

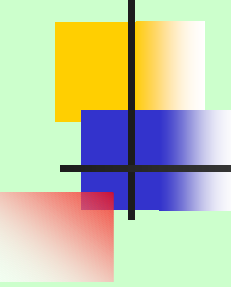
Fuzzy Modelling and Identification

**Fuzzy Clustering with Application
to Data-Driven Modelling**



Introduction

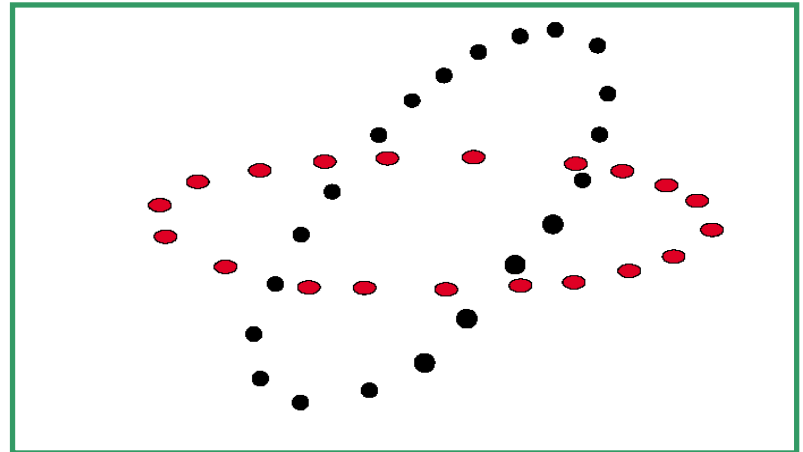
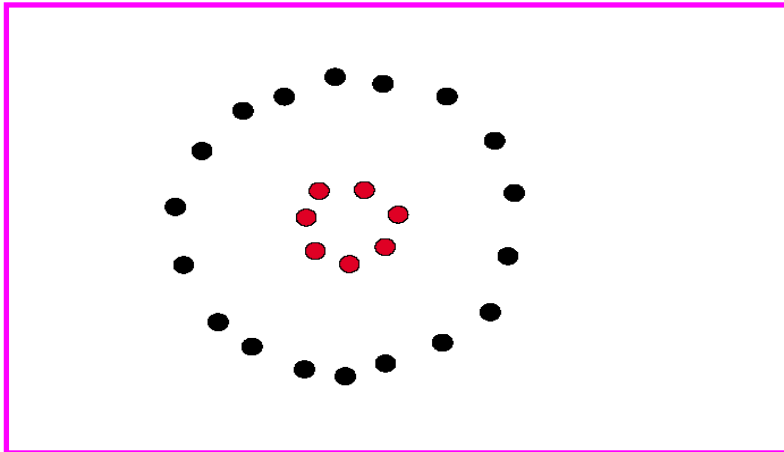
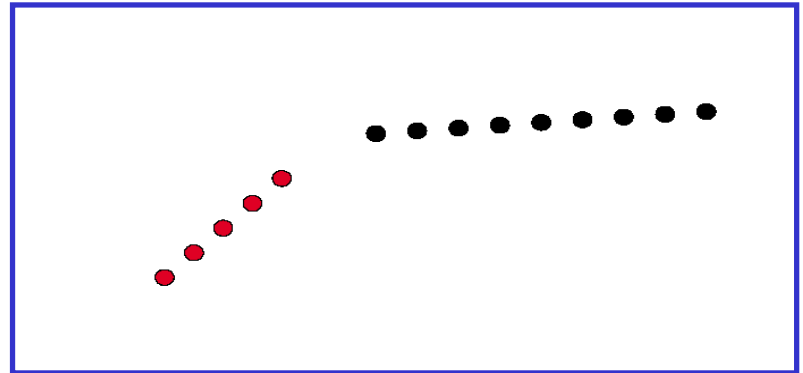
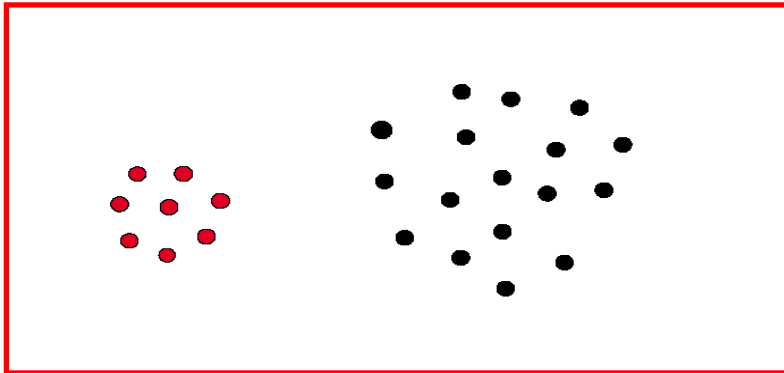
- The ability to cluster data (concepts, perceptions, etc.)
 - essential feature of human intelligence.
- A cluster is a set of objects that are more similar to each other than to objects from other clusters.
- Applications of clustering techniques in pattern recognition and image processing.
- Some machine-learning techniques are based on the notion of similarity (decision trees, case-based reasoning)
- Non-linear regression and black-box modelling can be based on the partitioning data into clusters.



Section Outline

- **Basic concepts in clustering**
 - data set
 - partition matrix
 - distance measures
- **Clustering algorithms**
 - fuzzy c-means
 - Gustafson–Kessel
- **Application examples**
 - system identification and modelling
 - diagnosis

Examples of Clusters

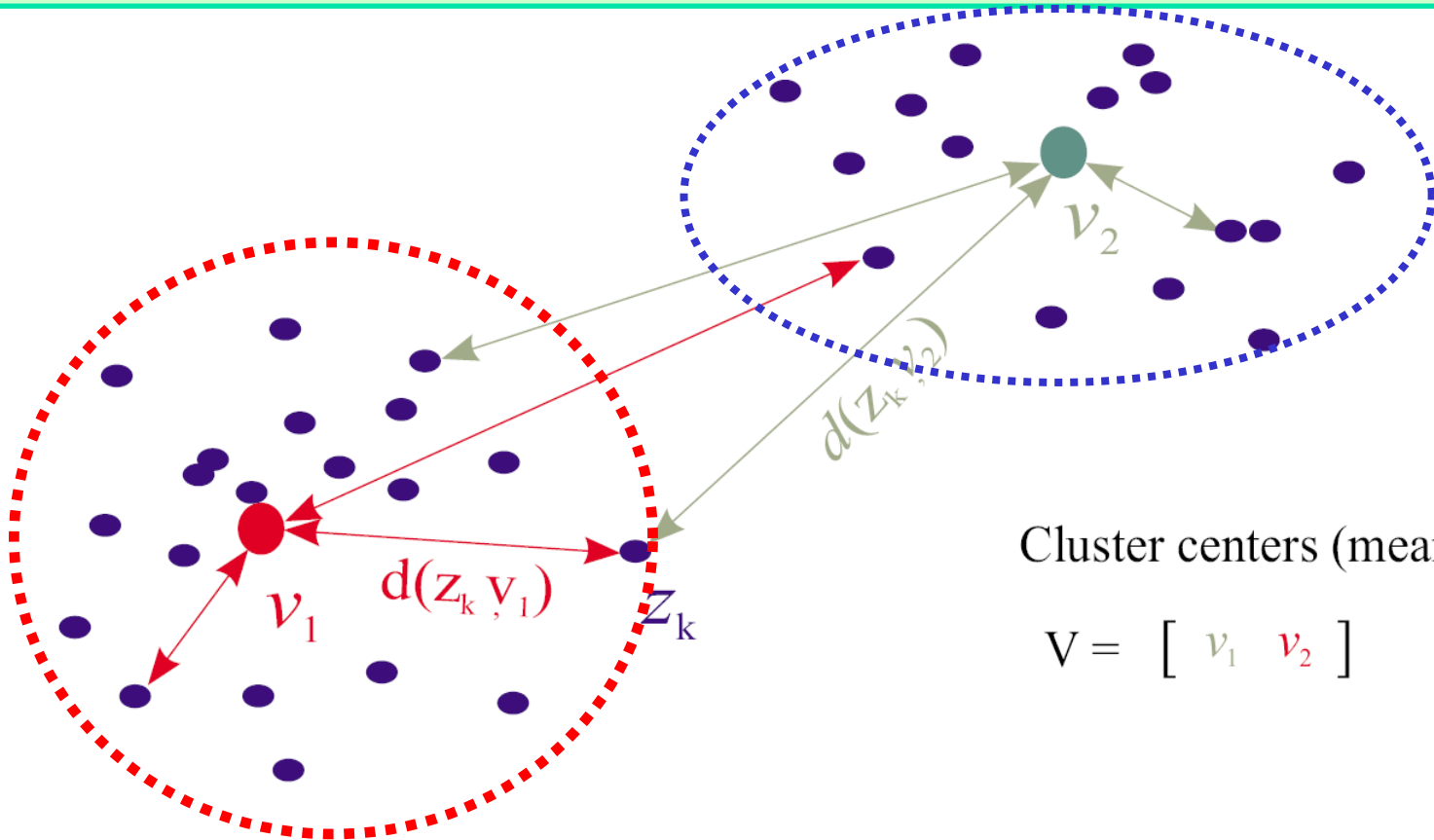




Problem Formulation

- **Given** is a set of data in R^n and the (estimated) number of clusters to look for (a difficult problem, more on this later).
- **Find** the partitioning of the data into subsets (clusters), such that samples within a subset are more similar to each other than to samples from other subsets.
- **Similarity** is mathematically formulated by using a distance measure (i.e., a dissimilarity function).
- Usually, each cluster will have a **prototype** and the distance is measured from this prototype.

Distance Measure



Cluster centers (means):

$$V = [v_1 \quad v_2]$$



Distance Measures

➤ **Euclidean norm:**

- $d^2(\mathbf{z}_j, \mathbf{v}_i) = (\mathbf{z}_j - \mathbf{v}_i)^T (\mathbf{z}_j - \mathbf{v}_i)$

➤ **Inner-product norm:**

- $d^2_{\mathbf{A}_i}(\mathbf{z}_j, \mathbf{v}_i) = (\mathbf{z}_j - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{z}_j - \mathbf{v}_i)$

➤ **Many other possibilities . . .**

Fuzzy Clustering: an Optimisation Approach

➤ Objective function (least-squares criterion):

$$J(\mathbf{Z}; \mathbf{V}, \mathbf{U}, \mathbf{A}) = \sum_{i=1}^c \sum_{j=1}^N \mu_{i,j}^m d_{\mathbf{A}_i}^2(\mathbf{z}_j, \mathbf{v}_i)$$

➤ subject to constraints:

$$0 \leq \mu_{i,j} \leq 1, \quad i = 1, \dots, c, \quad j = 1, \dots, N \quad \text{membership degree}$$

$$0 < \sum_{j=1}^N \mu_{i,j} < 1, \quad i = 1, \dots, c \quad \text{no cluster empty}$$

$$\sum_{i=1}^c \mu_{i,j} = 1, \quad j = 1, \dots, N \quad \text{total membership}$$



Fuzzy Algorithm

Repeat:

1. Compute cluster prototypes (means):

$$v_i = \frac{\sum_{k=1}^N \mu_{i,k}^m \mathbf{z}_k}{\sum_{k=1}^N \mu_{i,k}^m}$$

2. Calculate distances:

$$d_{ik} = (\mathbf{z}_k - \mathbf{v}_i)^T (\mathbf{z}_k - \mathbf{v}_i)$$

3. Update partition matrix:

$$\mu_{ik} = \frac{1}{\sum_{j=1}^c (d_{ik}/d_{jk})^{1/(m-1)}}$$

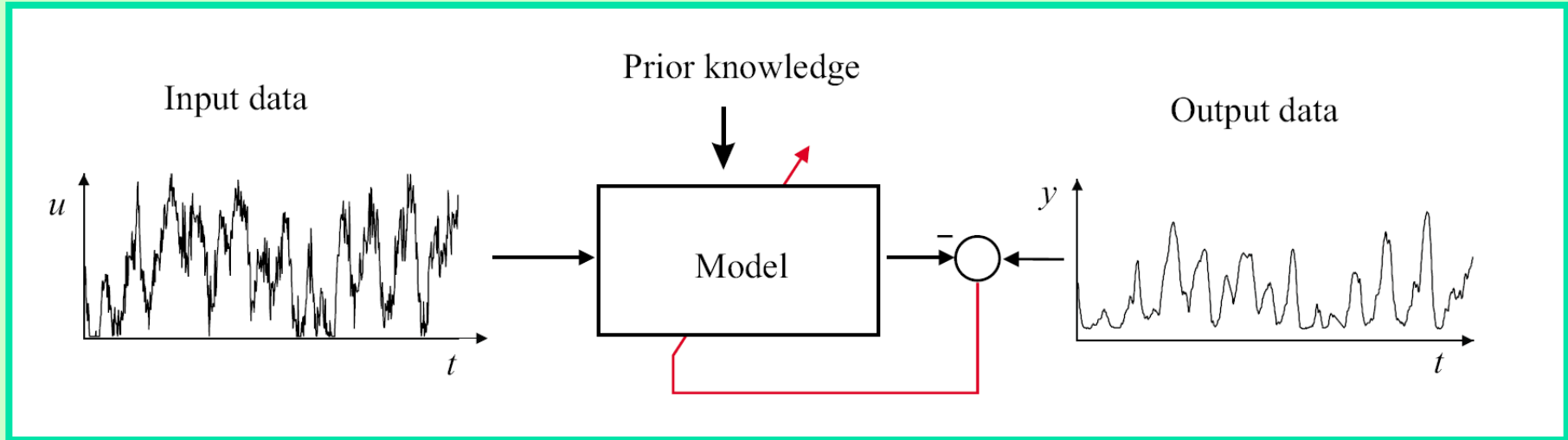
until

$$\|\Delta \mathbf{U}\| < \epsilon$$

$$(i = 1, \dots, c. k = 1, \dots, N)$$

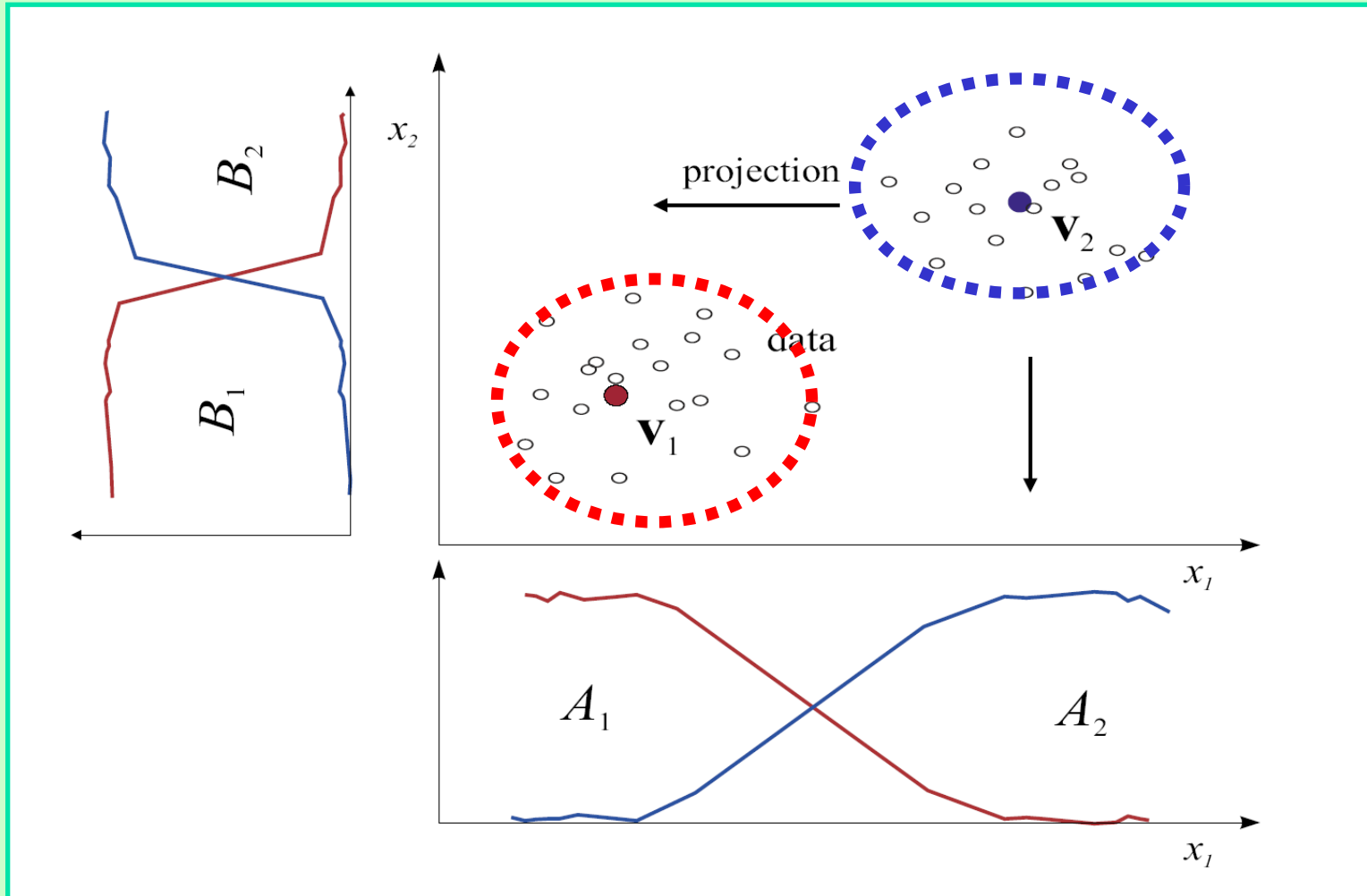
Data-Driven (Black-Box)

Modelling

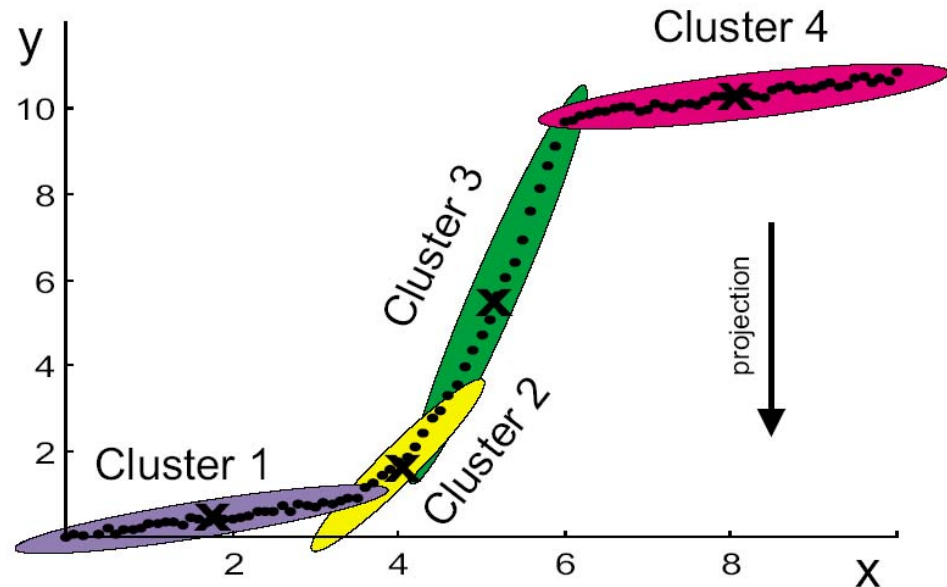


- **Linear model** (for linear systems only, limited in use)
- **Neural network** (black box, unreliable extrapolation)
- **Rule-based model** (more transparent, 'grey-box')

Extraction of Rules by Fuzzy Clustering



Extraction of Rules by Fuzzy Clustering



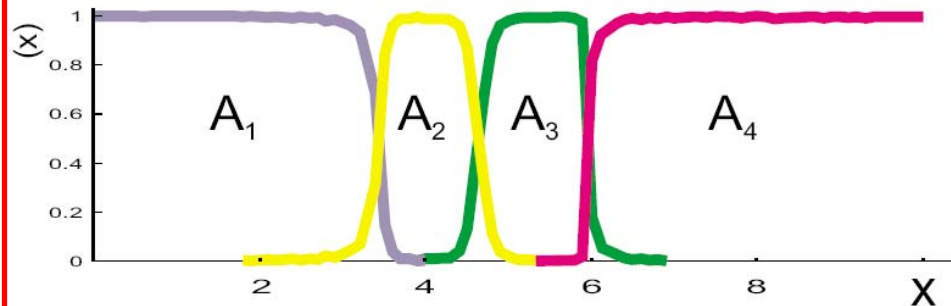
Takagi-Sugeno model

Rule-based description:

If x is A_1 then $y = a_1x + b_1$

If x is A_2 then $y = a_2x + b_2$

etc...



Example: Non-linear Autoregressive System (NARX)

$$x(k+1) = f(x(k)) + \epsilon(k)$$

$$f(x) = \begin{cases} 2x - 2, & 0.5 < x \\ -2x, & -0.5 \leq x < 0.5 \\ 2x + 2, & x \leq -0.5 \end{cases}$$

Structure Selection and Data Preparation

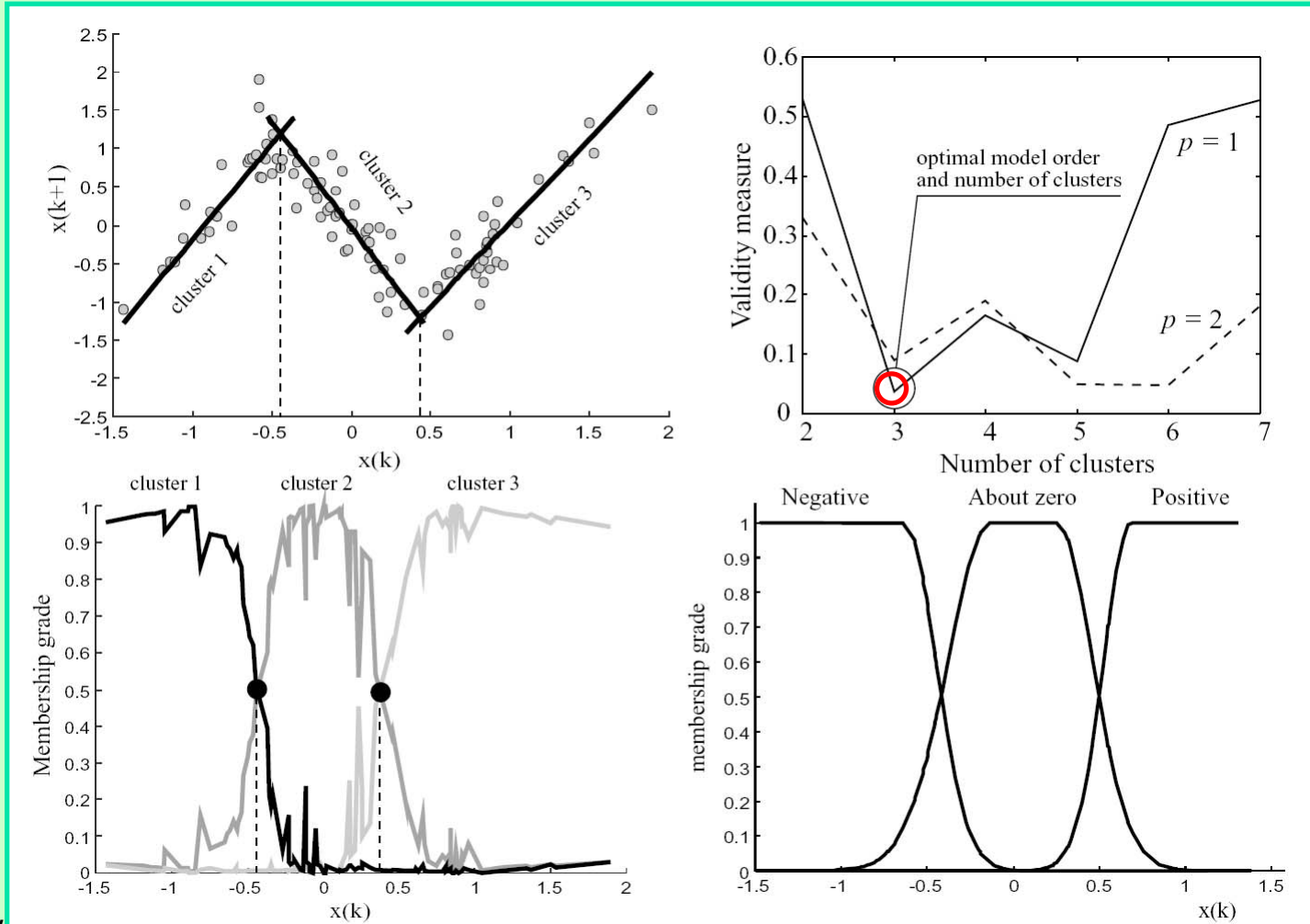
1. Choose model order p

$$x(k+1) = f(\underbrace{x(k), x(k-1), \dots, x(k-p+1)}_{\mathbf{x}(k)})$$

2. Form pattern matrix \mathbf{Z} to be clustered

$$\mathbf{Z}^T = \begin{bmatrix} x(1) & x(2) & \dots & x(p) & x(p+1) \\ x(2) & x(3) & \dots & x(p+1) & x(p+2) \\ \vdots & \vdots & & \vdots & \vdots \\ x(N-p) & x(N-p+1) & \dots & x(N-1) & x(N) \end{bmatrix}$$

Clustering Results






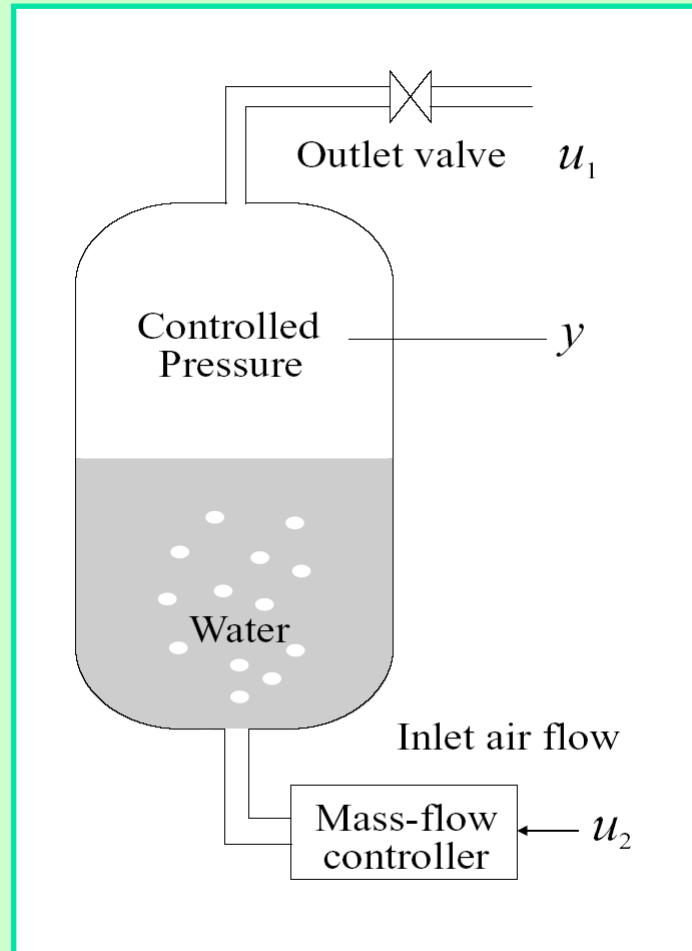
Rules Obtained

- 1) If $x(k)$ is *Positive* then $x(k+1) = 2.0244x(k) - 2.0289$
- 2) If $x(k)$ is *About zero* then $x(k+1) = -1.8852x(k) + 0.0005$
- 3) If $x(k)$ is *Negative* then $x(k+1) = 1.9050x(k) + 1.9399$

original function:
$$f(x) = \begin{cases} 2x - 2, & 0.5 < x \\ -2x, & -0.5 \leq x < 0.5 \\ 2x + 2, & x \leq -0.5 \end{cases}$$

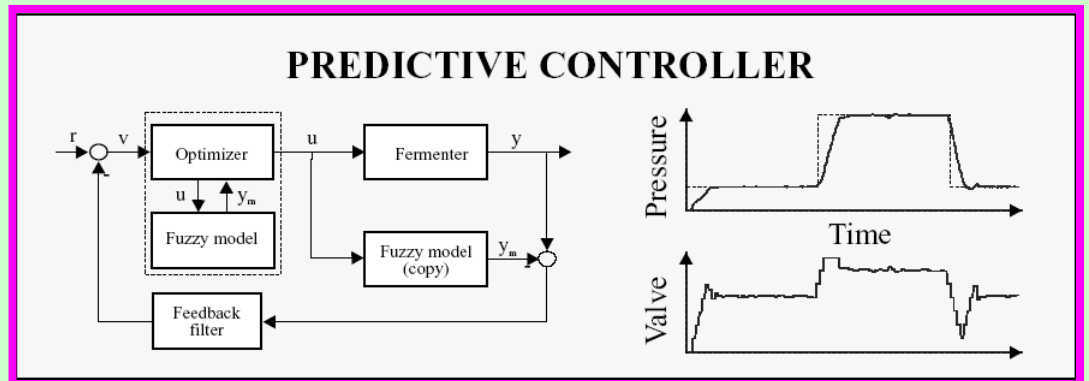
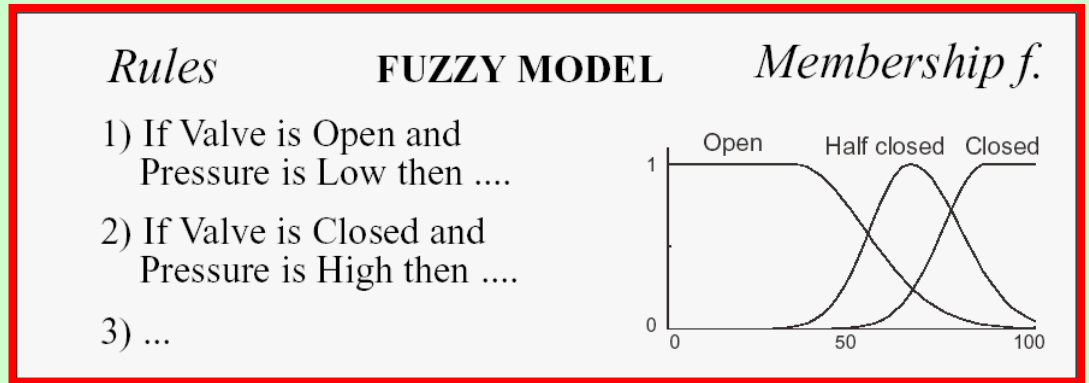
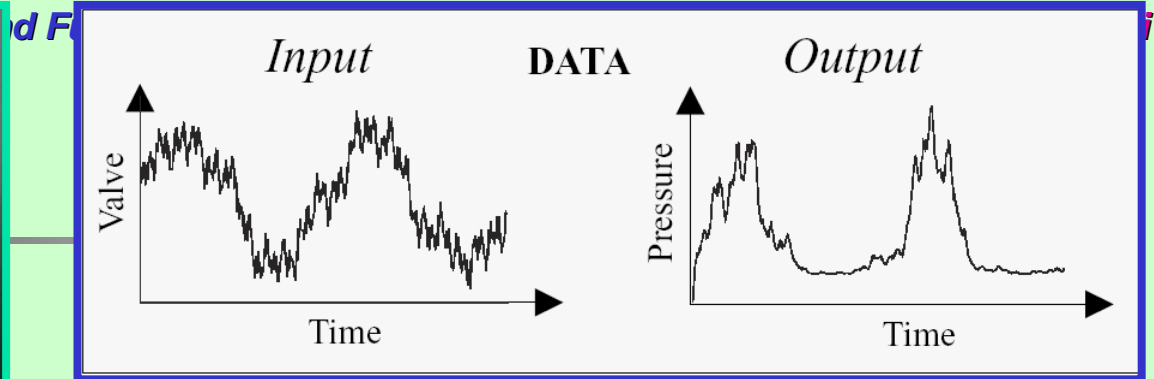


Identification of Pressure Dynamics





14/04/2009



123/148



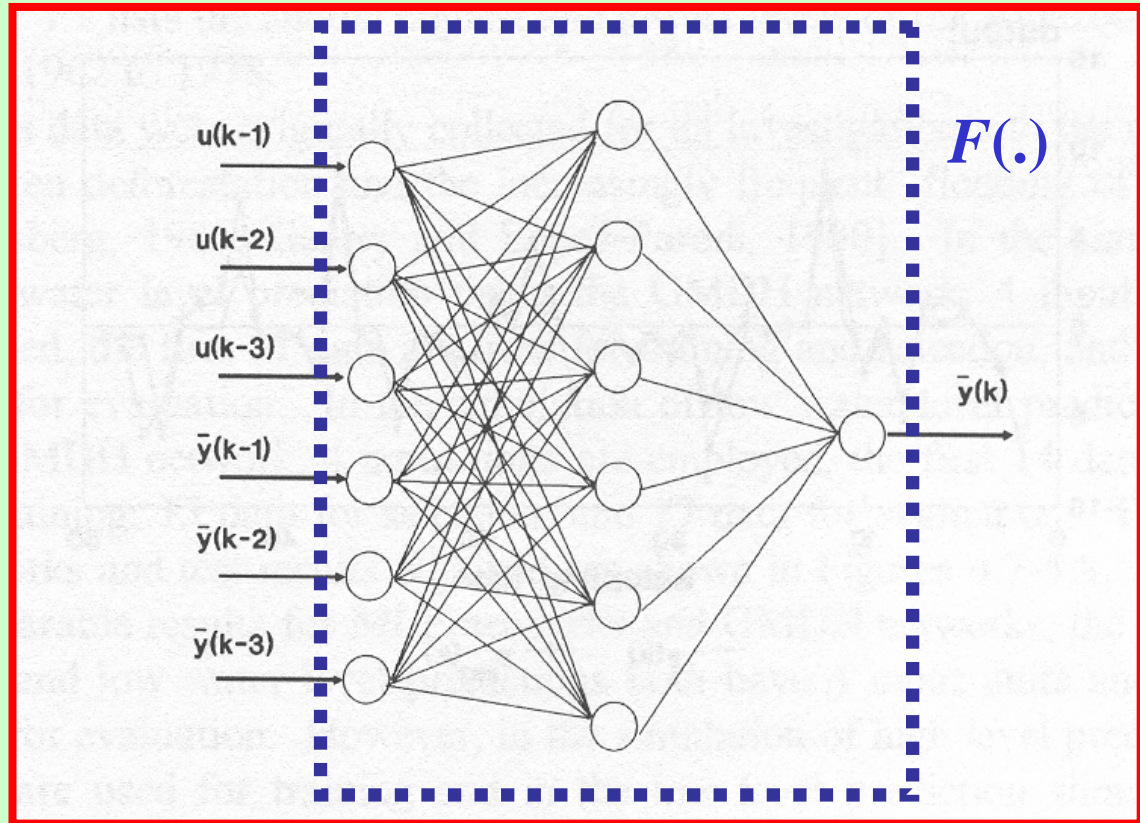
Application Examples

Neural Networks for

Non-linear Identification, Prediction and Control

Nonlinear *Dynamic* System

- Take a static NN
- From static to dynamic NN
- "Quasi-static" NN
- Add inputs, outputs and delayed signals



$$\tilde{y}(k) = F(u(k-1), u(k-2), u(k-3), \tilde{y}(k-1), \tilde{y}(k-2), \tilde{y}(k-3))$$

**Example of Quasi-static NN
with 3 delayed inputs and outputs**

Nonlinear System Identification

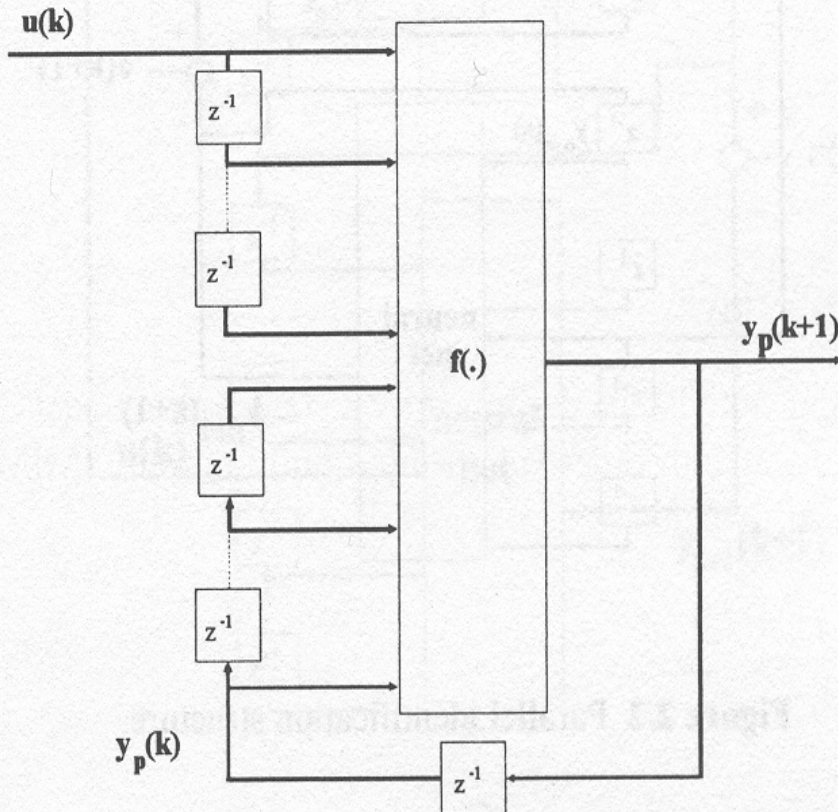


Figure 2.1 Input-output model

- $f(\cdot)$, unknown target function
- Nonlinear dynamic model
- Approximated via a quasi-static NN
- Nonlinear dynamic system identification
- Recall “*linear system identification*”

Nonlinear System Identification

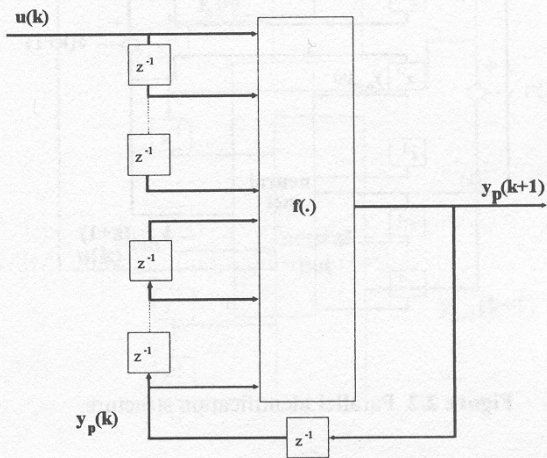
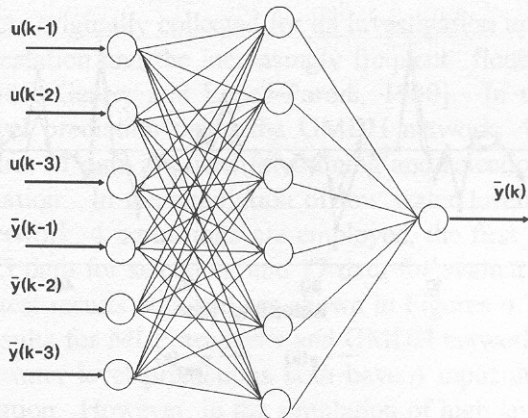
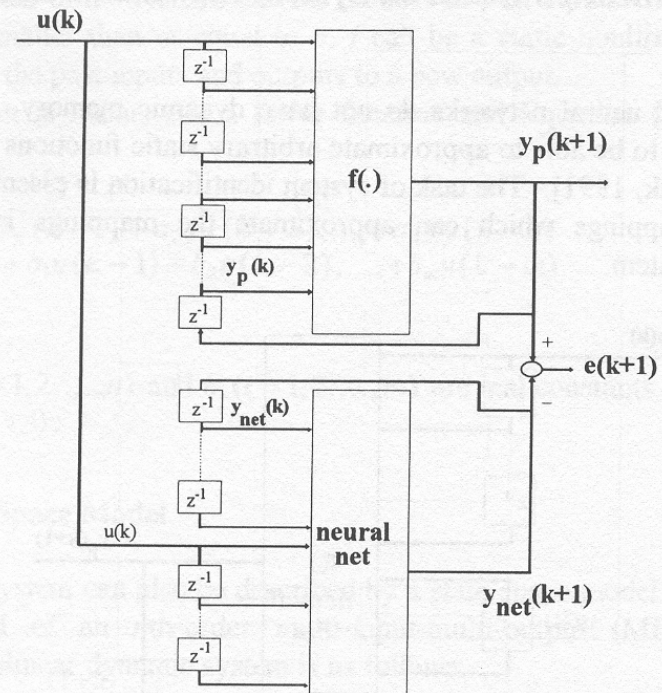
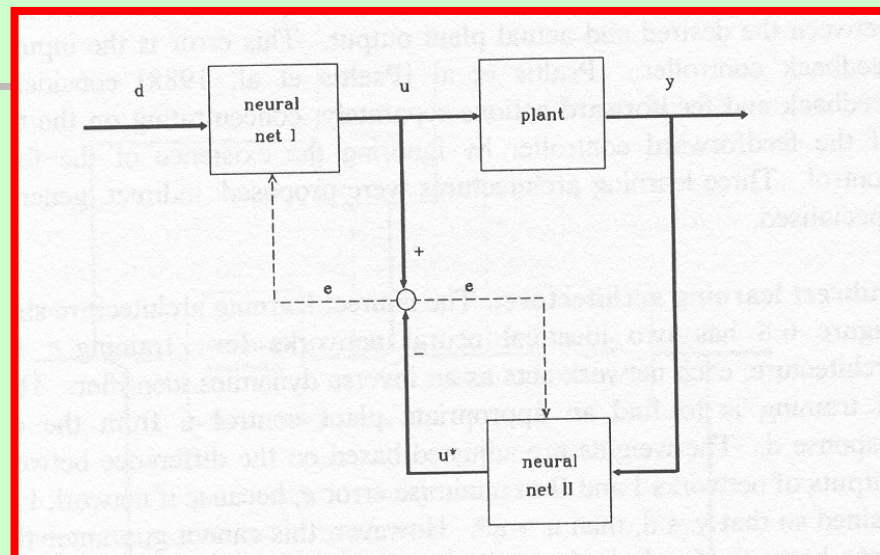
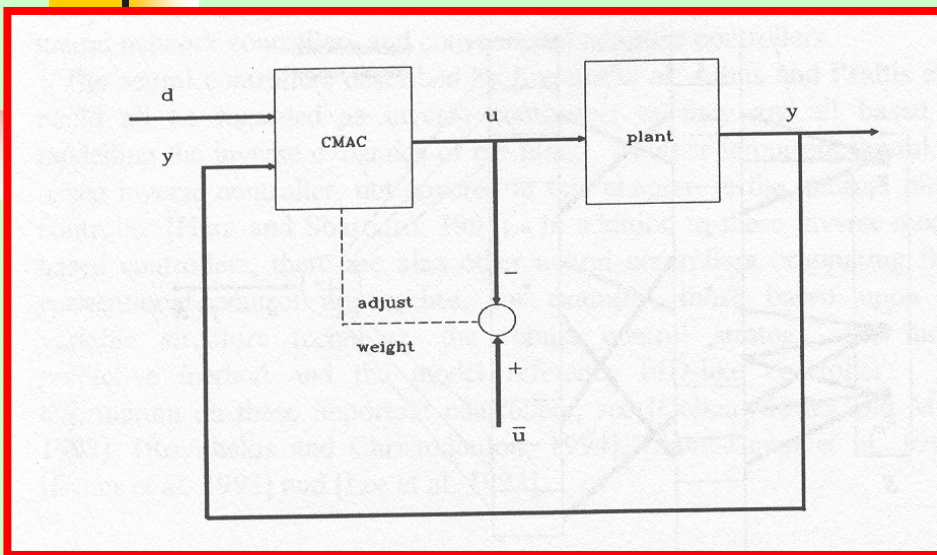


Figure 2.1 Input-output model



Target function: $y_p(k+1) = f(.)$
Identified function: $y_{NET}(k+1) = F(.)$
Estimation error: $e(k+1)$

Nonlinear System Neural Control



d : reference/desired response
 y : system output/desired output
 u : system input/controller output
 \bar{u} : desired controller input
 u^* : NN output
 e : controller/network error

The goal of training is to find an appropriate plant control u from the desired response d . The weights are adjusted based on the difference between the outputs of the networks I & II to minimise e . If network I is trained so that $y = d$, then $u = u^*$. Networks act as inverse dynamics identifiers.

Neural Networks for Control

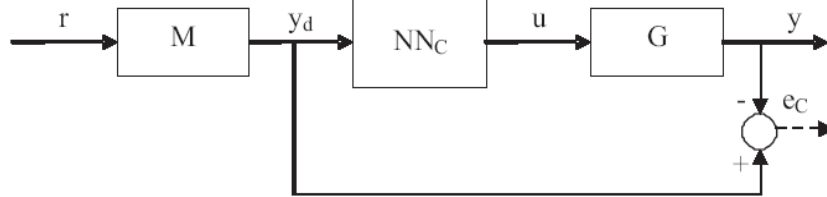


Figure 1: Direct Inverse Control using neural networks

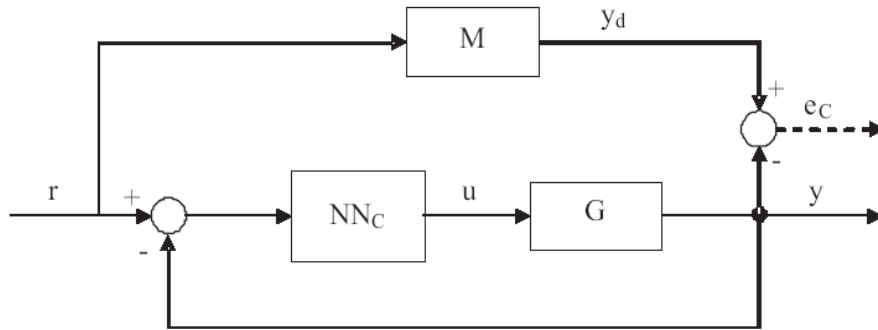


Figure 2: Model Reference Control using neural networks

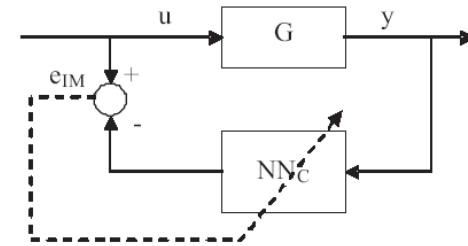
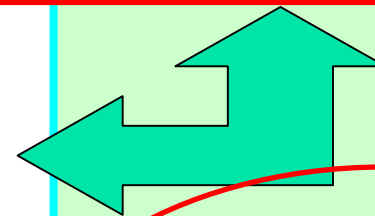


Figure 3: Training the neural network NN_C



Figures 1 and 3 Problems.

- Open-loop unstable models
- Disturbances

Neural Model Reference Adaptive Control

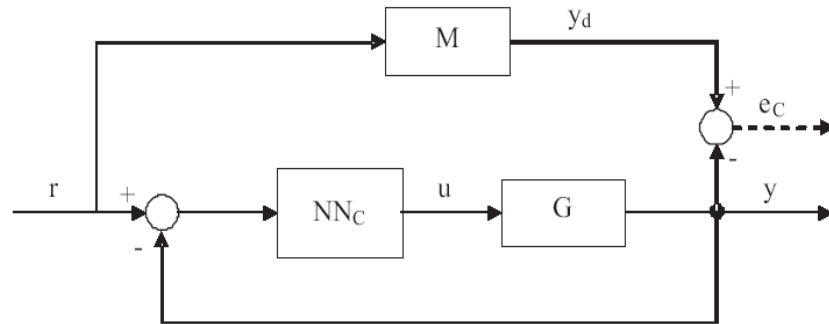


Figure 2: Model Reference Control using neural networks

The signal e_c is used to train or adapt online the weights of the controller NN_C . Two are the approaches used to design a MRAC control for an unknown plant: **Direct and Indirect Control**.

Direct Control: This procedure aims at designing a controller without having a plant model. As the knowledge of the plant is needed in order to train the neural network which corresponds to the controller (*i.e.* NN_C), until present, no method has been proposed to deal with this problem.

Neural Model Reference Adaptive Control

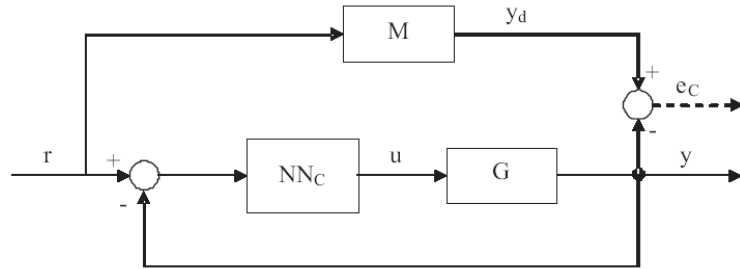


Figure 2: Model Reference Control using neural networks

- The signal e_c is used to train or adapt online the weights of the **neural controller** NN_C .

Indirect Control

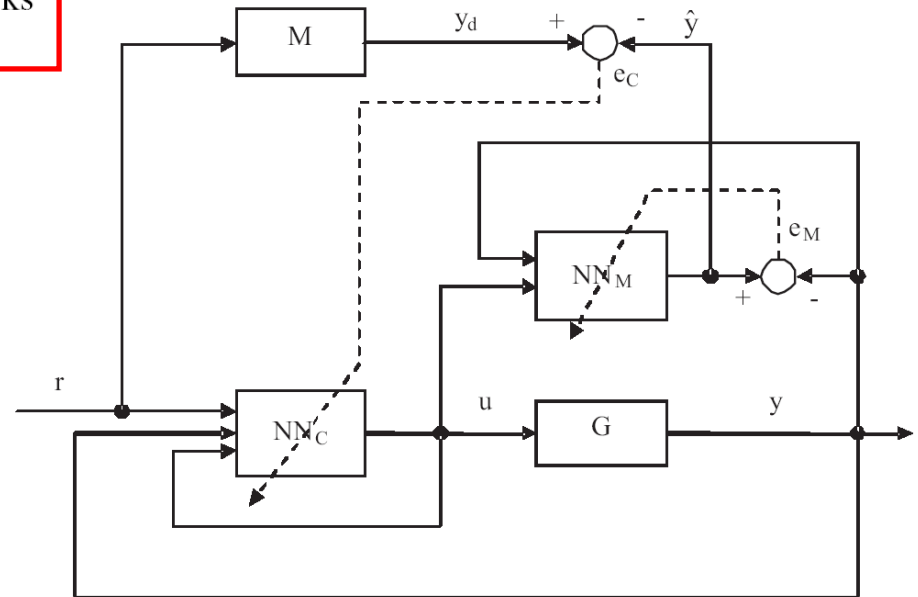


Figure 4: Indirect MRAC

Indirect Control: NN_M & NN_C

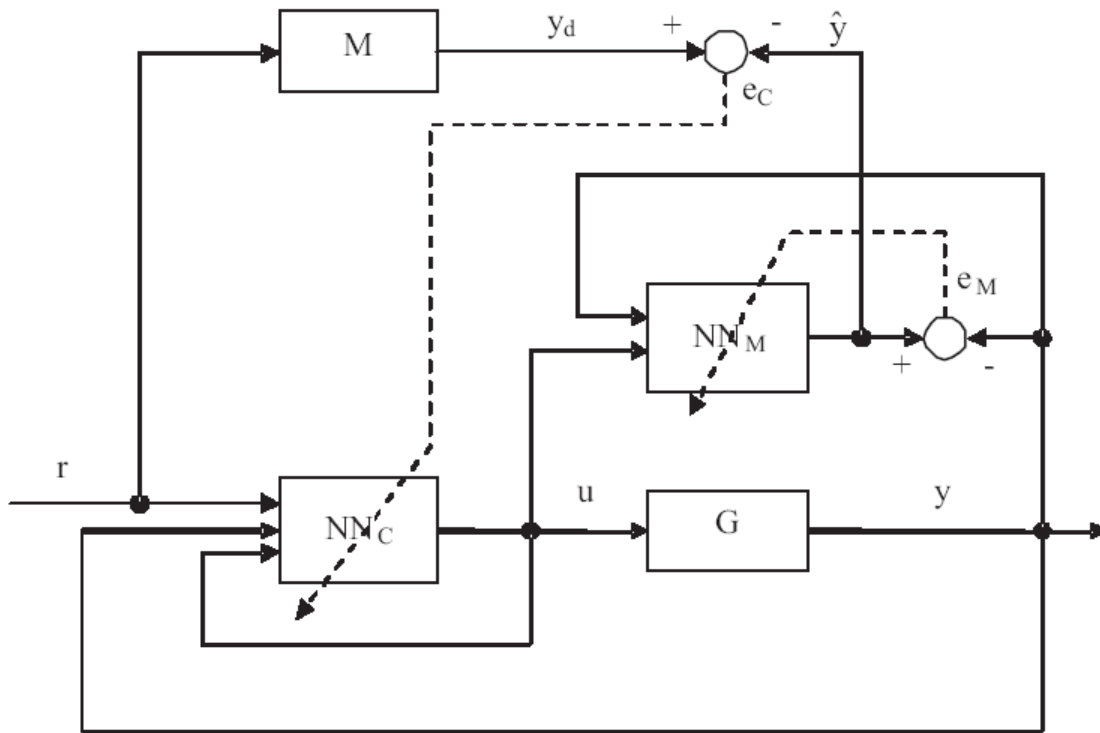


Figure 4: Indirect MRAC

This approach uses two neural networks: one for modelling the plant dynamics (NN_M), and another one trained to control the **real plant (G)** so as its behaviour is as close as possible to the **reference model (M)** via the neural controller (NN_C).

Indirect Control (1)

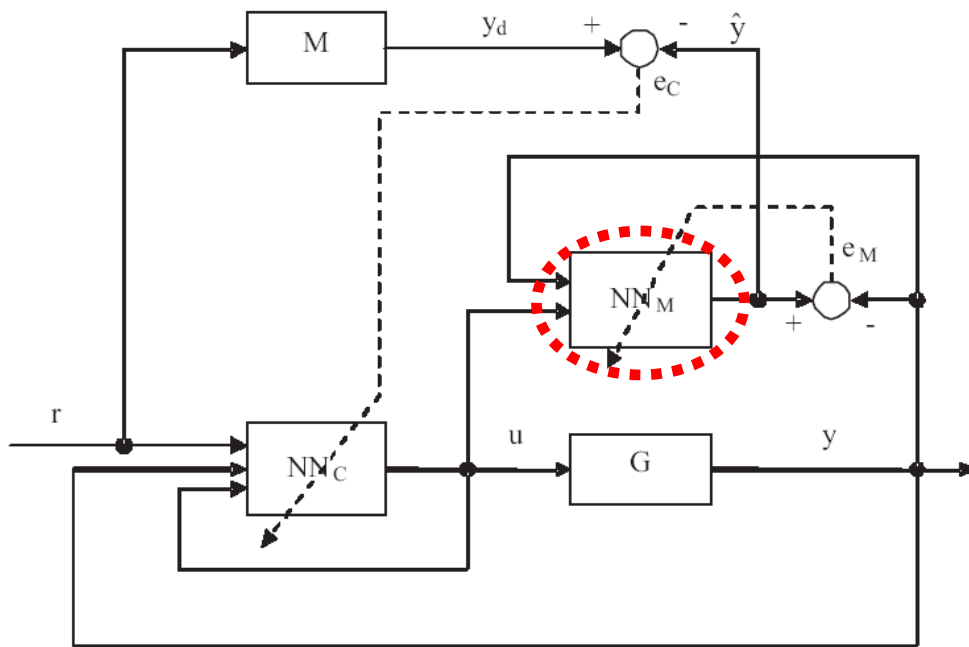


Figure 4: Indirect MRAC

The **neural network NN_M** is trained to approximate the plant **G** input/output relation using the signal e_M . This is usually done offline, using a batch of data gathered from **the plant in open loop**.

Indirect Control (2)

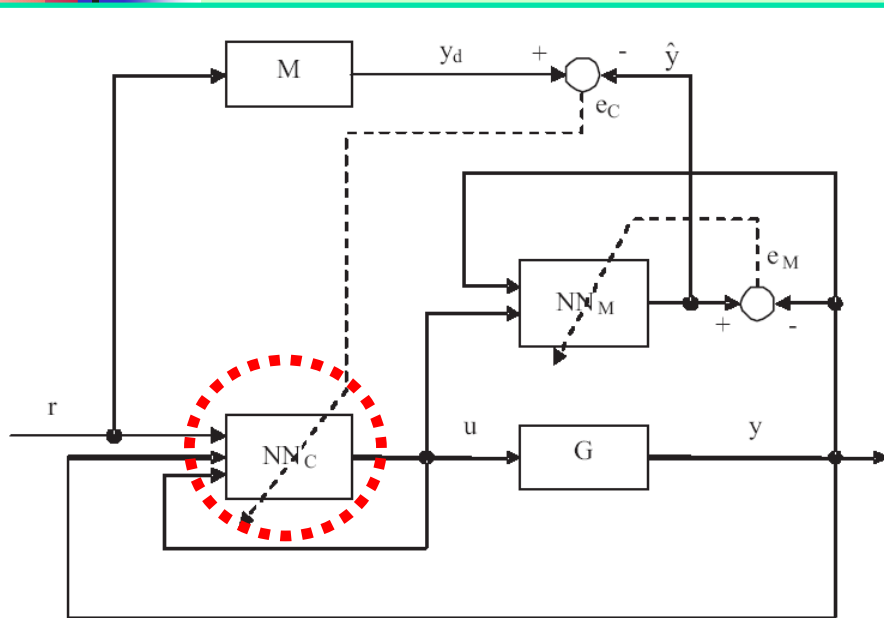


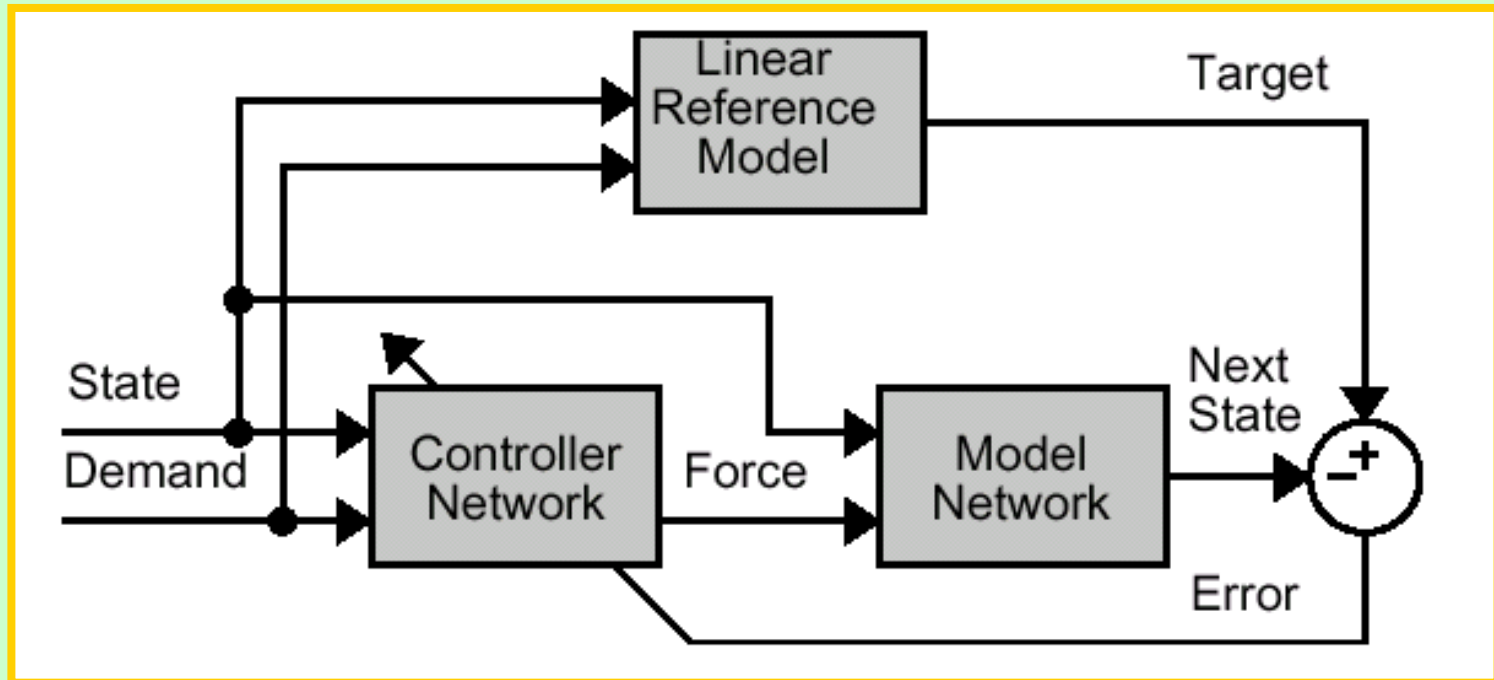
Figure 4: Indirect MRAC

Then, NN_M is fixed, its output and behaviour are known and easy to compute.

Once the model NN_M is trained, it is used to train the network NN_C which will act as the controller. The model NN_M is used instead of the real plant's output because the real plant is unknown, so back-propagation algorithms can not be used. In this way, the control error e_C is calculated as the difference between the desired reference model output y_d and \hat{y} , which is the closed loop predicted output.

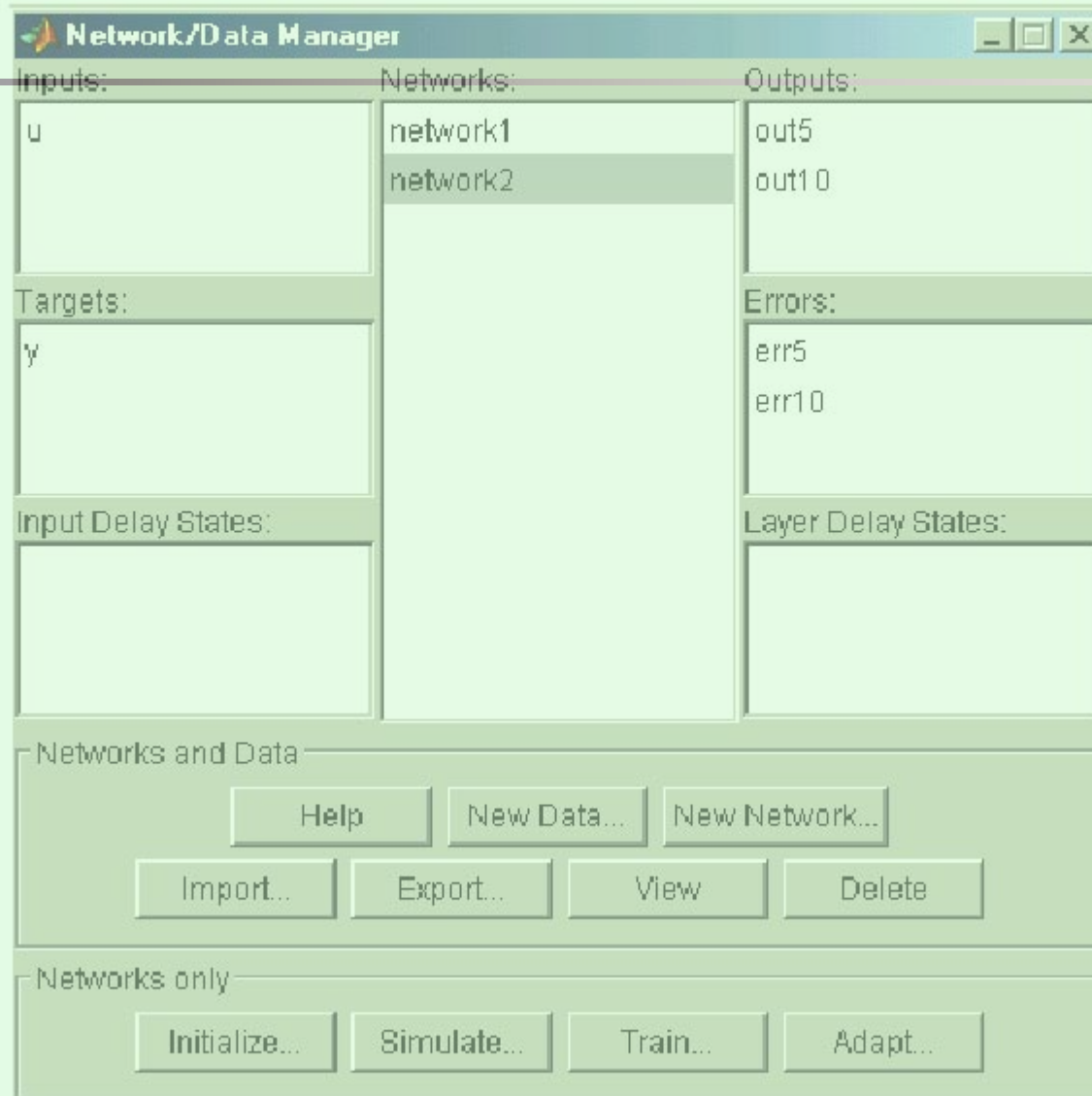
Model Reference Control

Matlab and Simulink solution

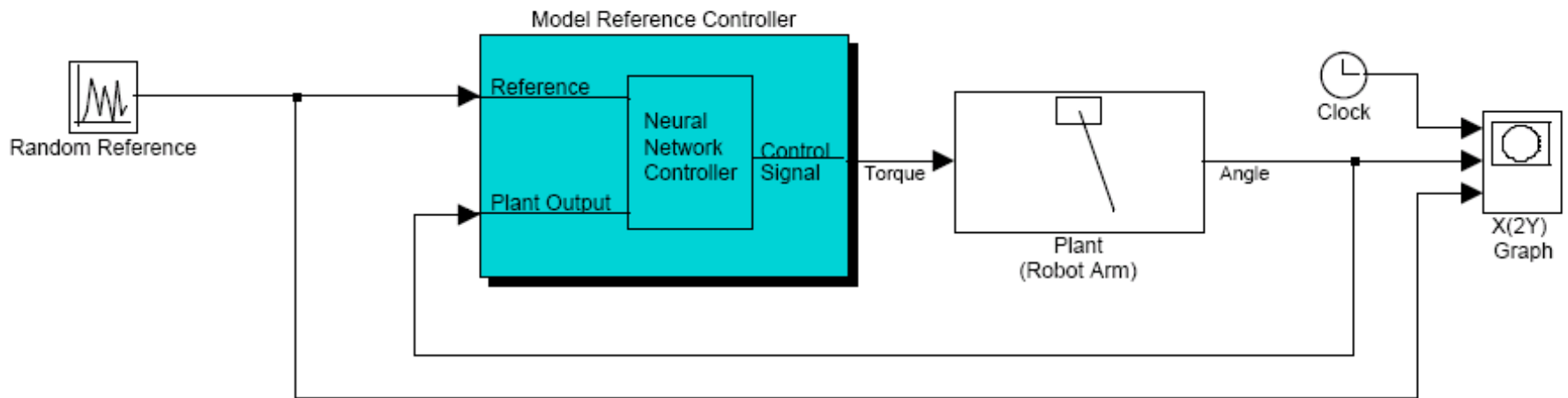


Neural controller, reference model, neural model

Matlab NNtool GUI (Graphical User Interface)



Control of a Robot Arm Example



Neural Network Model Reference Control of a Robot Arm
 (Double click on the "?" for more info)



Double click
 here for
 Simulink Help

To start and stop the simulation, use the "Start/Stop"
 selection in the "Simulation" pull-down menu

Control of a Robot Arm Example

Model Reference Control

File Window Help

Model Reference Control

Network Architecture

Size of Hidden Layer: No. Delayed Reference Inputs:

Sampling Interval (sec): No. Delayed Controller Outputs:

Normalize Training Data No. Delayed Plant Outputs:

Training Data

Maximum Reference Value: Controller Training Samples:

Minimum Reference Value:

Maximum Interval Value (sec): Reference Model:

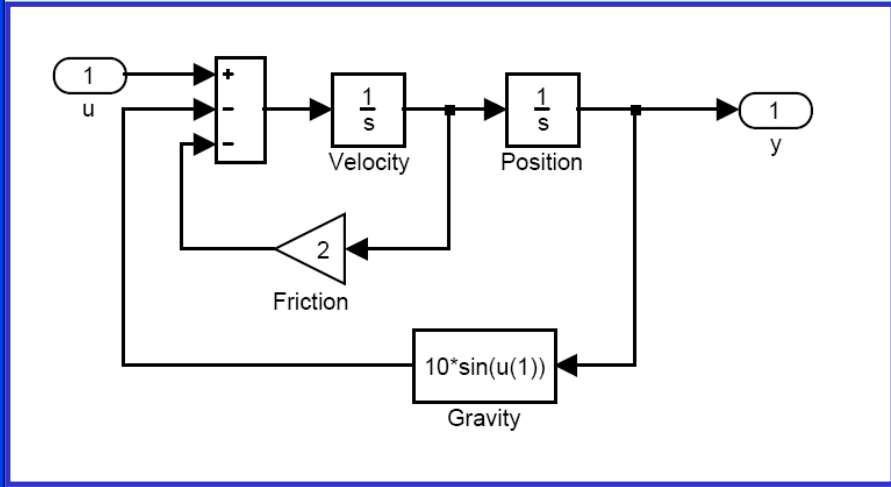
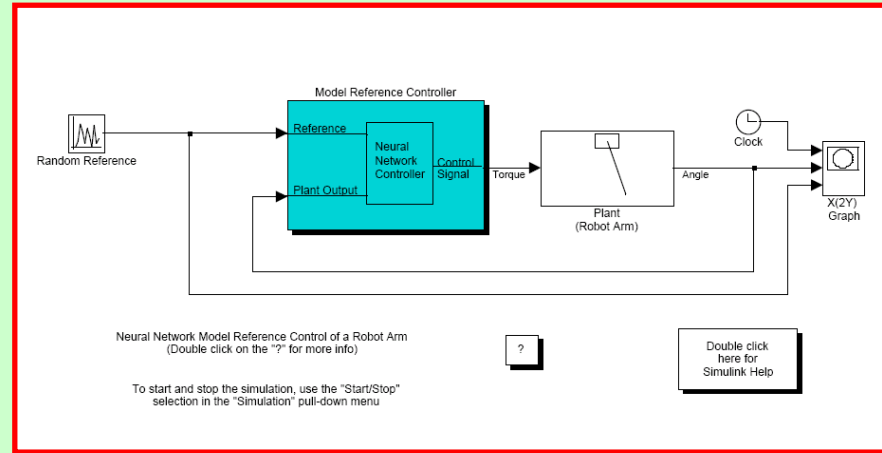
Minimum Interval Value (sec):

Training Parameters

Controller Training Epochs: Controller Training Segments:

Use Current Weights Use Cumulative Training

Perform plant identification before controller training.



Control of a Robot Arm Example

Plant Identification

File Window Help

Plant Identification

Network Architecture

Size of Hidden Layer: No. Delayed Plant Inputs:

Sampling Interval (sec): No. Delayed Plant Outputs:

Normalize Training Data

Training Data

Training Samples: Limit Output Data

Maximum Plant Input: Maximum Plant Output:

Minimum Plant Input: Minimum Plant Output:

Maximum Interval Value (sec): Simulink Plant Model:

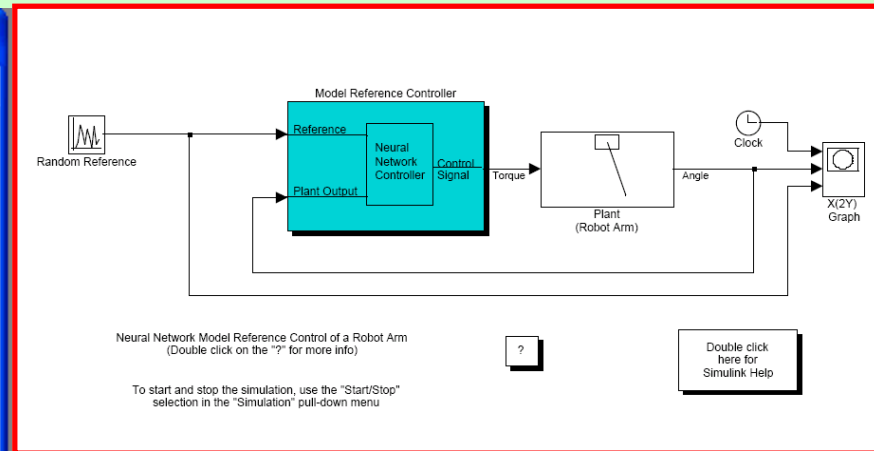
Minimum Interval Value (sec):

Training Parameters

Training Epochs: Training Function:

Use Current Weights Use Validation Data Use Testing Data

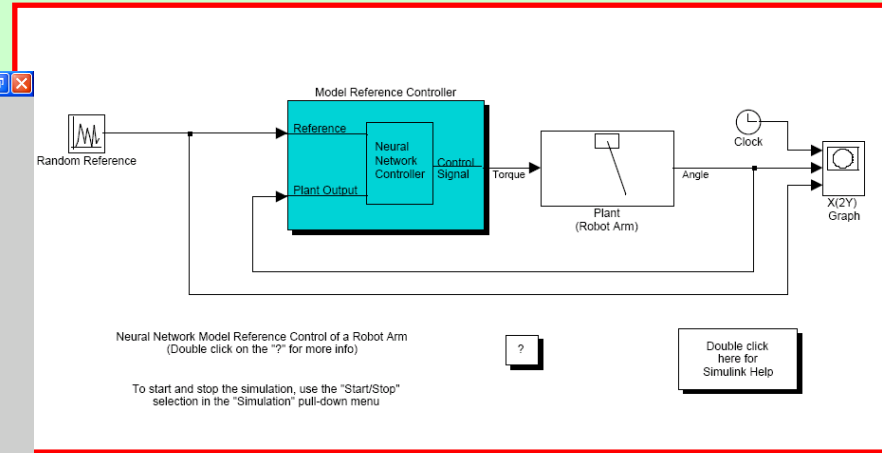
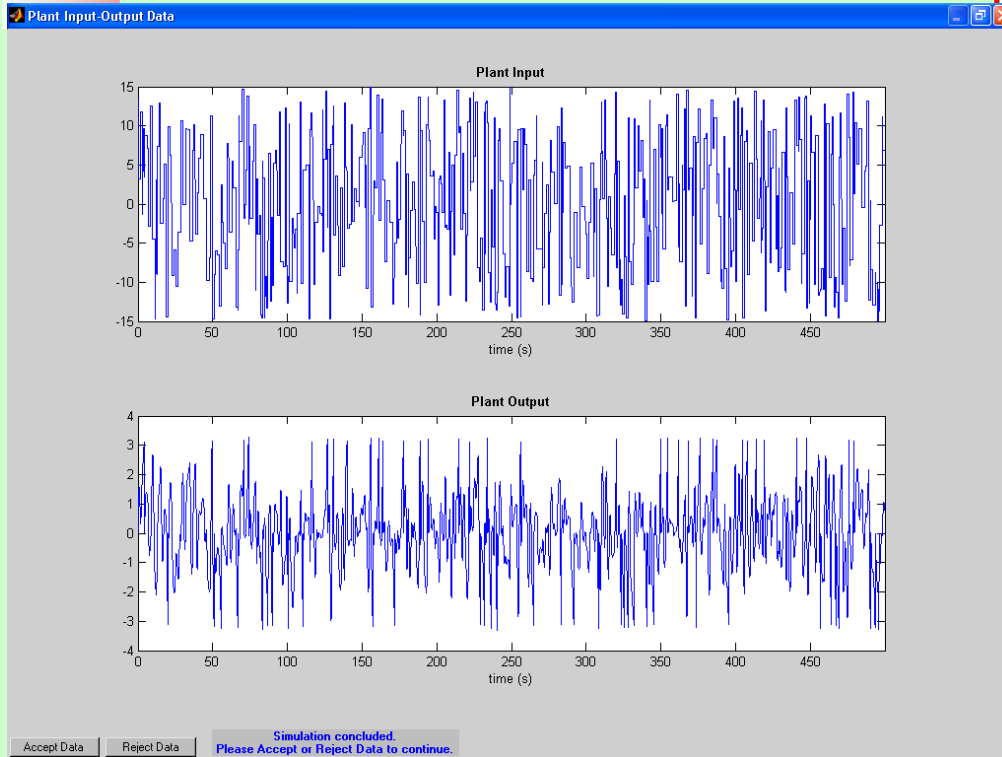
Generate or import data before training the neural network plant.



Plant Identification:

Data generation from the Reference Model for Neural Network training

Control of a Robot Arm Example



After Plant Identification:

Neural Network training

Control of a Robot Arm Example

Plant Identification

File Window Help

Plant Identification

Network Architecture

Size of Hidden Layer: 10
 No. Delayed Plant Inputs: 2
 Sampling Interval (sec): 0.05
 No. Delayed Plant Outputs: 2

Normalize Training Data

Training Data

Training Samples: 10000
 Maximum Plant Input: 15
 Minimum Plant Input: -15
 Maximum Interval Value (sec): 2
 Minimum Interval Value (sec): 0.1

Limit Output Data
 Maximum Plant Output: 3.1
 Minimum Plant Output: -3.1
 Simulink Plant Model: robotarm

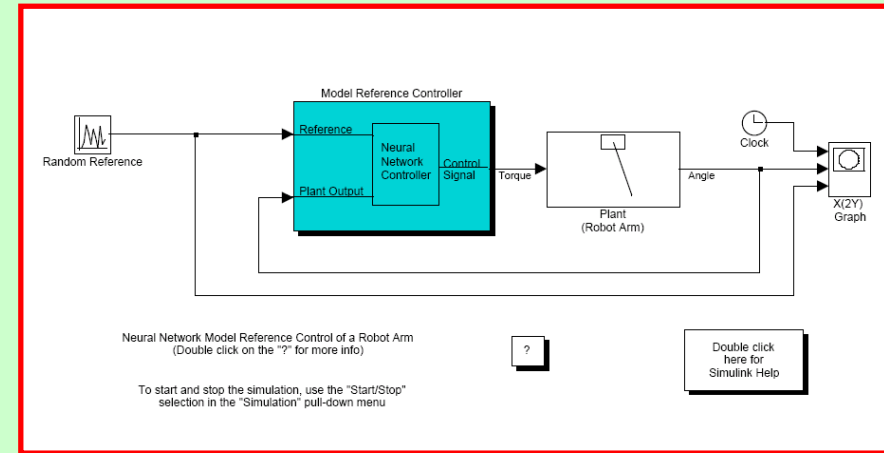
Training Parameters

Training Epochs: 300
 Training Function: trainlm

Use Current Weights
 Use Validation Data
 Use Testing Data

Buttons: Erase Generated Data, Import Data, Export Data, Train Network, OK, Cancel, Apply

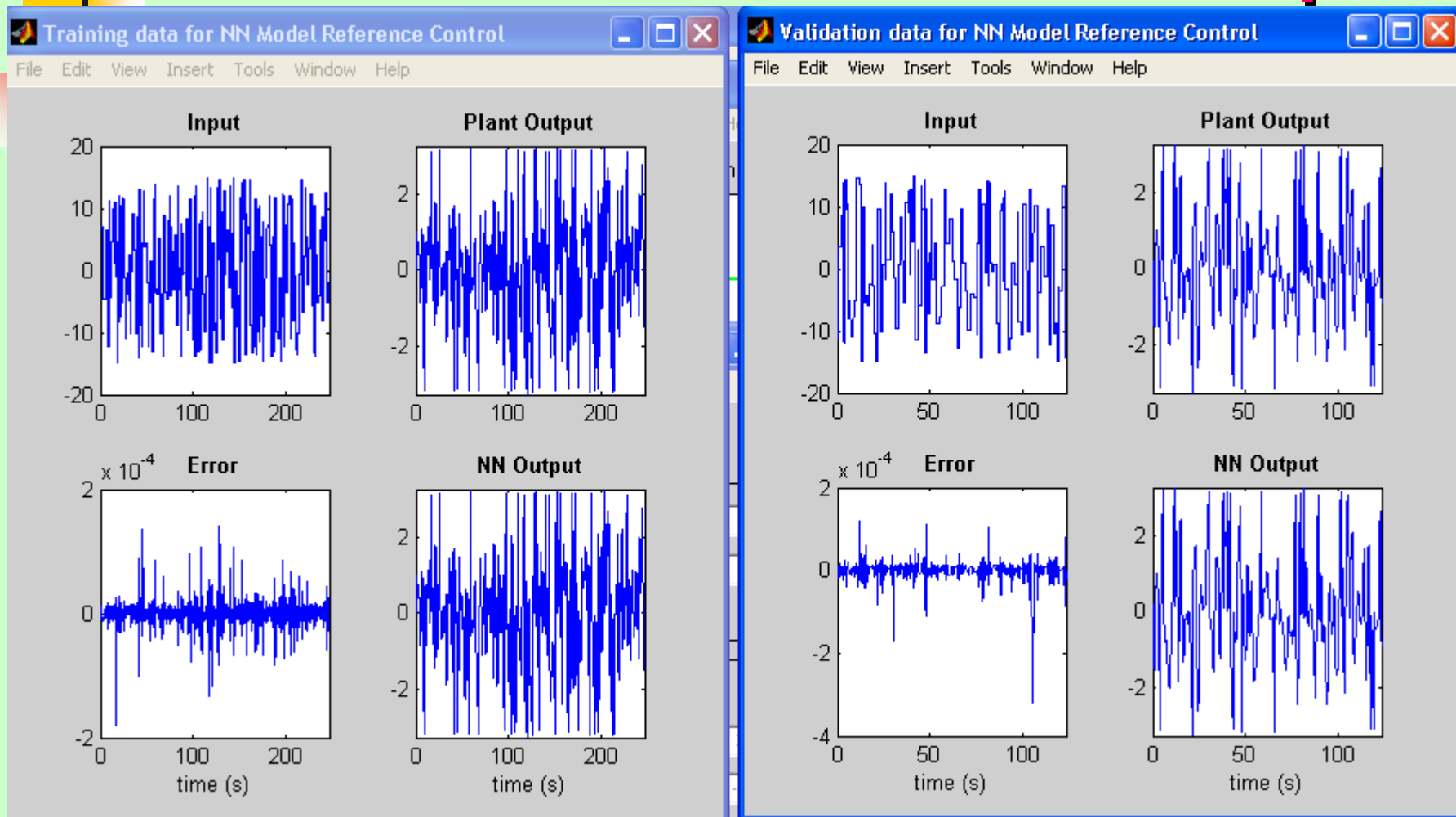
Your training data set has 10000 samples.
 You can now train the network.



After Plant Identification:

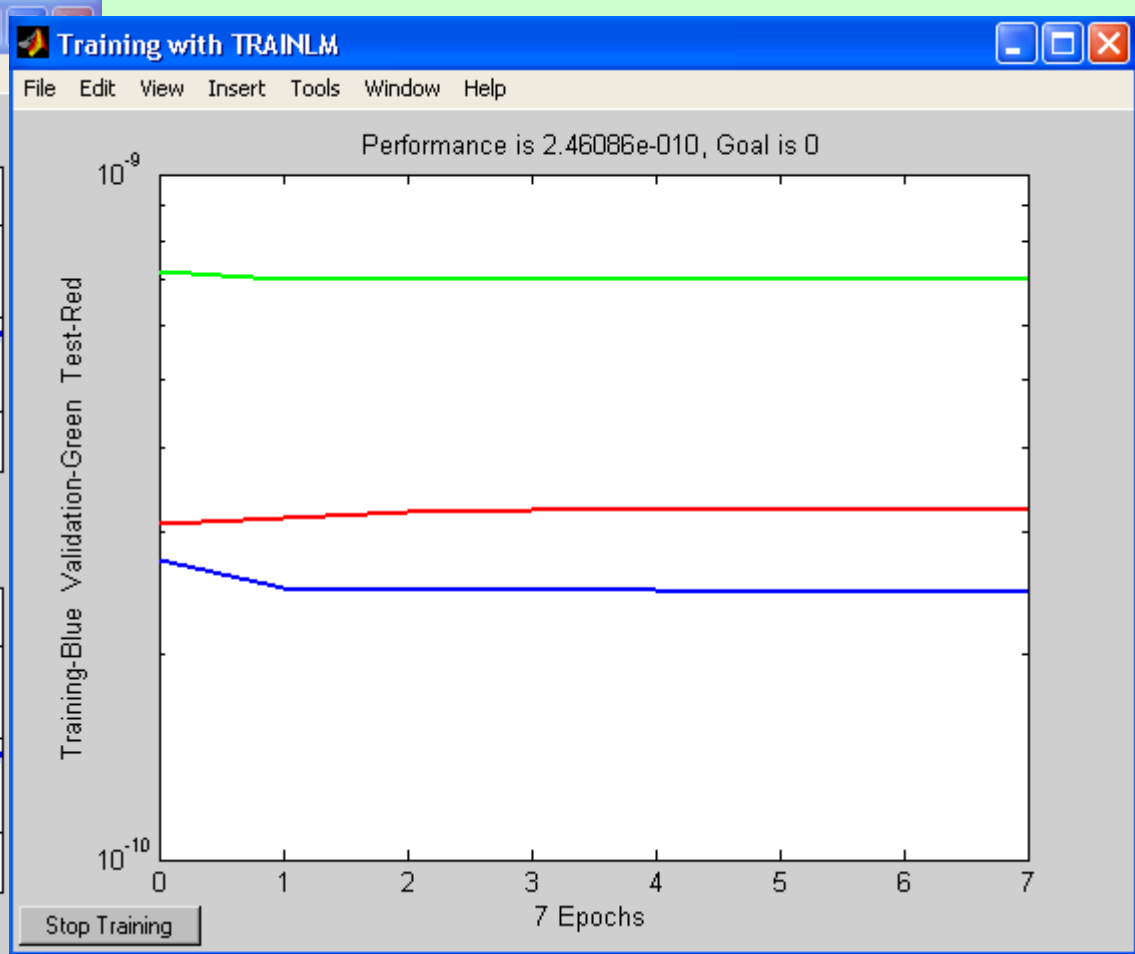
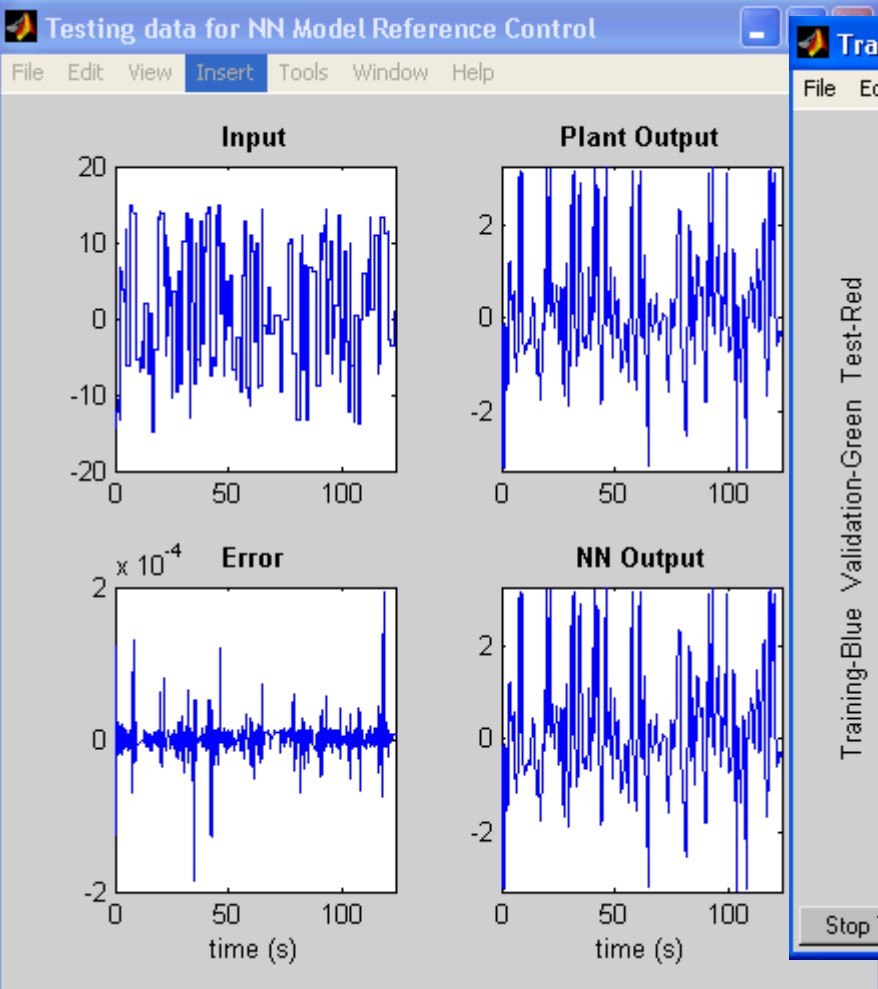
Neural Network training

Control of a Robot Arm Example



Training and Validation Data

Control of a Robot Arm Example



Testing Data and Training Results

Control of a Robot Arm Example

Model Reference Control

File Window Help

Model Reference Control

Network Architecture

Size of Hidden Layer: No. Delayed Reference Inputs:

Sampling Interval (sec): No. Delayed Controller Outputs:

Normalize Training Data No. Delayed Plant Outputs:

Training Data

Maximum Reference Value: Controller Training Samples:

Minimum Reference Value: Defines how many data points will be generated

Maximum Interval Value (sec): Reference Model:

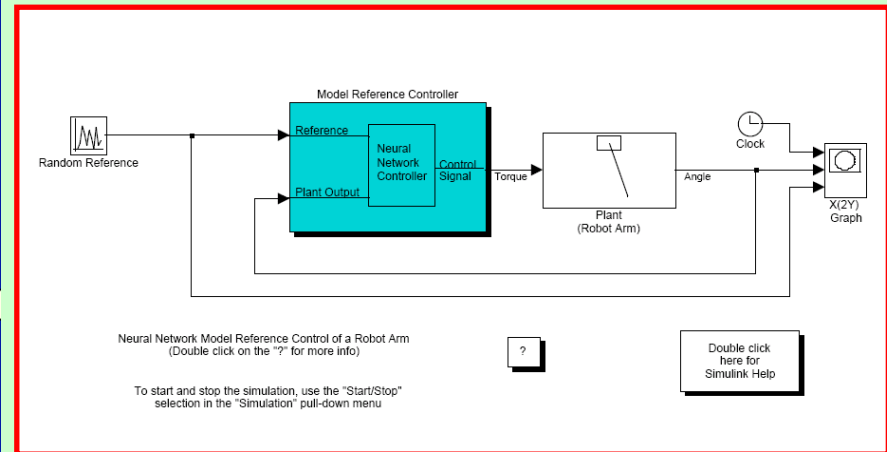
Minimum Interval Value (sec):

Training Parameters

Controller Training Epochs: Controller Training Segments:

Use Current Weights Use Cumulative Training

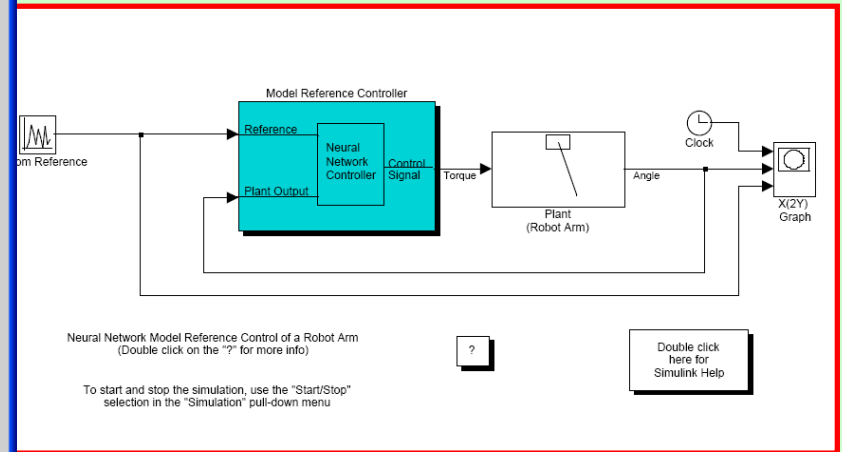
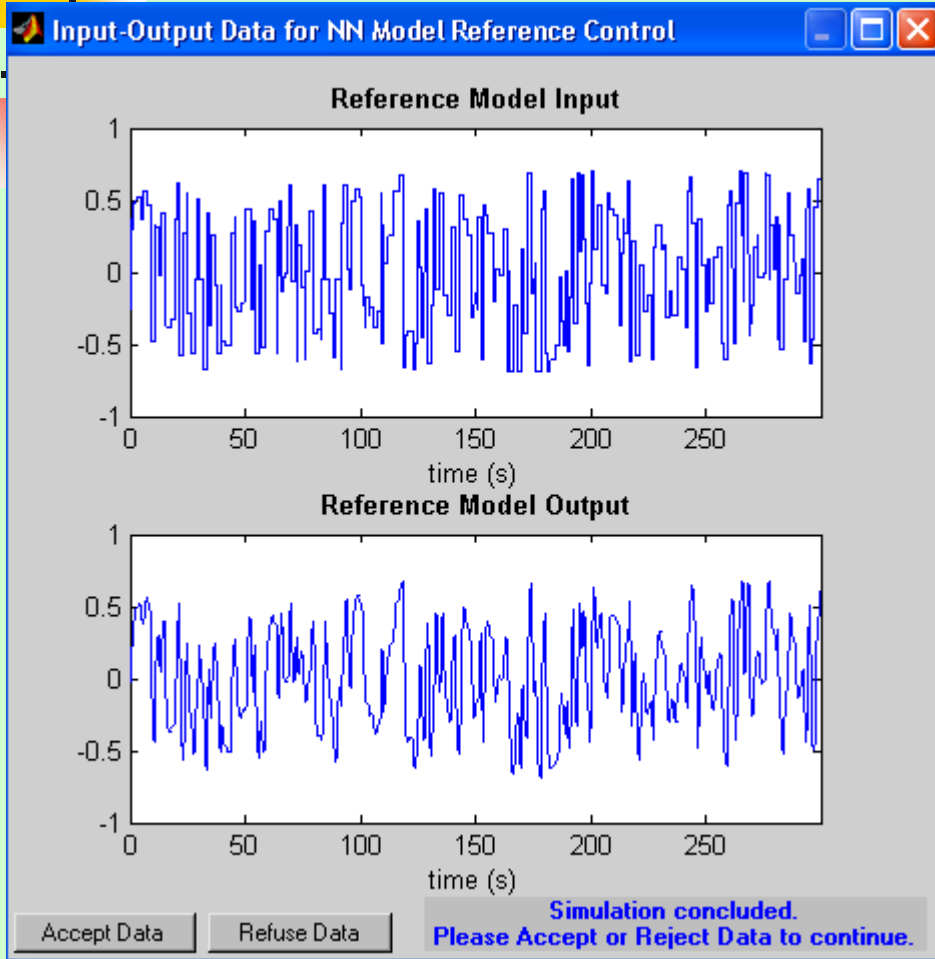
Generate or import data before training the neural network controller.



Plant identification with a NN

Data Generation for NN Controller Identification

Control of a Robot Arm Example



Accept the Data Generated for NN Controller Identification

Control of a Robot Arm Example

Model Reference Control

File Window Help

Model Reference Control

Network Architecture

Size of Hidden Layer: 13 No. Delayed Reference Inputs: 2

Sampling Interval (sec): 0.05 No. Delayed Controller Outputs: 1

Normalize Training Data No. Delayed Plant Outputs: 2

Training Data

Maximum Reference Value: 0.7 Controller Training Samples: 6000

Minimum Reference Value: -0.7

Maximum Interval Value (sec): 2 Reference Model: Browse

Minimum Interval Value (sec): 0.1 robotref

Erase Generated Data Import Data Export Data

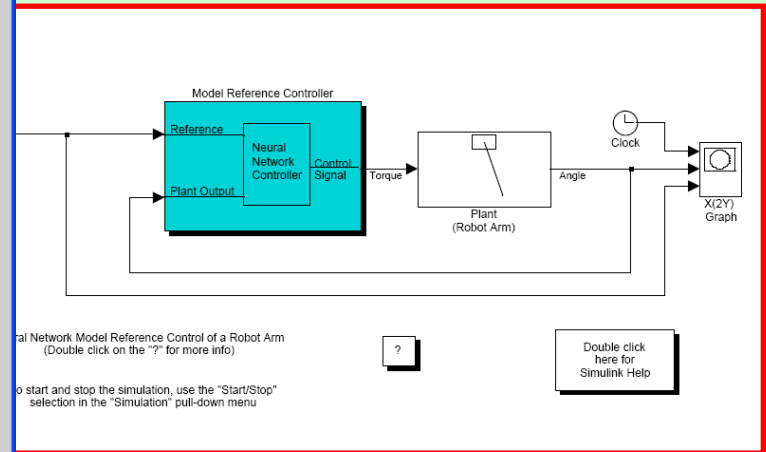
Training Parameters

Controller Training Epochs: 10 Controller Training Segments: 30

Use Current Weights Use Cumulative Training

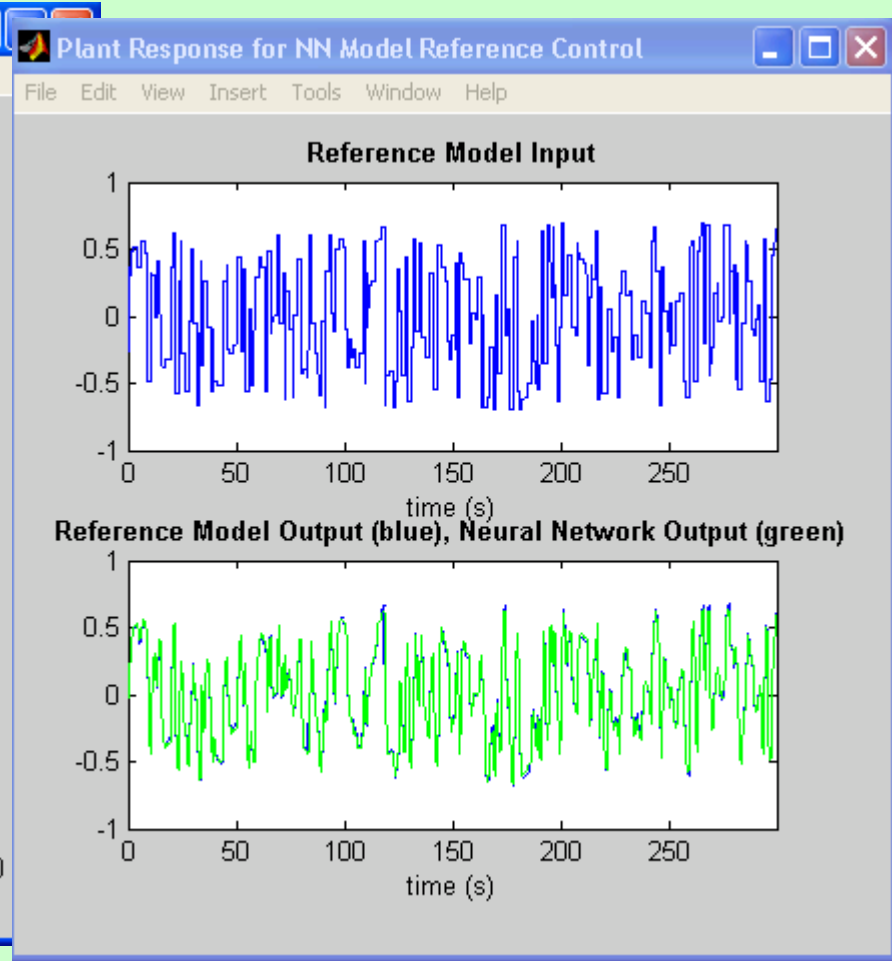
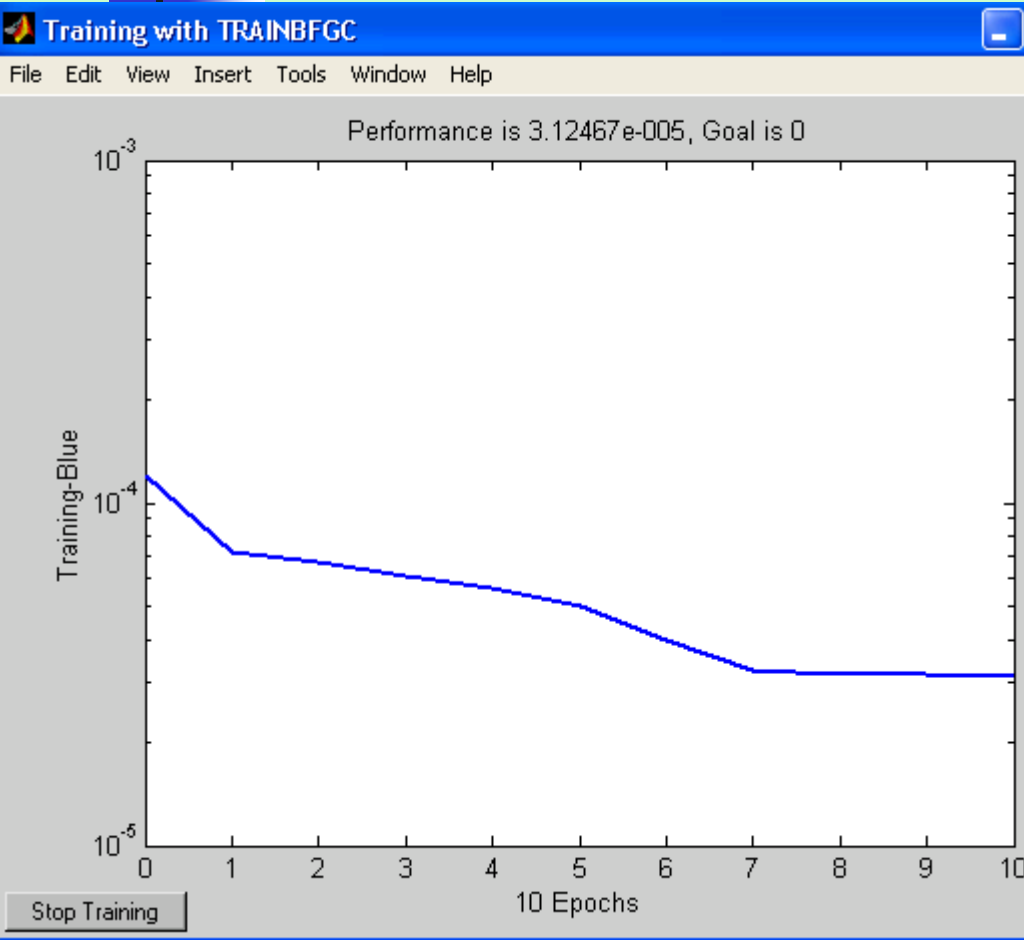
Plant Identification Train Controller OK Cancel Apply

Your training data set has 6000 samples.
You can now train the network.



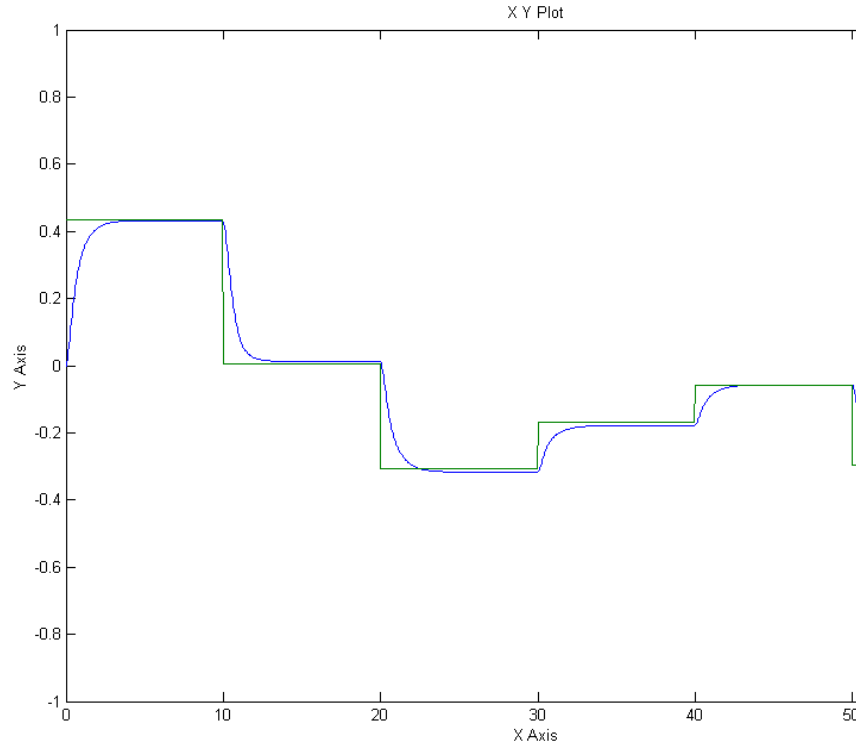
NN Controller Training

Control of a Robot Arm Example



NN Controller Training and Results

Control of a Robot Arm Example



Reference and Tracked Output Signals

Simulation Final Results

