UFME7K-15-M Lecture 6 28/2/2018

# ANFIS: Adaptive Neuro-Fuzzy Inference System

Dr J Charles Sullivan

# Why?

- We have seen how to construct Fuzzy Inference Systems
  - and how to use them in control applications

- but aren’t there any tools to automate the process?
  - particularly the optimisation in order to meet some criterion of accuracy

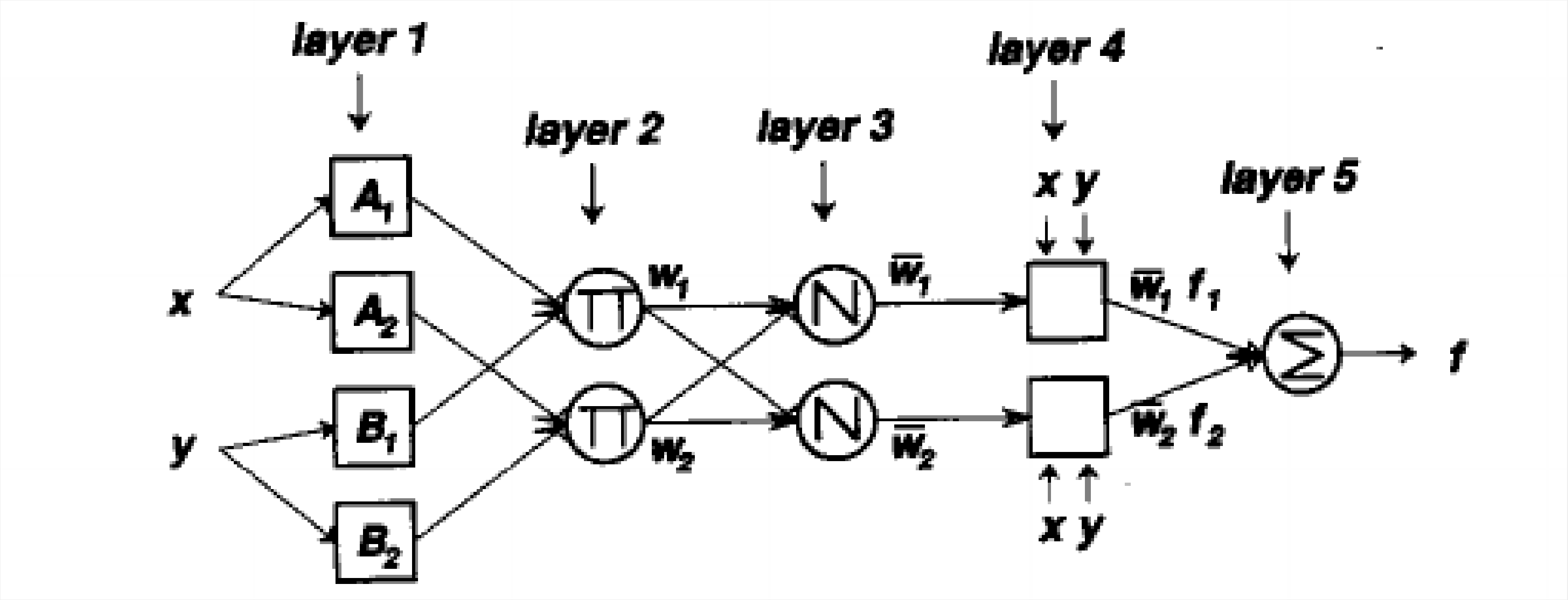
# What?

- applies **neurofuzzy** and **clustering** techniques to model and classify system behavior.
- can shape membership functions by training them with input/output data rather than specifying them manually.
- uses a back propagation algorithm alone or in combination with a least squares method, enabling fuzzy systems to learn from the data.
- supplies a fuzzy inference engine that can execute your fuzzy system as a stand-alone application or embedded in an external application.

# ANFIS defined

- ANFIS stands for Adaptive Neuro-Fuzzy Inference System.
- it is a **hybrid neuro-fuzzy** technique
  - brings learning capabilities of neural networks to fuzzy inference systems.
- the learning algorithm “tunes” the membership functions of a Sugeno-type Fuzzy Inference System
  - using training input-output data (supervised learning)

# 2 Rule ANFIS Network



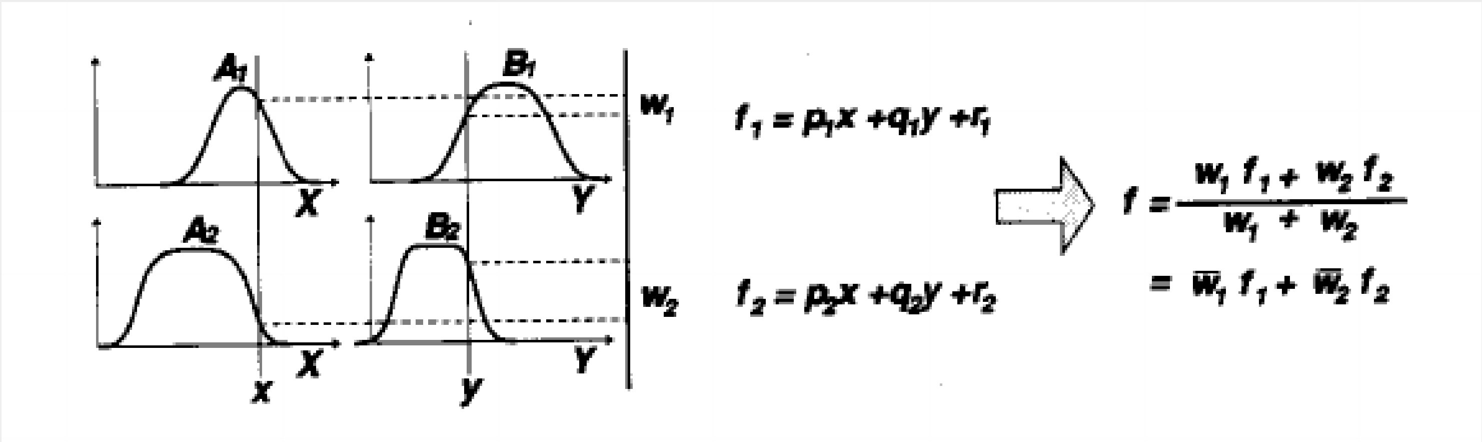
# ANFIS Learning Algorithm

- ANFIS can be trained by a ***hybrid learning algorithm*** (Jang, 1993)

- In the forward pass the algorithm uses a least-squares method to identify the consequent parameters in layer 4

- In the backward pass the errors are propagated backward and the premise parameters are updated by gradient descent.

# Example with 2 Rules




- Rule I: If x is $A_1$ and y is $B_1$, then $f_1 = p\ _1\ x + q_1y + r_1$
- Rule 2: If x is $A_2$ and y is $B_2$, then $f_2 = p_2x + q_2y + r_2$

# Layer 1

Every node $i$ in this layer is an adaptive node with a node function

$$O_i^1 = \mu_{A_i}(x)$$

where $O_i^1$ is the membership function of $A_i$ and it specifies the degree to which x satisfies $A_i$

# Layer 2

Imagine a set of archways

Every node in this layer is a fixed node labeled $\Pi$ which multiplies the incoming signals

For instance,

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

Each node output represents the firing strength of a rule.

# Layer 3

Every node in this layer is a fixed node labeled **N**

The *i*th node calculates the ratio of the *i*th rule's firing strength to the sum of all rules' firing strengths:

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

outputs of this layer are normalized firing strengths

# Layer 4

Every node i in this layer is an adaptive node with a node function

$$O_i^4 = \bar{w}_i f_i = \bar{w}_i (p_i x + q_i y + r_i)$$

where $\bar{w}_i$ is the output of layer 3, and $\{p_i, q_i, r_i\}$ is the parameter set.

Parameters in this layer are referred to as *consequent parameters*

# Layer 5

- The single node in this layer is a fixed node labeled $\sum$
- that computes the overall output as the summation of all incoming signals

$$O_i^5 = \sum_i \bar{w}_i f_i$$

# ANFIS is a Universal Approximator

- A zero-order Sugeno model has unlimited approximation power for matching any nonlinear function arbitrarily well
  - on a compact set.
  - when the number of rules is not restricted

- This can be proved using the Stone-Weierstrass theorem

# Advantages and Disadvantages

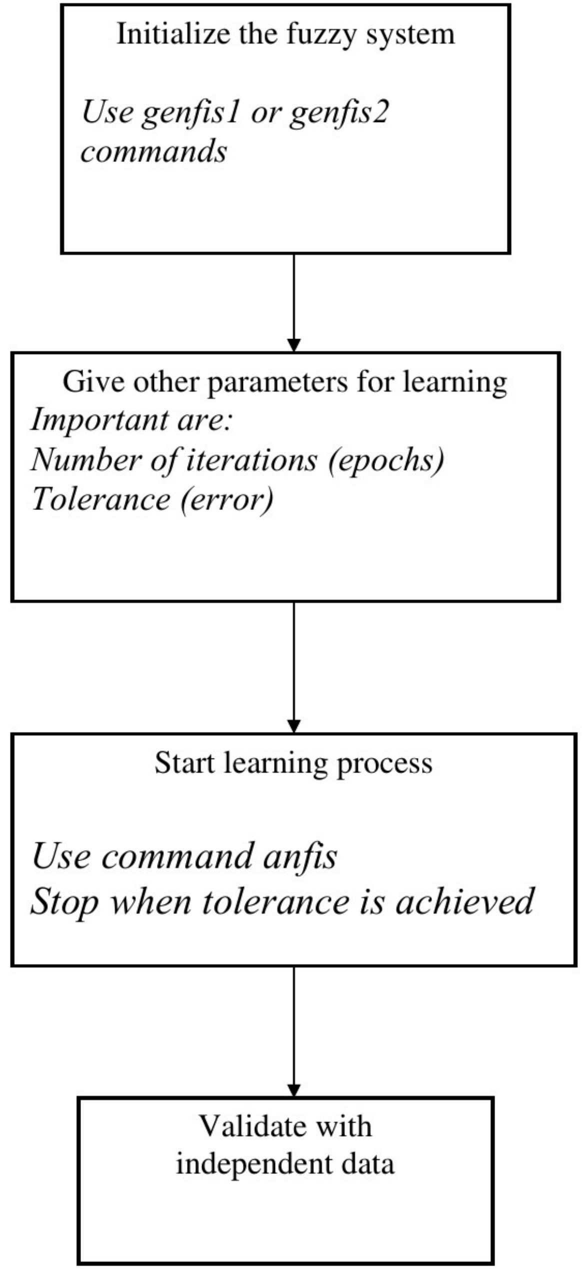
- Advantages
  - Automates the process of defining a Fuzzy Inference System
  - Built-in optimisation algorithms

- Disadvantages
  - Needs training data
  - Only Sugeno FIS
  - Not easy to interpret

# How to implement ANFIS networks using MATLAB

- We will be using MATLAB examples to illustrate the ANFIS Neuro-Fuzzy system
  - Using ***genfis*** to generate a FIS automatically
  - Using ***anfis*** to optimise the FIS by learning from training data
  - Using “check” data to validate

- Note that when we use lower-case “anfis” we are referring to the MATLAB implementation rather than the generic method


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

# A simple example

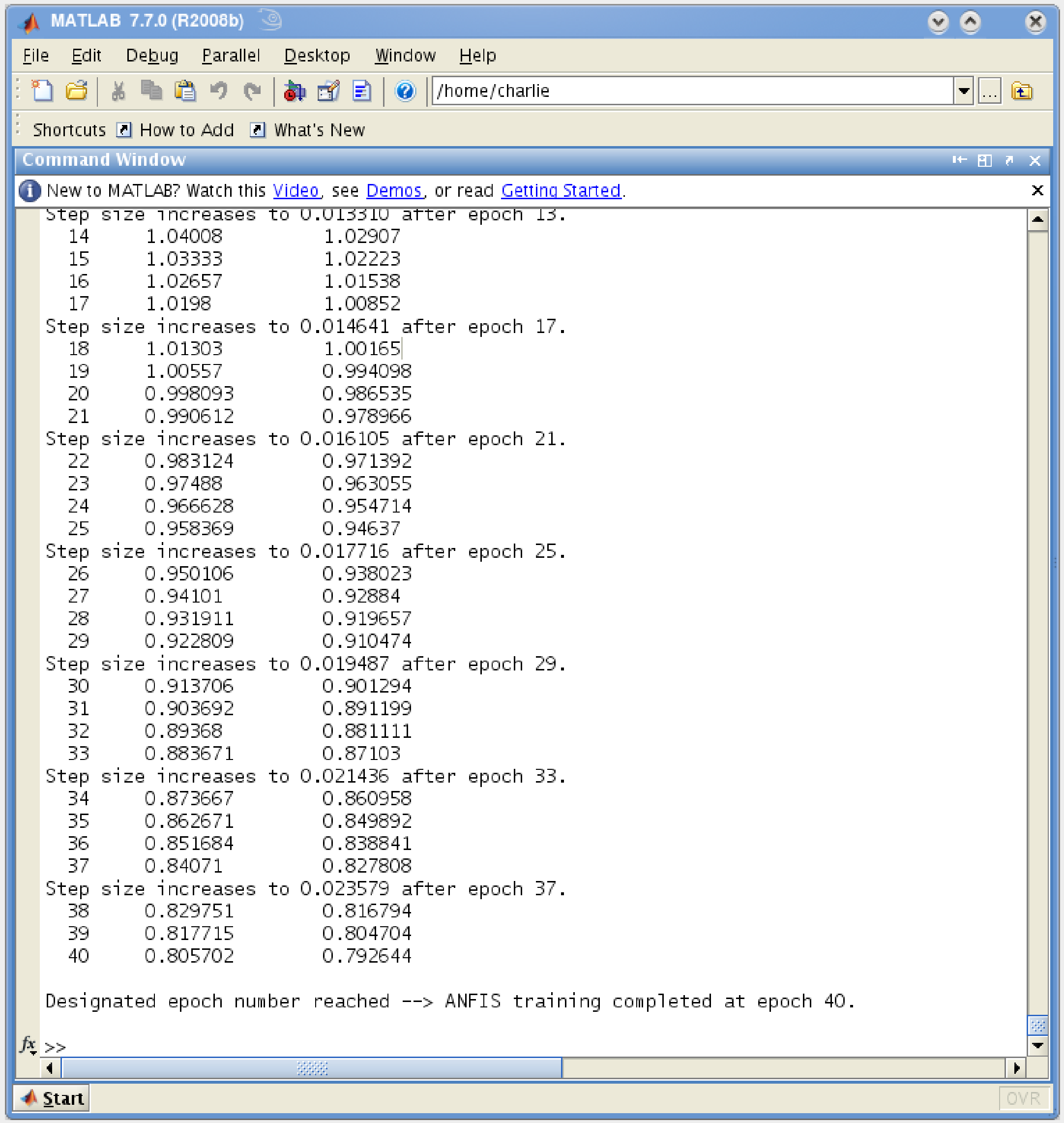
Editor - /home/charlie/fuzzy_ex2.m

```matlab
numPts=51;
x=linspace(-10,10,numPts)';
y=-2*x-x.^2;
data=[x y];
trndata=data(1:2:numPts,:);
chkdata=data(2:2:numPts,:);
 numMFs=5;
 mfType='gbellmf';
 fismat=(genfis1(trndata,numMFs,mfType))
 numEpochs=40;
[fismat1,trnErr,ss,fismat2,chkErr]=anfis(trndata,fismat,numEpochs,NaN,chkdata);
%Compare training and checking data to the fuzzy approximation:
anfis_y=evalfis(x(:,1),fismat1);

plot(trndata(:,1),trndata(:,2),'o',chkdata(:,1),chkdata(:,2),'x',x,anfis_y,'-')

hold on;
plot(x,y)
```

# ANFIS running

```
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      1.11156         1.10165
   2      1.10659         1.0966
   3      1.10161         1.09154
   4      1.09662         1.08647
   5      1.09163         1.08139
Step size increases to 0.011000 after epoch 5.
   6      1.08663         1.07631
   7      1.08112         1.07071
   8      1.0756          1.06511
   9      1.07007         1.0595
Step size increases to 0.012100 after epoch 9.
  10      1.06453         1.05387
  11      1.05843         1.04768
  12      1.05233         1.04149
  13      1.04621         1.03528
Step size increases to 0.013310 after epoch 13.
  14      1.04008         1.02907
  15      1.03333         1.02223
  16      1.02657         1.01538
  17      1.0198          1.00852
Step size increases to 0.014641 after epoch 17.
  18      1.01303         1.00165
  19      1.00557         0.994098
  20      0.998093        0.986535
  21      0.990612        0.978966
```

# End of run



MATLAB 7.7.0 (R2008b)
File Edit Debug Parallel Desktop Window Help
/home/charlie
Shortcuts How to Add What's New
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Step size increases to 0.013310 after epoch 13.
14 1.04008 1.02907
15 1.03333 1.02223
16 1.02657 1.01538
17 1.0198 1.00852
Step size increases to 0.014641 after epoch 17.
18 1.01303 1.00165
19 1.00557 0.994098
20 0.998093 0.986535
21 0.990612 0.978966
Step size increases to 0.016105 after epoch 21.
22 0.983124 0.971392
23 0.97488 0.963055
24 0.966628 0.954714
25 0.958369 0.94637
Step size increases to 0.017716 after epoch 25.
26 0.950106 0.938023
27 0.94101 0.92884
28 0.931911 0.919657
29 0.922809 0.910474
Step size increases to 0.019487 after epoch 29.
30 0.913706 0.901294
31 0.903692 0.891199
32 0.89368 0.881111
33 0.883671 0.87103
Step size increases to 0.021436 after epoch 33.
34 0.873667 0.860958
35 0.862671 0.849892
36 0.851684 0.838841
37 0.84071 0.827808
Step size increases to 0.023579 after epoch 37.
38 0.829751 0.816794
39 0.817715 0.804704
40 0.805702 0.792644
Designated epoch number reached --> ANFIS training completed at epoch 40.
>>
Start
OVR

# The result

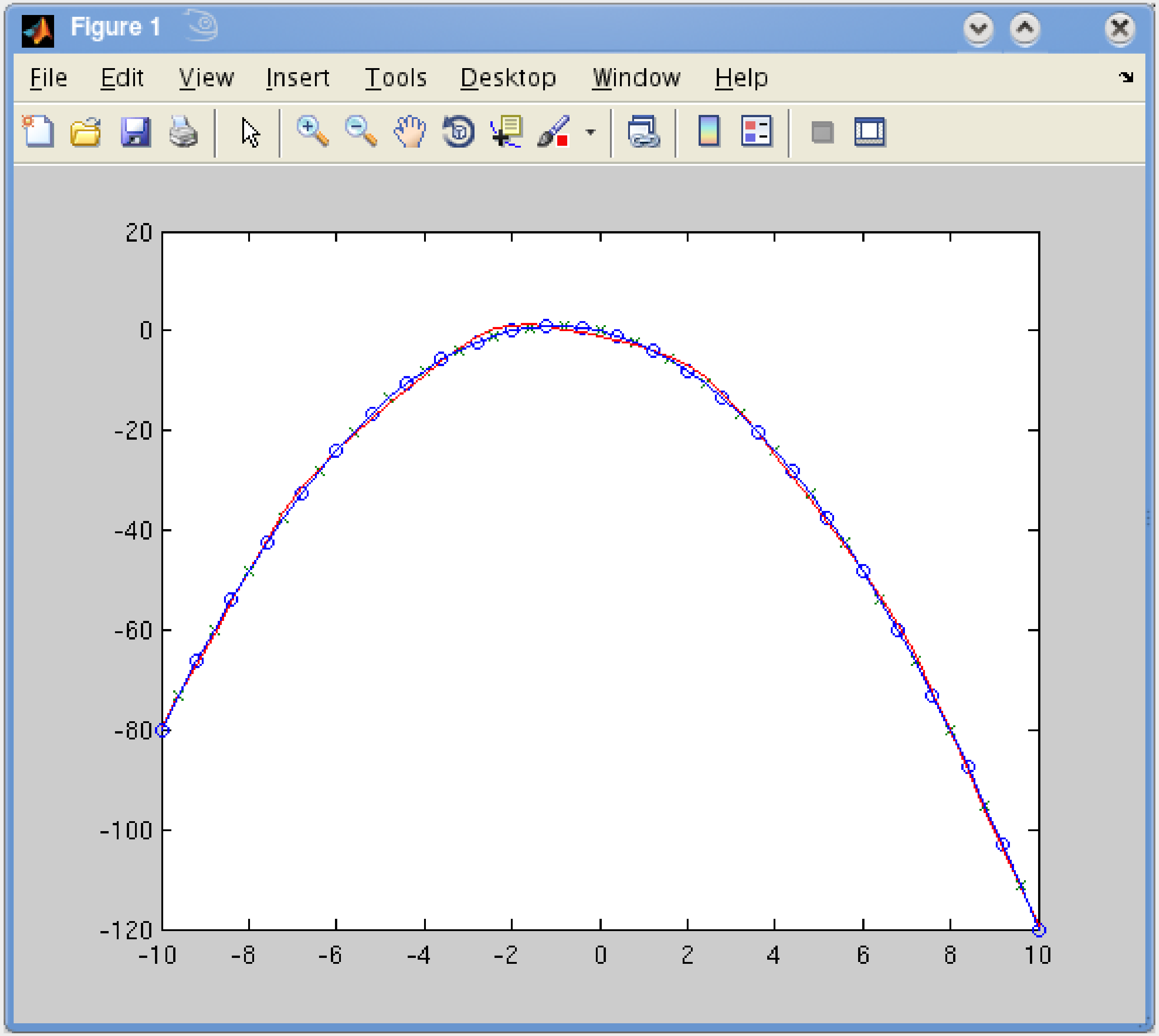


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
20
0
-20
-40
-60
-80
-100
-120
-10
-8
-6
-4
-2
0
2
4
6
8
10

# Another example

```
numPts=51;
x=linspace(-1,1,numPts)';
y=0.6*sin(pi*x)+0.3*sin(3*pi*x)+0.1*sin(5*pi*x);
data=[x y];
trndata=data(1:2:numPts,:);
chkdata=data(2:2:numPts,:);
```

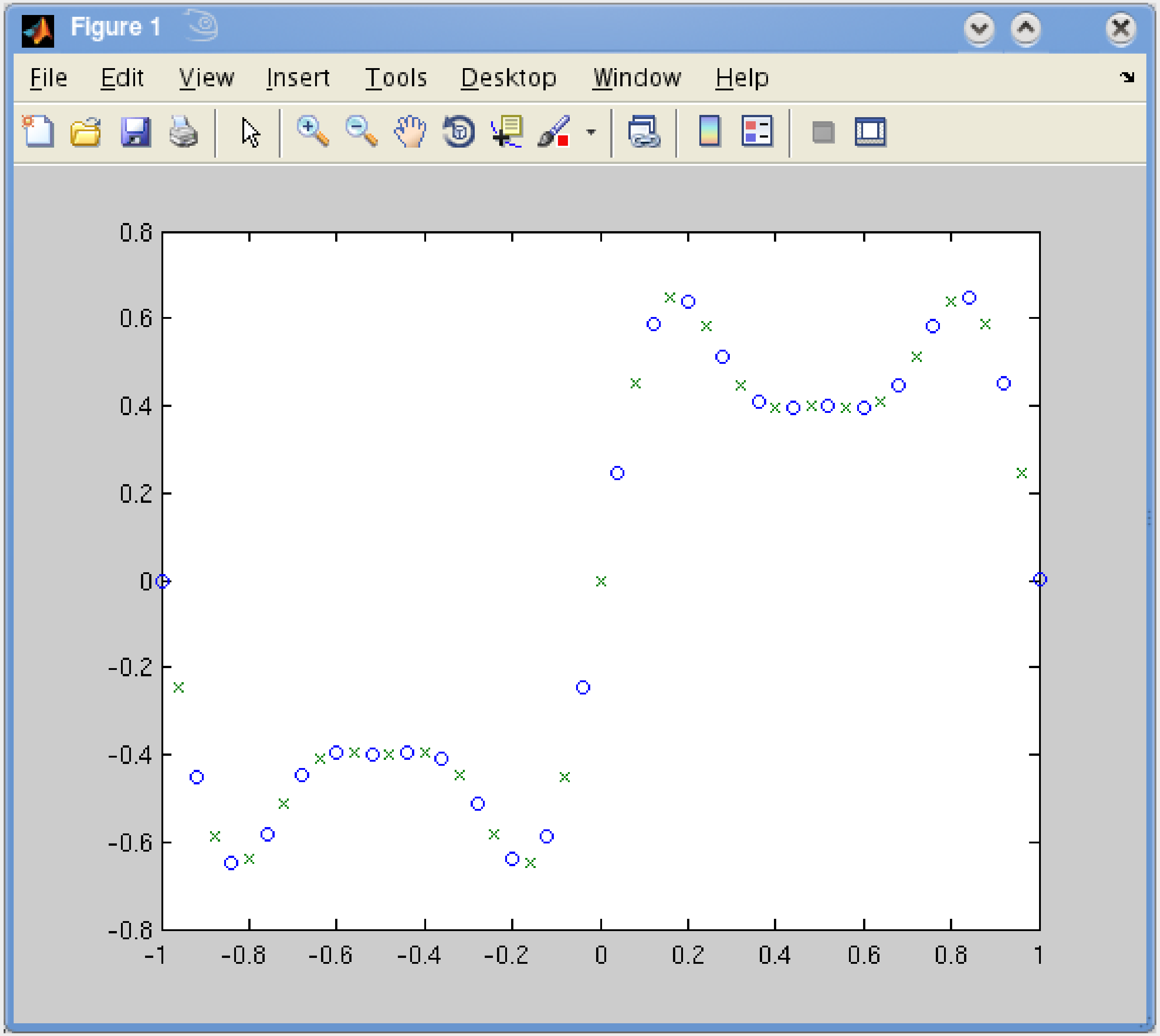
setting up list.

popul. list.

Split data

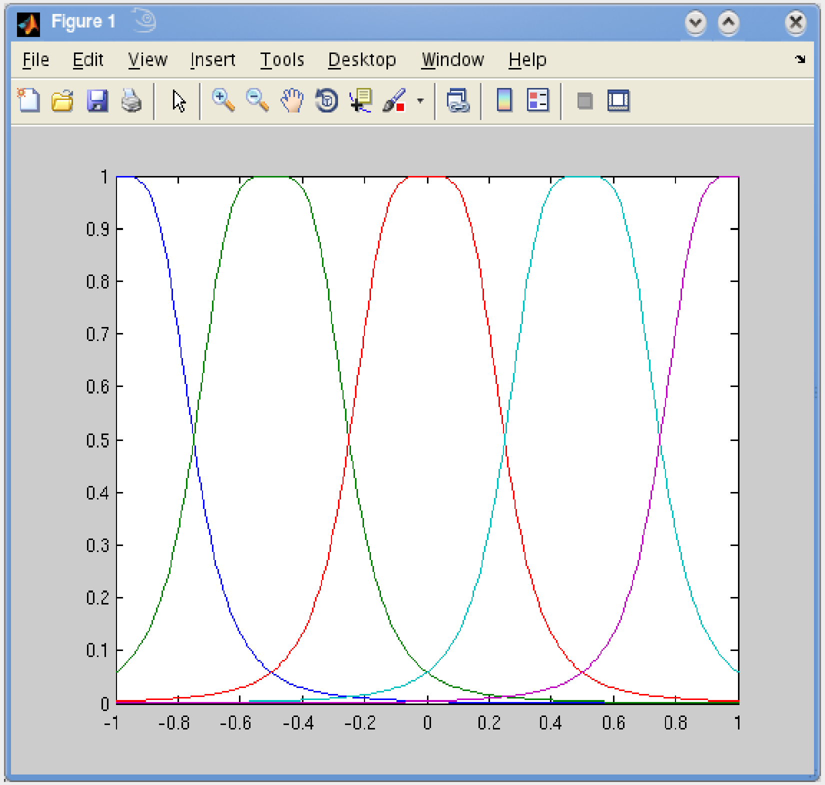
```
plot(trndata(:,1),trndata(:,2),'o',chkdata(:,1),c
hkdata(:,2),'x')
```

# Training and Validation Data



# Initial Membership Functions Generated by Genfis1

```
numMFs=5;

mfType='gbellmf';

fismat =genfis1(trndata,numMFs,mfType);


[x,mf]=plotmf(fismat,'input',1);

plot(x,mf)
```



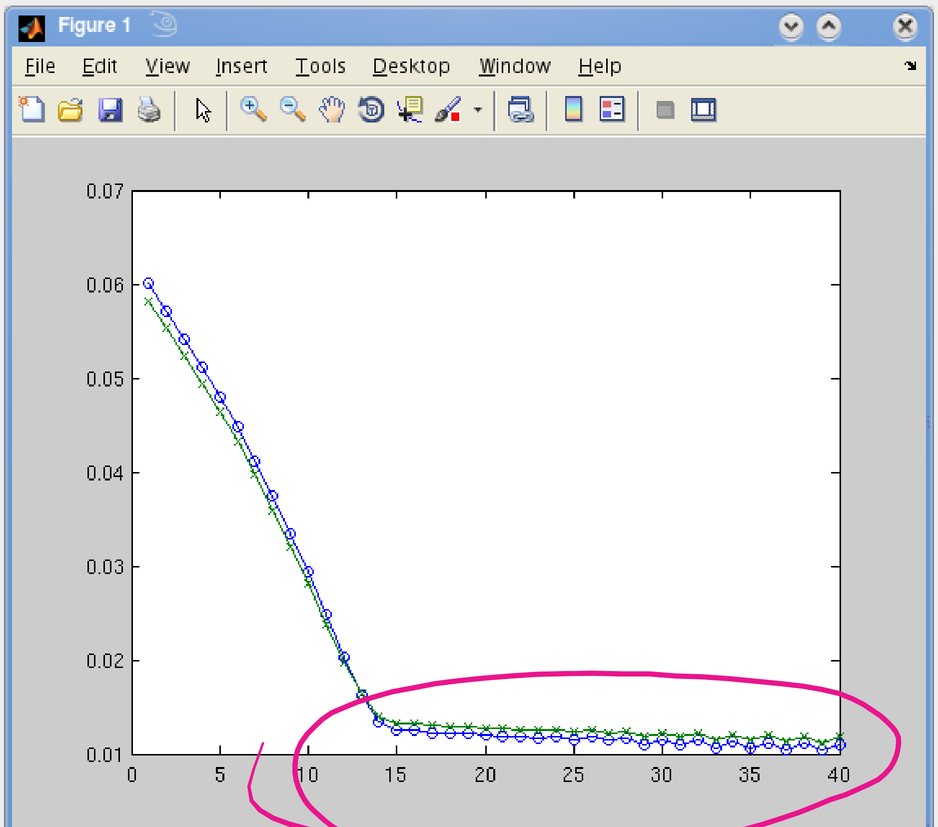
# The anfis command

```
numEpochs=40;

[fismat1,trnErr,ss,fismat2,chkErr]=anfis(trndata,fismat,numEpochs,NaN,chkdata);
```

# Using evalfis to evaluate the trained system

```
trnOut=evalfis(trndata(:,1),fismat1);

trnRMSE=norm(trnOut-
trndata(:,2))/sqrt(length(trnOut))  ,

 chkOut=evalfis(chkdata(:,1),fismat2);

 chkRMSE=norm(chkOut-
chkdata(:,2))/sqrt(length(chkOut))
```

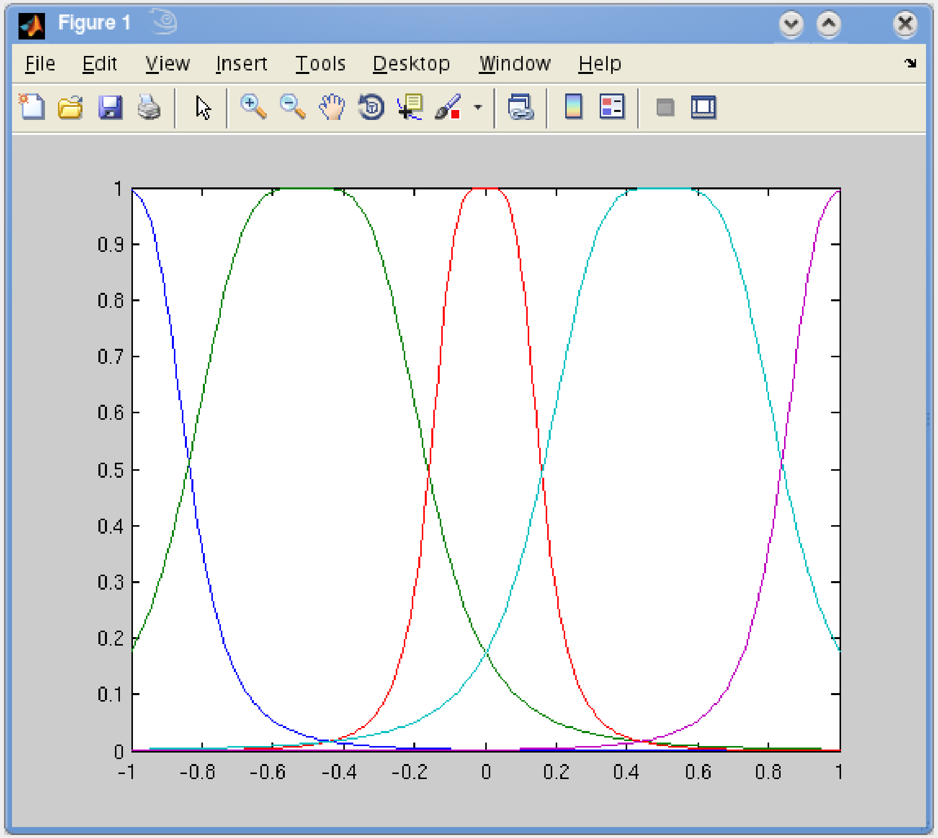
# Errors reducing during trainning




```
epoch=1:numEpochs;

plot(epoch,trnErr,'o',epoch,chkErr,'x')

 hold on;

 plot(epoch,[trnErr chkErr])

 hold off;
```

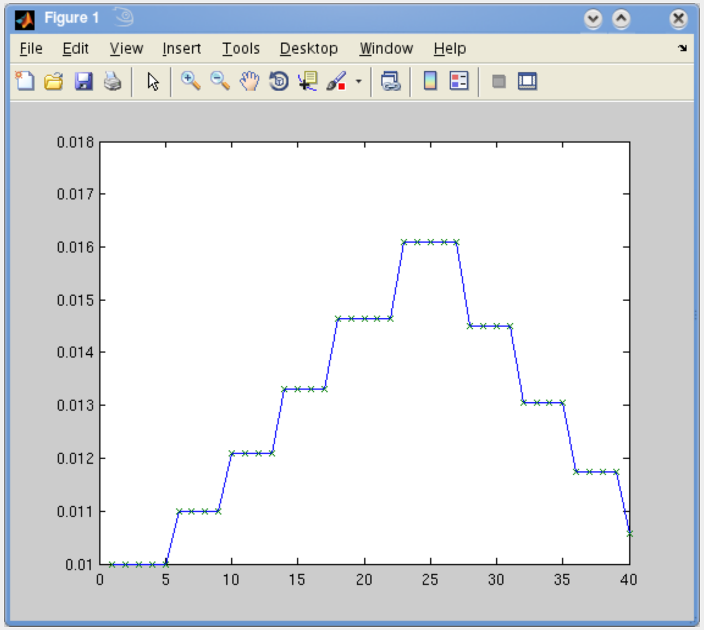
Note the monotonic decrease!
Learning has worked!

# Membership functions after training




```
[x,mf]=plotmf(fismat1,'input',1);
plot(x,mf)
```

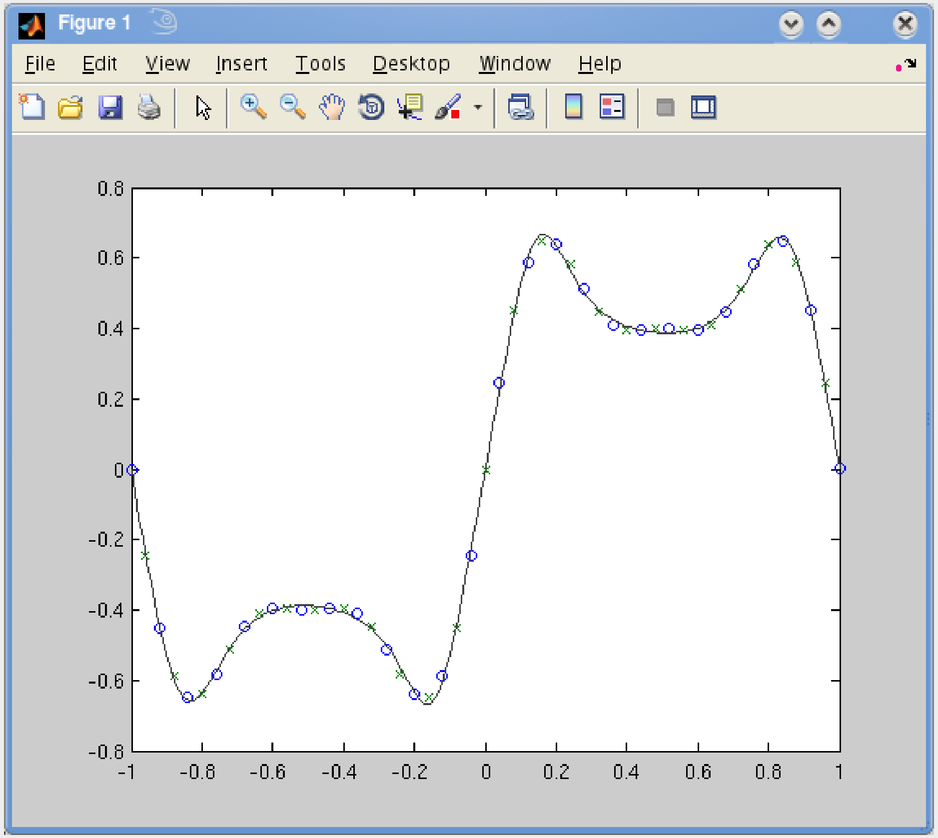
# Step size adaptation




plot(epoch,ss,'-',epoch,ss,'x')

# The output compared with training and check data

```
anfis_y=evalfis(x(:,1),fismat1);

plot(trndata(:,1),trndata(:,2),'o',chkdata(:,1),chkdata(:,2),'x',x,anfis_y,'-')
```



# A bigger example

In this example we apply the genfis2 function to model the relationship between the number of automobile trips generated from an area and the area's demographics.

Demographic and trip data are from 100 traffic analysis zones in New Castle County, Delaware.

Five demographic factors are considered:

population, number of dwelling units, vehicle ownership, median household income, and total employment.

Hence the model has five input variables and one output variable

```
mytripdata
subplot(2,1,1), plot(datin)
subplot(2,1,2), plot(datout)

fismat=genfis2(datin,datout,0.5);
fuzout=evalfis(datin,fismat);
trnRMSE=norm(fuzout-datout)/sqrt(length(fuzout));
chkfuzout=evalfis(chkdatin,fismat);
chkRMSE=norm(chkfuzout-chkdatout)/sqrt(length(chkfuzout))
```

# Optimisation

At this point, we can use the optimization capability of ANFIS to improve the model.

First, we will try using a relatively short anfis training (50 epochs)

without implementing the checking data option, but test the resulting FIS model against the test data.

```
fismat2=anfis([datin datout],fismat,[20 0 0.1]);
```

After the training is done, we type

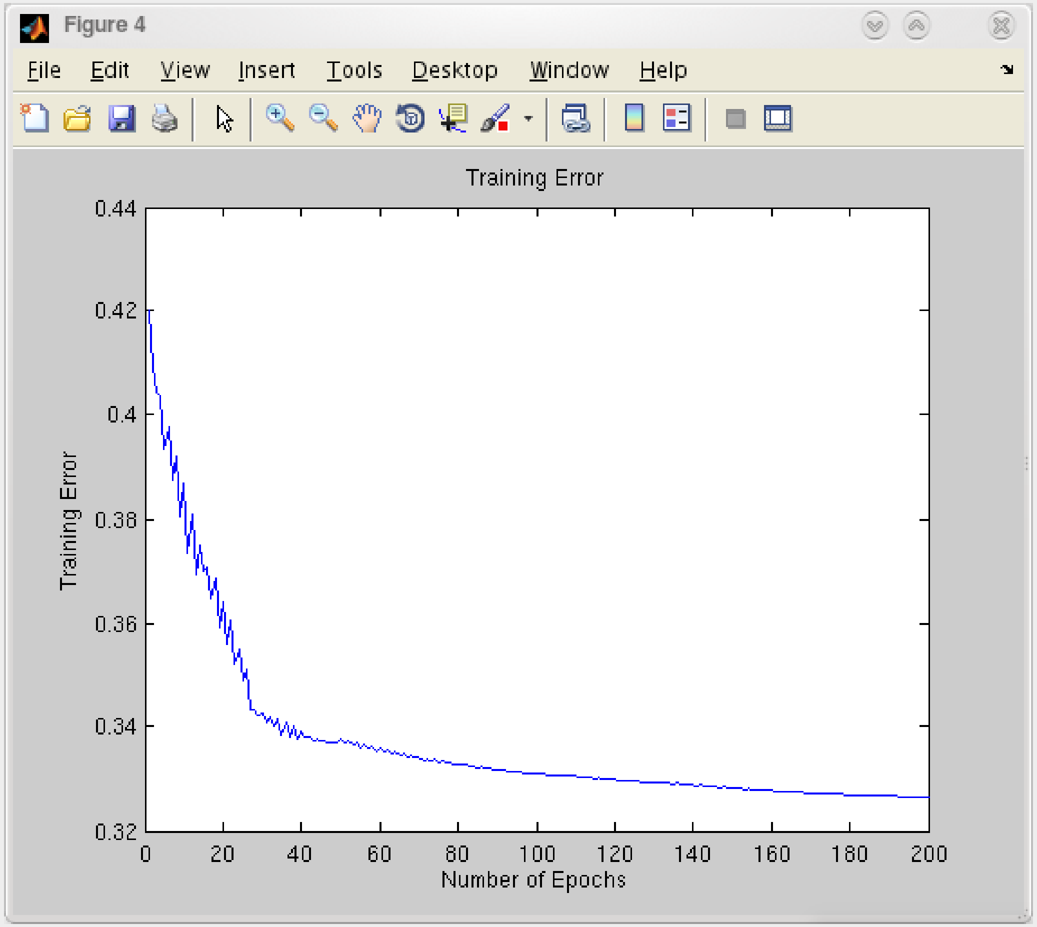
```
fuzout2=evalfis(datin,fismat2);
trnRMSE2=norm(fuzout2-datout)/sqrt(length(fuzout2));

chkfuzout2=evalfis(chkdatin,fismat2);
chkRMSE2=norm(chkfuzout2-chkdatout)/sqrt(length(chkfuzout2));
```

# Training over a longer period

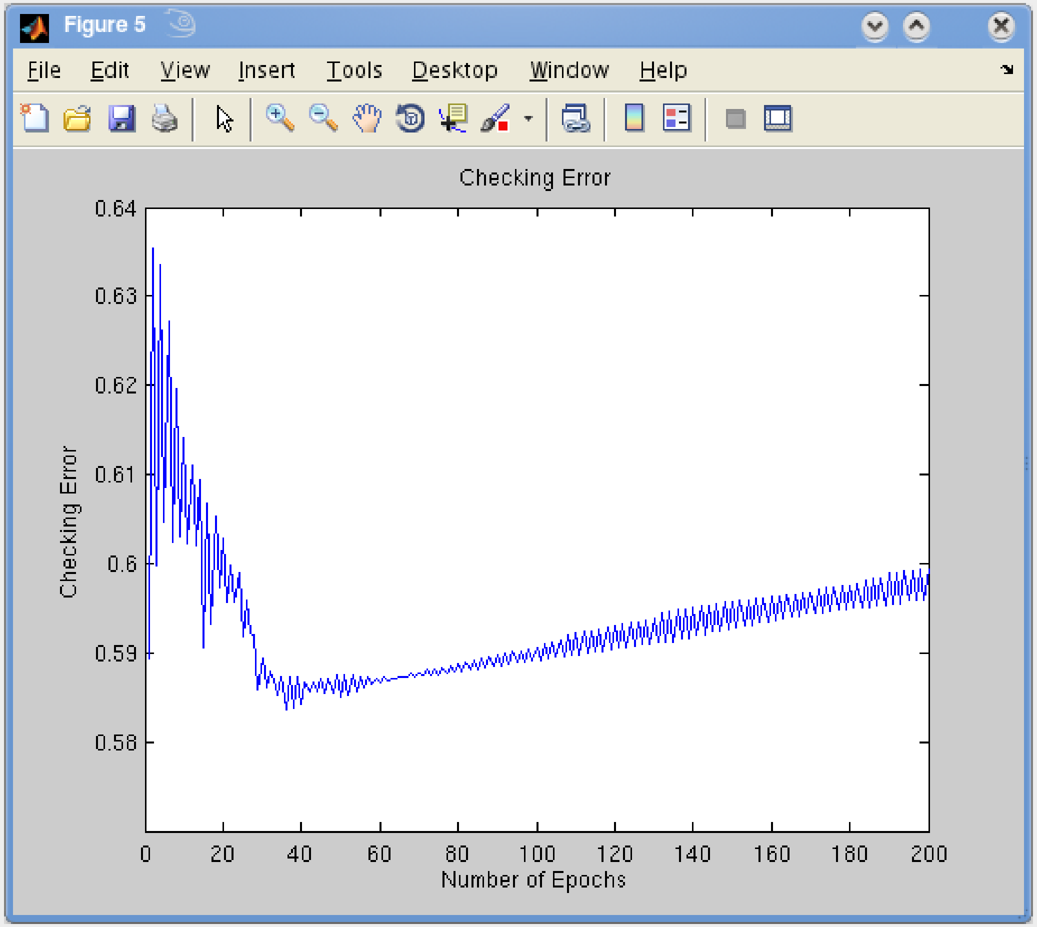
what happens if we carry out a longer (200 epoch) training of this system using anfis, including its checking data option.

```
[fismat3,trnErr,stepSize,fismat4,chkErr]= ...
        anfis([datin datout],fismat2,[200 0 0.1],[], ...
        [chkdatin chkdatout]);
figure
plot(trnErr)
title('Training Error')
xlabel('Number of Epochs')
ylabel('Training Error')

figure
plot(chkErr)
title('Checking Error')
xlabel('Number of Epochs')
ylabel('Checking Error')
```

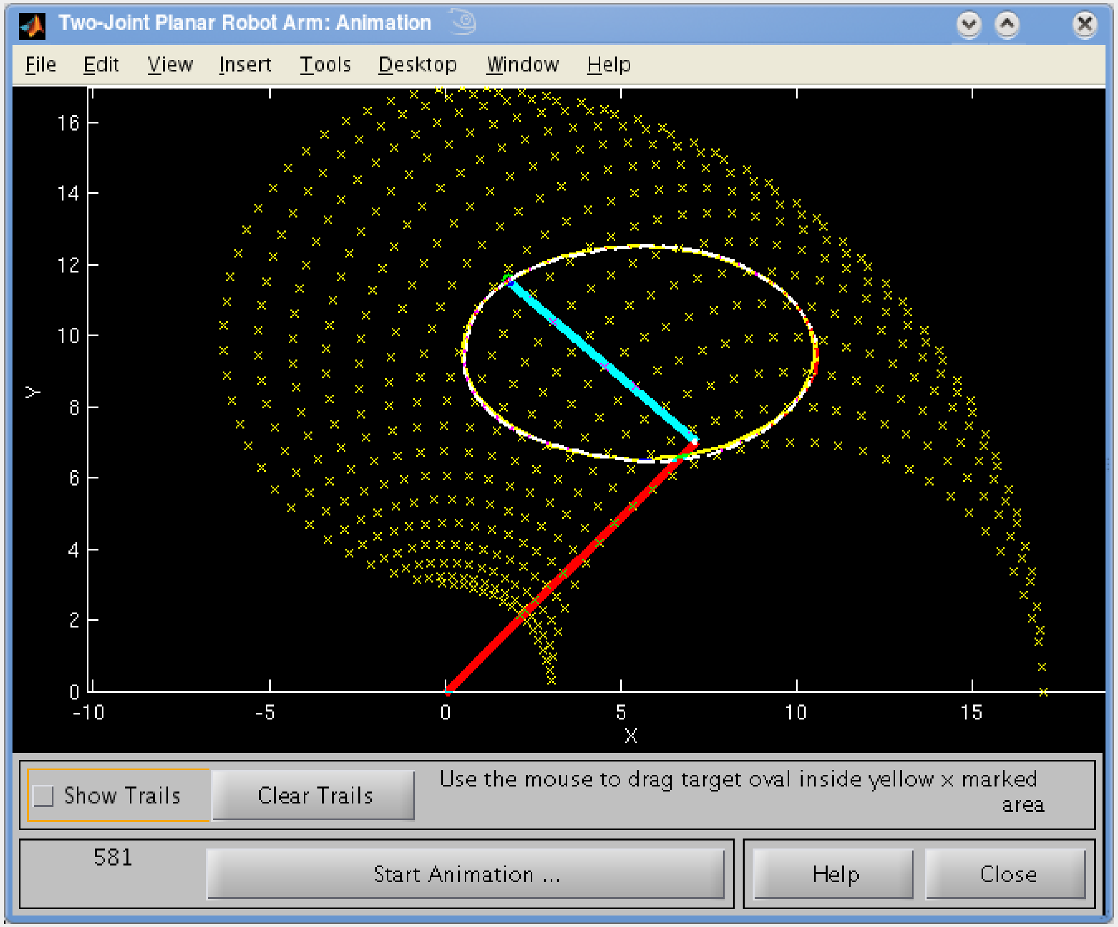
# Training error



# Validation Error (showing overfitting)



# The invkine example GUI



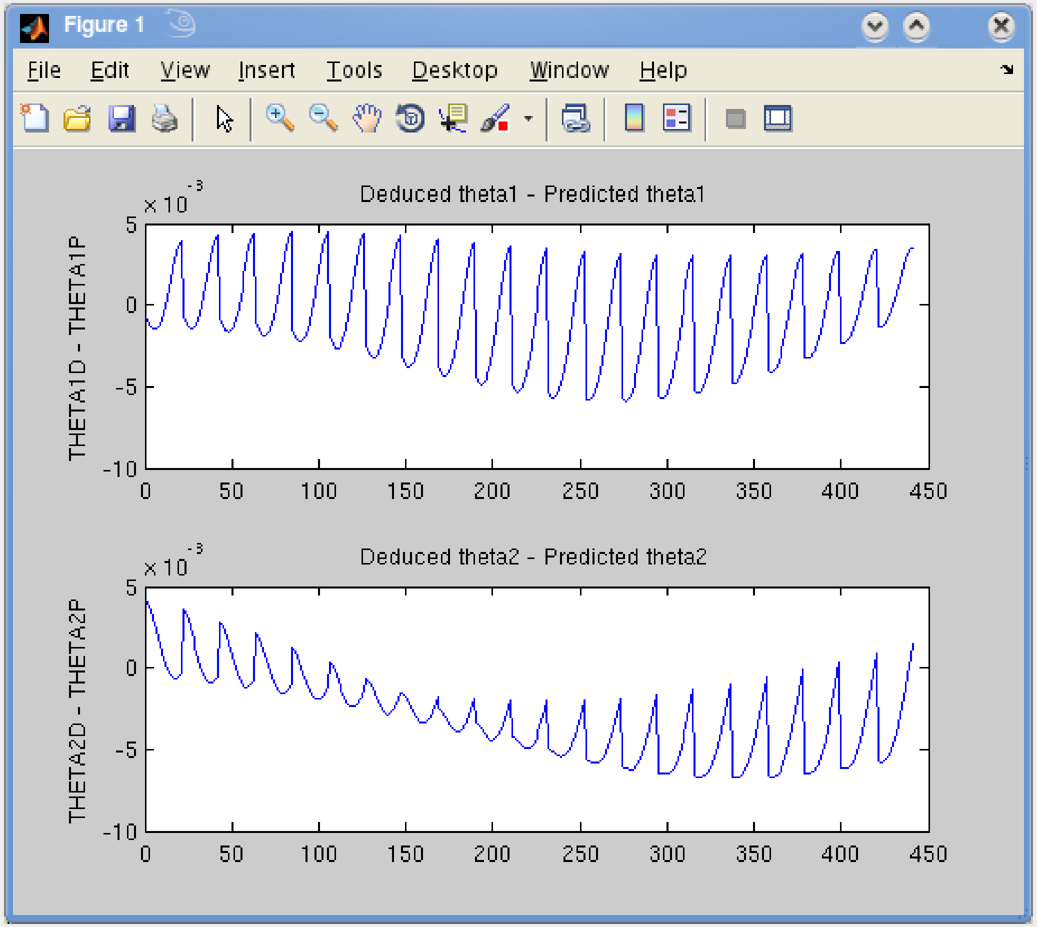
# Setting up training data for 2D Inverse Kinematics

```
l1 = 10; % length of first arm link
l2 = 7; % length of second arm link

theta1 = 0:0.1:pi/2; % all possible theta1 values
theta2 = 0:0.1:pi; % all possible theta2 values

[THETA1, THETA2] = meshgrid(theta1, theta2);
% generate a grid of theta1 and theta2 values

X = l1 * cos(THETA1) + l2 * cos(THETA1 + THETA2);
% compute x coordinates
Y = l1 * sin(THETA1) + l2 * sin(THETA1 + THETA2);
% compute y coordinates

data1 = [X(:) Y(:) THETA1(:)]; % create x-y-theta1 dataset
data2 = [X(:) Y(:) THETA2(:)]; % create x-y-theta2 dataset
```

# The anfis command

```
anfis1 = anfis(data1, 7, 150, [0,0,0,0]);
% train first ANFIS network

anfis2 = anfis(data2, 6, 150, [0,0,0,0]);
 % train second ANFIS network
```

# Error compared with Analytical Solution



# References


- Jang, J.-S. R., "Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm," Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91), pp. 762-767, July 1991.

- Jang, J.-S. R., "ANFIS: Adaptive-Network-based Fuzzy Inference Systems," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 3, pp. 665-685, May 1993.

# Further Reading

Passino and Yukovich, *Fuzzy Control,* Addison Wesley, 1998

http://www2.ece.ohio-state.edu/~passino/FCbook.pdf

provides a control-engineering perspective on fuzzy systems

*"We are concerned with both the construction of nonlinear controllers for challenging real-world applications and with gaining a fundamental understanding of the dynamics of fuzzy control systems so that we can mathematically verify their properties (e.g., stability) before implementation.*

*We emphasize engineering evaluations of performance and comparative analysis with conventional control methods. We introduce adaptive methods for identification, estimation, and control. We examine numerous examples, applications, and design and implementation case studies"*