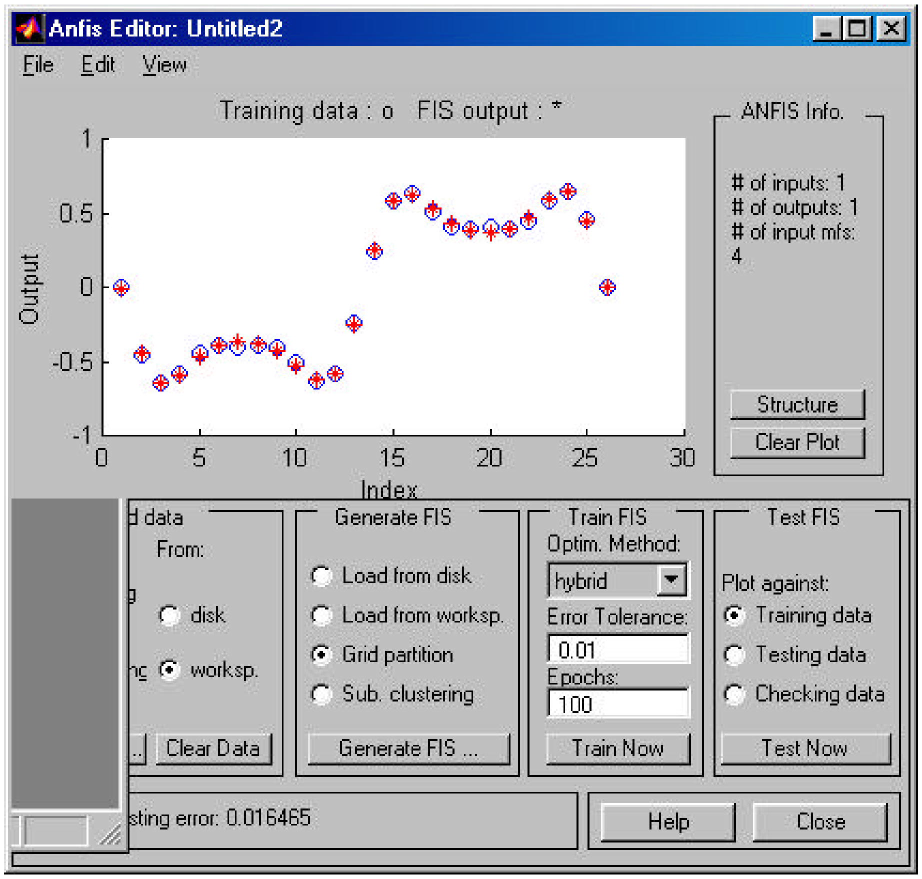
Plot against: [✓ Training data] Testing data Checking data

Epoch 63:error= 0.016465

You can improve the situation by introducing more membership functions.

4. Test FIS
   Once the result is satisfactory, you can test the performance of FIS against either training, testing or checking data. Default value is training data.

   Click Test now and you will see the following display.



MATLAB Command side

Now start the actual optimization procedure by calling **anfis**.

% Determine number of iterations
numepochs = 40;

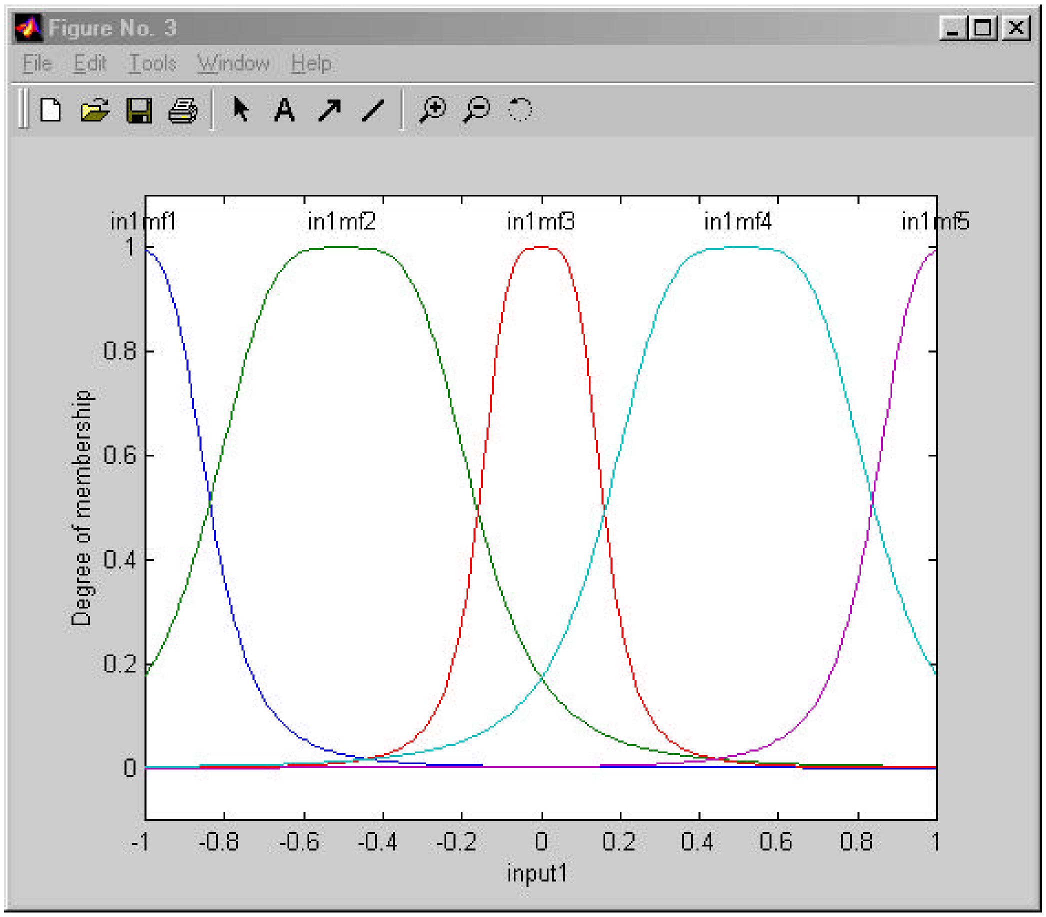
% Start the optimization
```
 [fismat1, trnerr, ss, fismat2, chkerr]= ...
        anfis(trndata, fismat, numepochs, NaN, chkdata);
```

```
ANFIS info:
        Number of nodes: 24
        Number of linear parameters: 10
        Number of nonlinear parameters: 15
        Total number of parameters: 25
```

```
            Number of training data pairs: 26
            Number of checking data pairs: 25
            Number of fuzzy rules: 5


Start training ANFIS ...

   1      0.0601596    0.058193
   2      0.0571745    0.0553133
   3      0.0541812    0.0524169
   4      0.0511445    0.0494663
   5      0.0480347    0.0464298
……

Step size decreases to 0.010567 after epoch 39.
  40      0.0110313    0.0117976

Designated epoch number reached --> ANFIS training completed at epoch 40.

Minimal training RMSE = 0.010380
Minimal checking RMSE = 0.0112159
```

Next check the final membership functions again by *plotmf* or
using FIS Editor

```
 plotmf(fismat1,'input',1)
```



Draw the fitted curve and checking data in the same figure:

out=evalfis(chkdata(:,1),fismat1);% Evaluates the output of the fuzzy system fismat1 - input checking
hold;plot(chkdata(:,1),out); % Plots out vs x
plot(chkdata(:,1),chkdata(:,2));% Plots y vs x (training data)
plot(x,y); grid

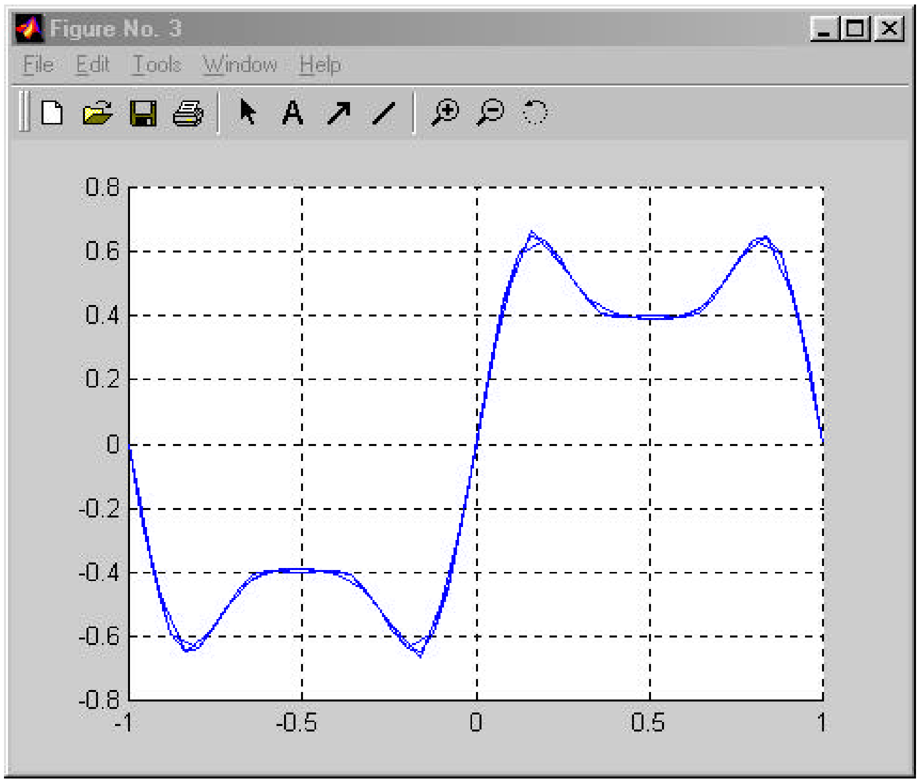


Figure No. 3
File Edit Tools Window Help
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.5
0
0.5
1

EXERCISE 1. Choose different value for number of membership functions and different type of membership functions. Study also other case when you add noise to your measurement data set.

EXERCISE 2. Repeat Example 1 and Exercise 1 for *humps(x)* function.

REMARK: **Anfis** command has limitations. It can only be used for Sugeno systems and further:

- Constant and linear output membership functions only
- Single output derived by weighted average defuzzification

  In MATLAB *help anfis* the following is said

  When a specific Sugeno fuzzy inference system is used for fitting, we can call ANFIS with from 2 to 4 input arguments and it returns from 1 to 3 output arguments
- Unity weight for each rule

APPENDIX

To generate the initial FIS matrix use *genfis1*.

Type *help genfis1* in MATLAB to see the following.

GENFIS1 *Generate FIS matrix using generic method.*

> ***GENFIS1(DATA, MF_N, IN_MF_TYPE)*** *generates a fismatrix from training data DATA, using grid partition style. MF_N is a vector specifying the number of membership functions on all inputs. IN_MF_TYPE is a string array where each row specifies the MF type of an input variable.*
>
> *If MF_N is a number and/or IN_MF_TYPE is a single string, they will be used for all inputs. Note that MF_N and IN_MF_TYPE are pass directly to GENPARAM for generating input MF parameters.*
>
> *Default number of membership function MF_N is 2; default input membership function type is 'gbellmf'.*
>
> *For example:*

```
 % Generate random data
          NumData = 1000;
          data = [rand(NumData,1) 10*rand(NumData,1)-5 rand(NumData,1)];

% Specify number and type of membership functions
          NumMf = [3 7];
          MfType = str2mat('pimf', 'trimf');

% Use GENFIS1 to generate initial membership functions
          FisMatrix = genfis1(data, NumMf, MfType);

% Plot the membership functions
          figure('name', 'genfis1', 'numbertitle', 'off');
          NumInput = size(data, 2) - 1;
          for i = 1:NumInput;
                    subplot(NumInput, 1, i);
                    plotmf(FisMatrix, 'input', i);
                    xlabel(['input ' num2str(i) ' (' MfType(i, :) ')']);
          end
```

> *See also GENPARAM, ANFIS.*

To determine the best FIS system use *anfis*.

ANFIS

Training routine for Sugeno-type FIS (MEX only).
ANFIS applies the least-squares method and the back-propagation gradient descent for identifying linear and nonlinear parameters, respectively, in a Sugeno-type fuzzy inference systems.

FISMAT = ANFIS(TRN_DATA) attempts to fit a training data TRN_DATA using a Sugeno FIS with 2^N rules, where N is the number of inputs.
(This is recommanded only when N is less than 7.) The final fuzzy inference system is returned as a FIS matrix FISMAT. The format of TRN_DATA is described below.

When a specific Sugeno fuzzy inference system is used for fitting, we can call ANFIS with from 2 to 4 input arguments and it returns from 1 to 3 output arguments:

[FISMAT, ERROR, STEPSIZE] = ANFIS(TRN_DATA, IN_FISMAT, T_OPT, D_OPT)

TRN_DATA: training data matrix, where each row is a desired input/output data pairs, with output at the last column.

IN_FISMAT: input FIS matrix to specify the structure and initial parameters for fitting.

This can be generated from data directly using the command GENFIS.
T_OPT: training options.
T_OPT(1): training epoch number (default: 10)
T_OPT(2): training error goal (default: 0)
T_OPT(3): initial step size (default: 0.01)
T_OPT(4): step size decrease rate (default: 0.9)
T_OPT(5): step size increase rate (default: 1.1)

If any element of T_OPT is NaN (not a number), then the default value is used. Default values can be changed directly by modifying this file. If T_OPT itself is missing, a null matrix, or an NaN, then it will also take default value.
The training process stops when the designated epoch number is reached or when the training goal is achieved.
The step size is decreased (by multiplying the decrease rate) if the error measure undergoes two consecutive combinations of increase and decrease; increased (by multiplying the increase rate) if the error measure undergoes four consecutive decreases.
D_OPT: display options.
D_OPT(1): display ANFIS information (default: 1)
D_OPT(2): display error measure (default: 1)
D_OPT(3): display step size (default: 1)
D_OPT(4): display final results (default: 1)
The parsing of D_OPT is the same as T_OPT.

FISMAT: output FIS matrix, which is the FIS matrix corresponding to the minimum training error.
ERROR: array of root mean squared errors.
STEPSIZE: array of step sizes.

If checking data is involved in the training process, then ANFIS should be called with 5 input arguments and it returns from 1 to 5 output arguments:

[FISMAT1, T_ERROR, STEPSIZE, FISMAT2, C_ERROR] = ...
ANFIS(TRN_DATA, IN_FISMAT, T_OPT, D_OPT, CHK_DATA)

TRN_DATA: training data matrix, see above.
IN_FISMAT: input FIS matrix, see above.
T_OPT: training options, see above.
D_OPT: display options, see above.
CHK_DATA: checking data matrix, with the same format as TRN_DATA.
FISMAT1: output FIS matrix corresponding to the min. training error.
T_ERROR: array of root mean squared training errors.
STEPSIZE: array of step sizes.
FISMAT1: output FIS matrix corresponding to the min. checking error.
C_ERROR: array of root mean squared checking errors.

For example:

```
x = (0:0.2:10)';
y = sin(2*x)./exp(x/5) + randn(size(x))/30;
TrainData = [x y];
NumMfs = 5;
MfType = 'gbellmf';
NumEpochs = 20;
StepSize = 0.1;
InputFismat = genfis1(TrainData, NumMfs, MfType);
OutputFismat = anfis(TrainData, InputFismat, [NumEpochs nan StepSize]);
yy = evalfis(x, OutputFismat);
plot(x, y, 'o', x, yy, 'x', x, y, 'y', x, yy, 'm');
legend('Training Data', 'ANFIS Output');
```

See also GENFIS1, ANFISMEX.

To determine the output of the FIS system for given input use *evalfis.*

EVALFIS Evaluation of a fuzzy inference system.

OUTPUT_STACK = EVALFIS(INPUT_STACK, FISMATRIX) computes the output of a fuzzy inference system specified by FISMATRIX.

INPUT_STACK can be a vector specifying the input vector, or a matrix where each row specifies an input vector.

OUTPUT_STACK is a stack of output (row) vectors.

For example:

```
[xx, yy] = meshgrid(-5:5);
input = [xx(:) yy(:)];
fismat = readfis('mam21');
out = evalfis(input, fismat);
surf(xx, yy, reshape(out, 11, 11))
title('evalfis')
```

produces

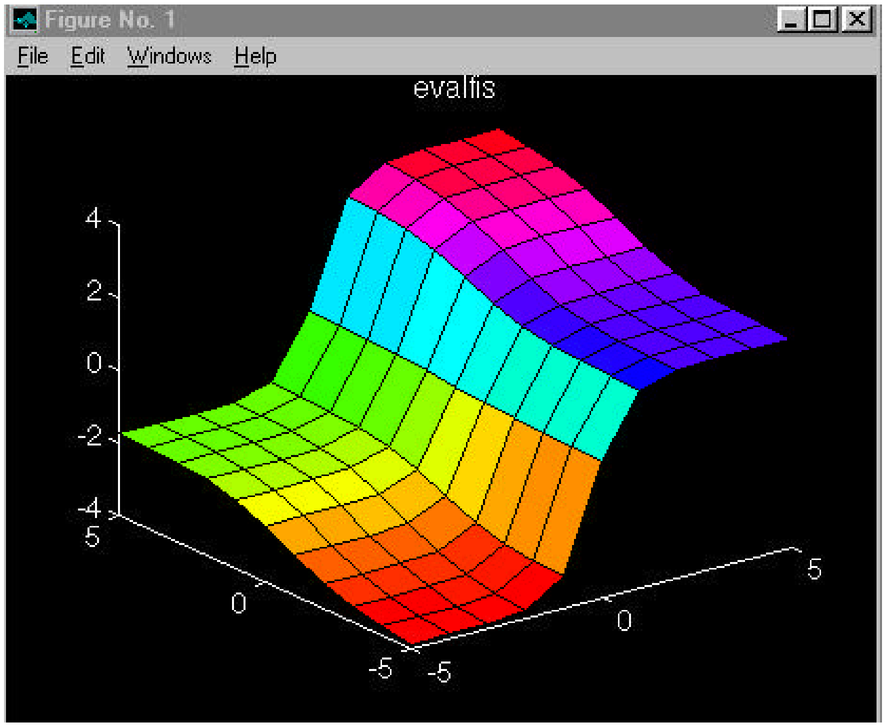


Figure No. 1
File
Edit
Windows
Help
evalfis