Soft Computing in Fault Detection and Isolation

PART II

Artificial neural networks in fault diagnosis

Marcin Witczak

8 października 2012

OUTLINE

- ➥ Introduction to Artificial Neural Networks (ANNs) theory
- ➥ ANNs in fault diagnosis
- ➥ Modeling of system dynamics via ANNs
- ➥ Robust fault detection approaches
- ➥ ANNs-based symptom evaluation

☛ INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS THEORY

➜ Outline of ANNs history:

1943: McCulloch-Pitts model of neuron

1949: Hebb designed the first learning law for ANNs

1950-1960: First golden age for ANNs

1959: Rosenblatt developed ANNs called *perceptrons*

1960: Widrow *ADALINE* (ADAptive Linear Neuron) and *MADALINES* – multi-layer extensions of *ADALINE*

1968: Minsky and Papert wrote the book *Perceptrons* showing limitations of perceptron models

1970s: quiet years

1972-82: associative memory neural nets; self-organizing feature maps

1985: Carpenter and Grossberg: self-organizing neural networks called *adaptive resonance theory*, ART1 and ART2

1980s: renewed enthusiasm

1982: Hopfield nets: a simple and effective NN model which stimulated interest in ANNs

1985: Rumelhart, McClelland, Hinton *et al.* wrote *Parallel Distributed Processing, Vols. I & II*
the back-propagation algorithm was discovered (or rather re-discovered).
This showed that multi-layer networks could overcome limitations discussed by Minsky and Papert

...: further improvements of the existing neural networks

➜ Biological neural networks

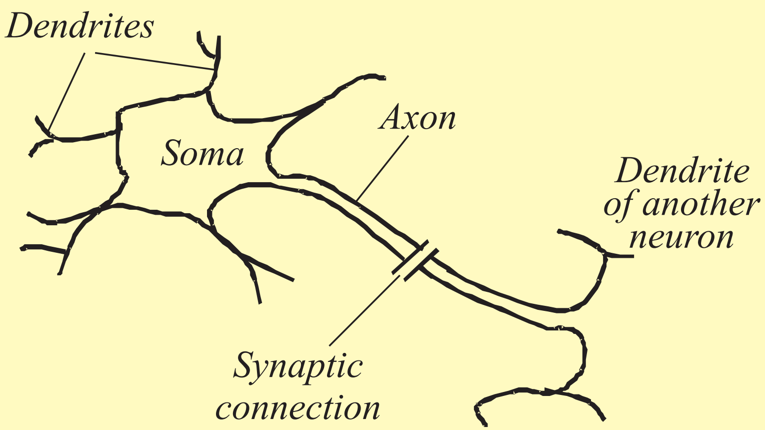
Biological neural networks have

- $\sim 10^{13}$ of neurons (very simple processors)
- $\sim 10^{17}$ of synaptic connections (storage of information)

Biological neural networks

- are extensively parallel in operation
- can complete complex computational tasks despite using very slow components (neurons)

➜ Biological neural networks



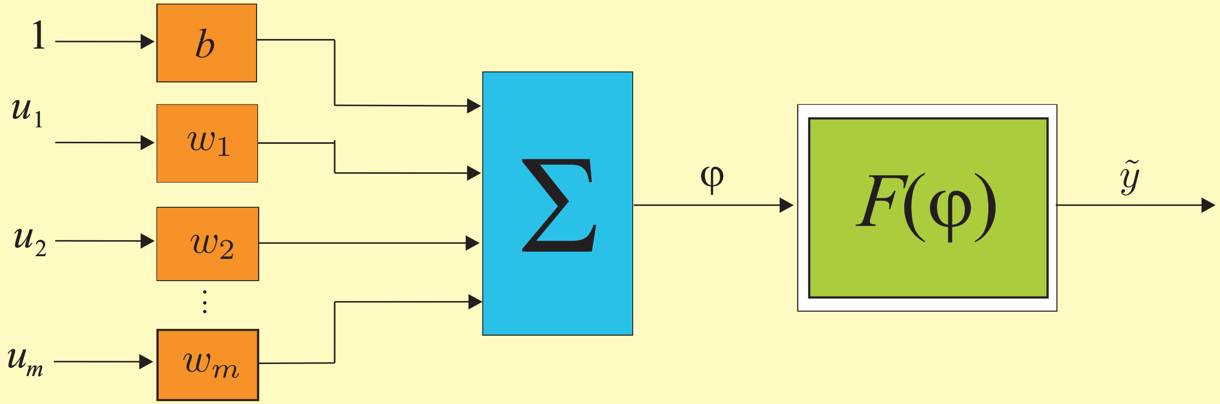
A biological neuron has three types of components:

- *dendrites* – receive signals from other neurons. The signals are electric impulses that are transmitted across a synaptic connection by means of chemical processes.
- *soma* or *cell body* – sums the incoming signals. When a sufficient input is received, the cell fires: it transmits a signal over the axon to other cells. The neuron only fires if its membrane potential $\varphi >$ the threshold.
- *axon* – the output of the neuron.

➜ Some features of ANNs that are suggested by biological neurons

- Signals may be modified by a weight at the receiving synapse
- Processing element sums the weighted inputs
- Information processing is local
- Memory is distributed:
  - long-term memory resides in neurons' synapses or weight
  - short-term memory corresponds to signals sent by neurons
- Synapse's strength may be modified by experience
- ANN is fault tolerant: if some neurons fail or if some connections between neurons are broken, the performance of the ANN is not affected

➜ General model of artificial neurons (a static case)

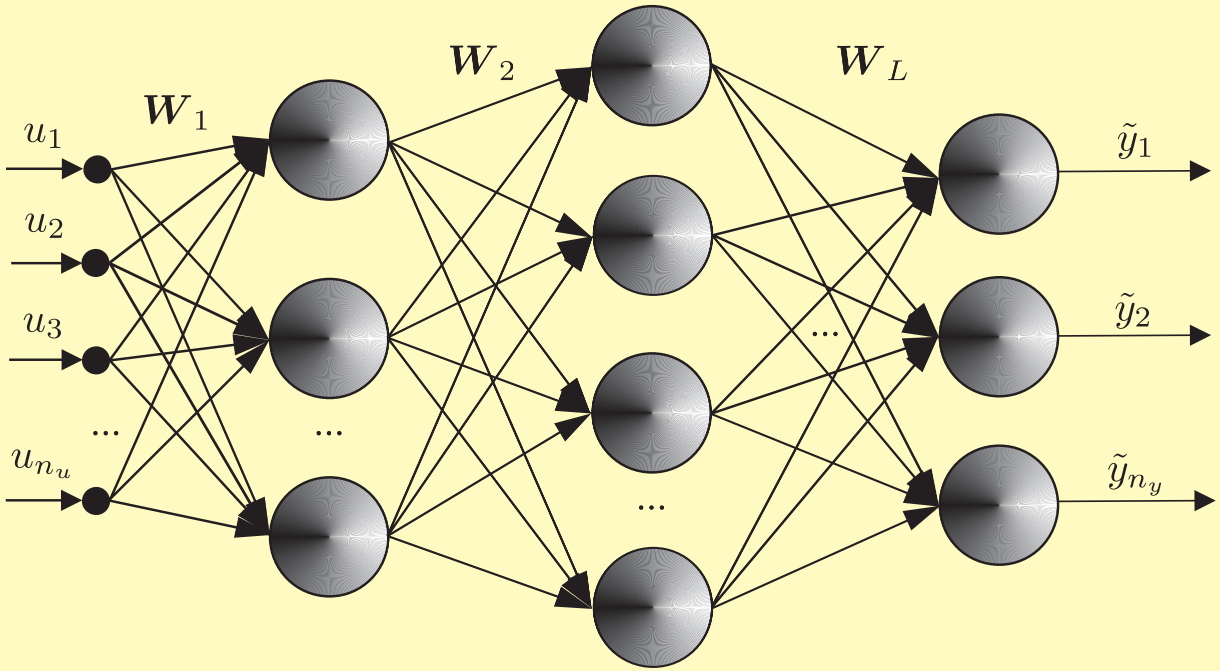


- $\boldsymbol{u} = [u_1, u_2, \ldots, u_m]^T$ – input vector
- $\boldsymbol{w} = [w_1, w_2, \ldots, w_m]^T$ – parameters or connection strength vector
- $\varphi = \sum_{i=1}^{m} p_i u_i + b = \boldsymbol{u}^T \boldsymbol{w} + b$ – membrane potential
- $F(\varphi)$ – activation function, e.g. linear, binary, sigmoid, etc.
- $\tilde{y} = F(\varphi)$ – neuron output

➜ Typical ANN architectures

The arrangement of neurons into layers and connection patterns within and between the layers is called the net architecture.

❒ Feedforward networks: a single-layer or multi-layer perceptron (MLP):



$$\tilde{\boldsymbol{y}} = f_L(\boldsymbol{W}_L, ..., f_2(\boldsymbol{W}_2 f_1(\boldsymbol{W}_1 \boldsymbol{u})))$$

➜ Setting weights – training

The method of setting the values of weights (training) is an important distinguishing characteristic of different neural networks.

Types of training:

- supervised: training is accomplished by presenting a sequence of training vectors or patterns, u^μ, $\mu = 1, 2, \ldots, P$, each with an associated target output vector, y_d.
- unsupervised: self-organizing neural networks group similar input vectors u^μ, $\mu = 1, 2, \ldots, P$ together without the use of training data to specify what a typical member of each group looks like or to which group each vector belongs.
  A sequence of input vectors u^μ, $\mu = 1, 2, \ldots, P$, is provided, but no target vectors are specified.

➜ Supervised training algorithms

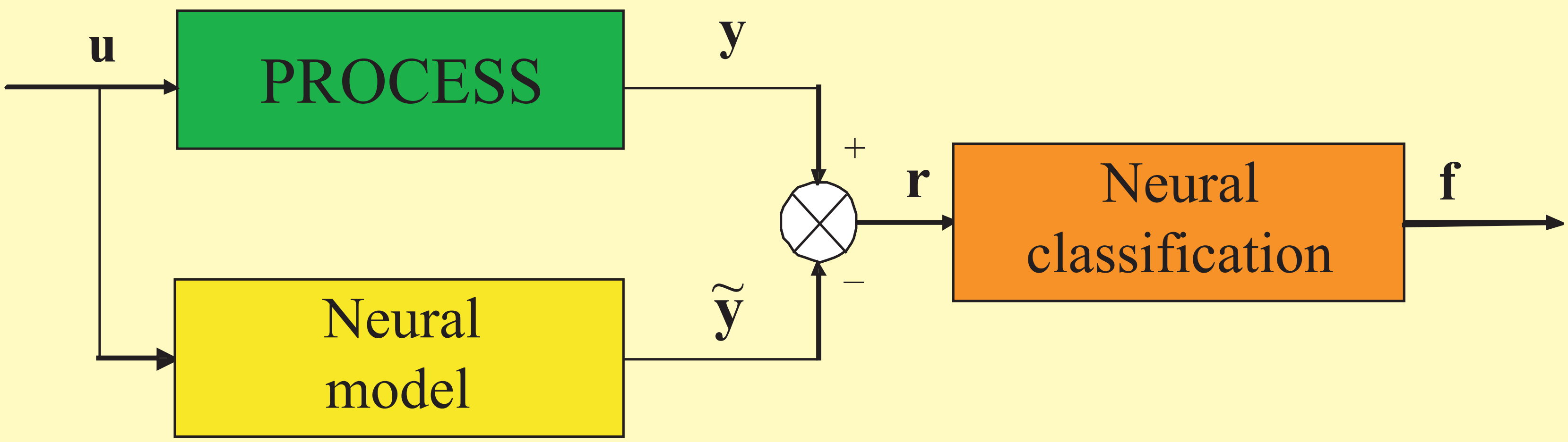
- Hebb rule
- Perceptron rule
- Delta rule (Widrow and Hoff, Least Mean Square, LMS)
- Back-Propagation (BP) algorithms
- Levenberg-Marquardt algorithm

☛ ANNs IN FAULT DIAGNOSIS

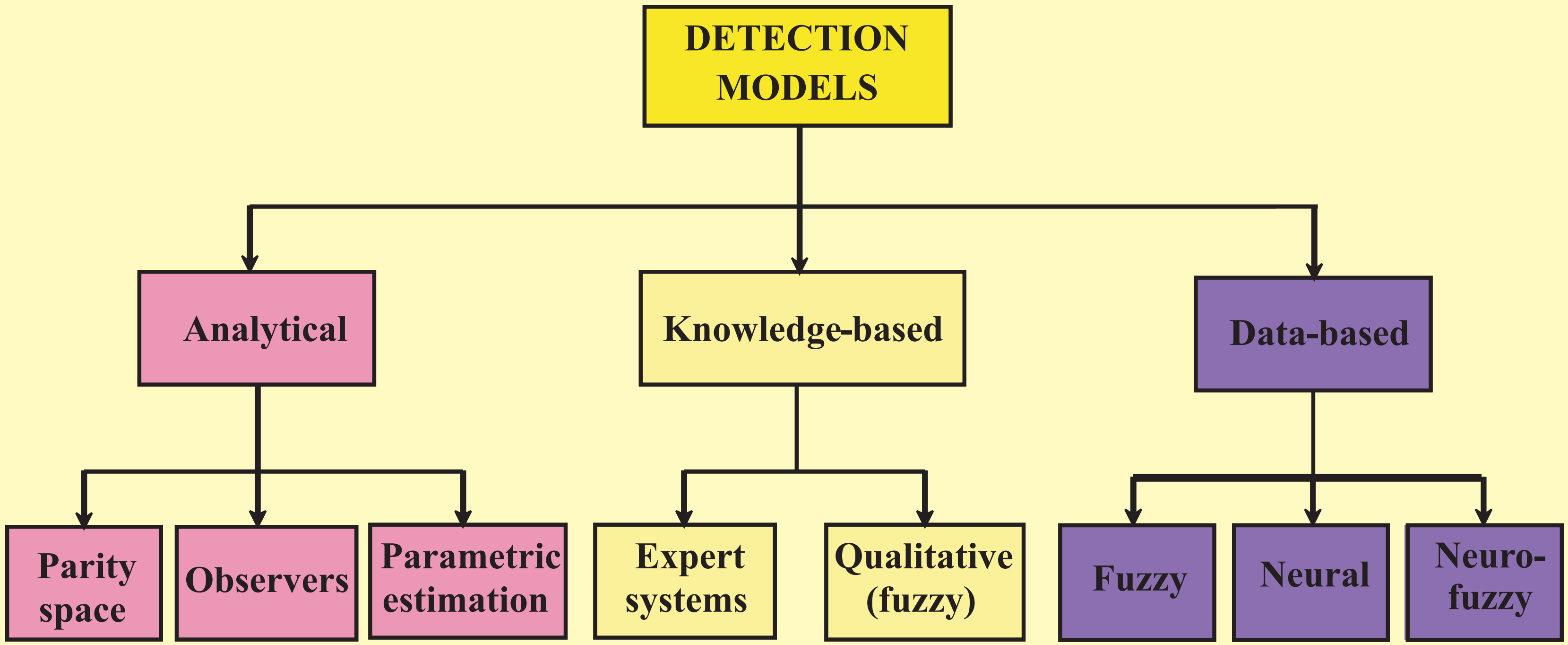
- Do not require an accurate analytical model of the diagnosed process
- Provide an excellent mathematical tool for dealing with non-linear problems
- Approximate any continuous function on a compact set with any accuracy, assuming that an infinite number of hidden neurons is available
- Ideal in cases where the required mapping algorithm is not known and tolerance to faulty input information is required
- Deal with the problem of dynamic system identification
- Need representative training data

➜ ANNs in fault diagnosis
(Frank and Köppen-Seliger, 1997; Isermann, 2005)

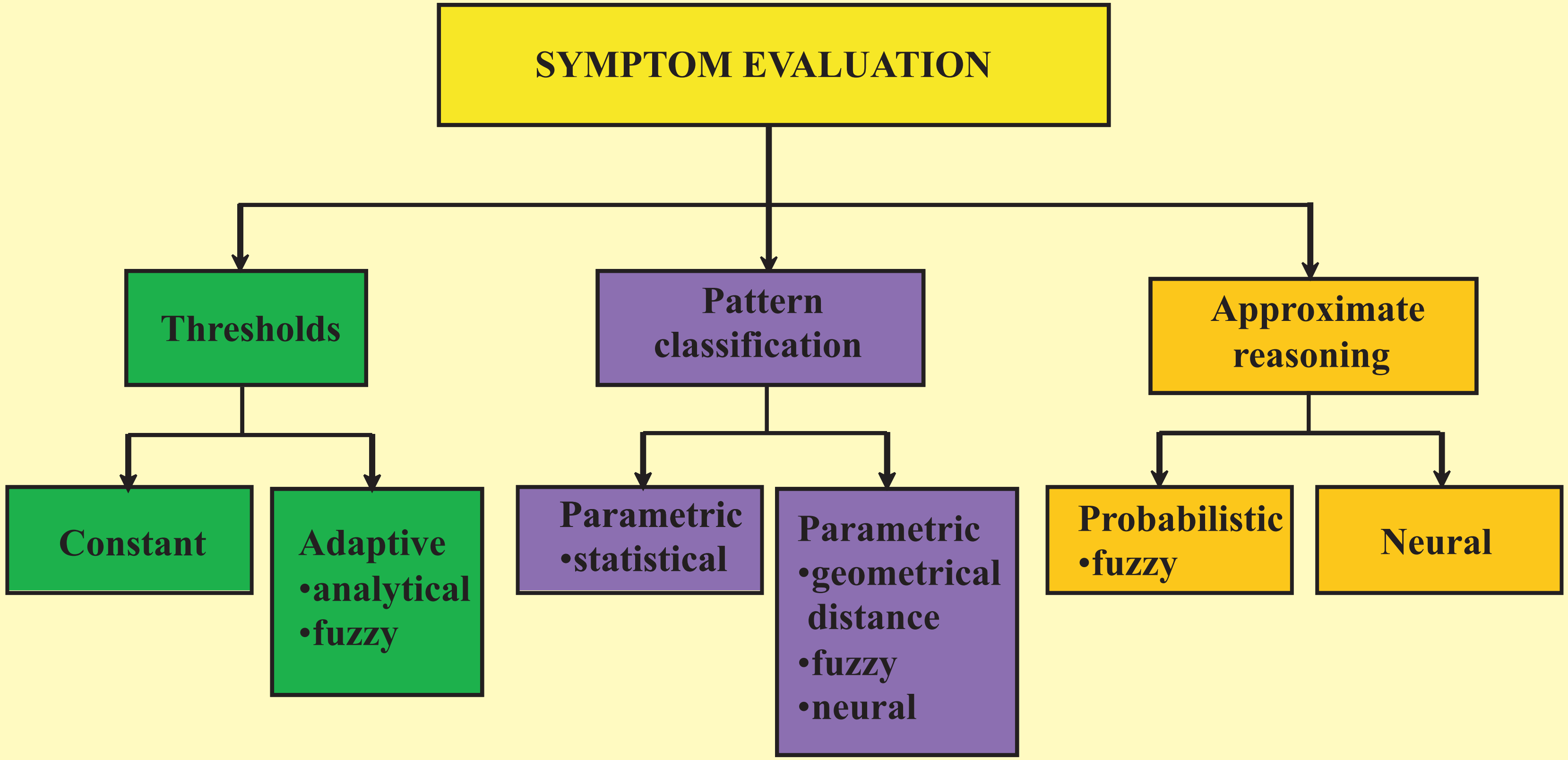
- Symptom generation (fault detection) – generation of symptoms which reflect faults
- Symptom evaluation (fault classification) – logical decision-making on the time of the occurrence and location of a fault



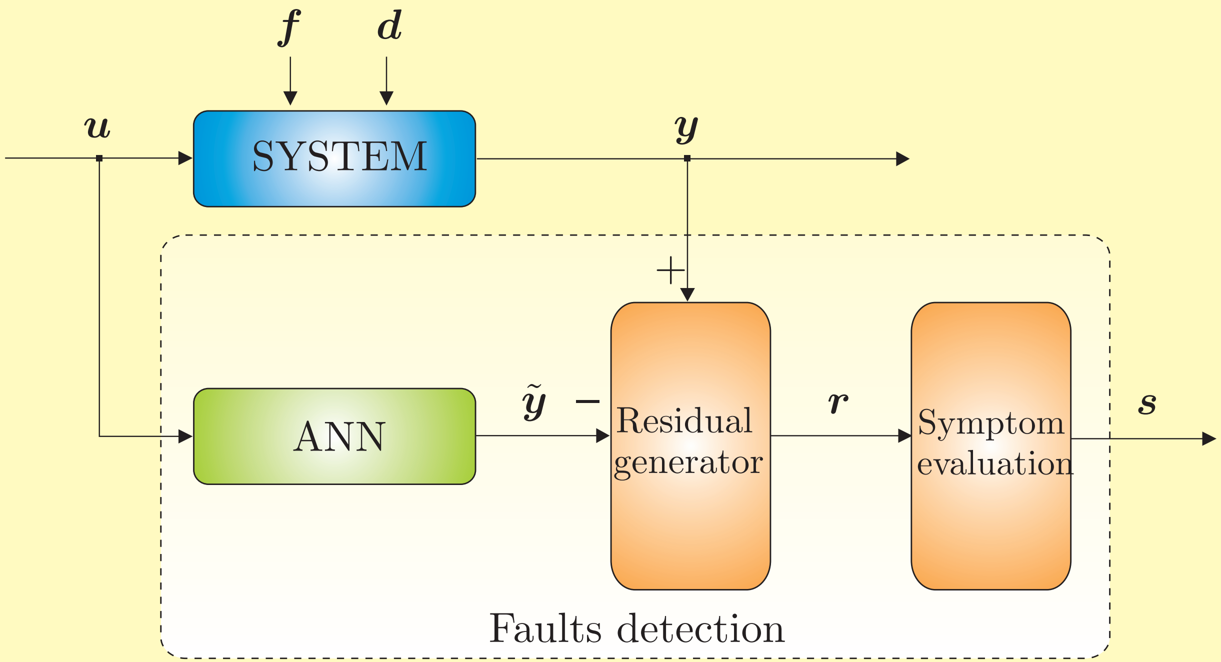
➜ Models for symptom generation



➜ Models for symptom evaluation

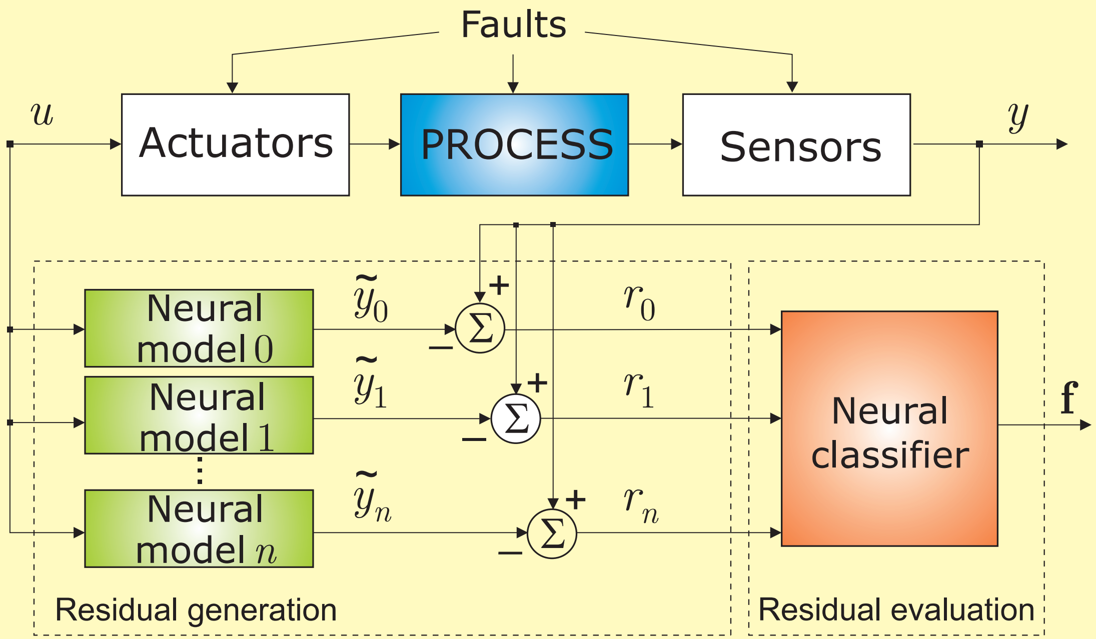


➜ ANN-based symptom generation



- For symptom generation purposes the ANN replaces the generally analytical model describing the process in the normal operation
- Before symptom generation, the ANN has to be trained for this task
- For the training purpose, an input and a corresponding output data are known
- For the validation purpose, data containing different faulty situations are known
- Problem: how to obtain such data from real processes?
- After completing the training, the ANN can be applied to on-line symptom generation

➜ ANNs-based symptom evaluation



- The task is to match each pattern of the residual vector with one of the pre-assigned classes of faults and the fault-free case
- In order to apply ANNs to residual evaluation, first of all residuals have to exist (they can be generated by another ANN or by one of analytical methods such as observers or parameter estimation)
- The residual $\mathbf{r} = [r_0, r_1, \ldots, r_n]^T$, which characterizes the classes of system behaviour, should be transformed by a classifier to determine the location and time of the occurence of faults.

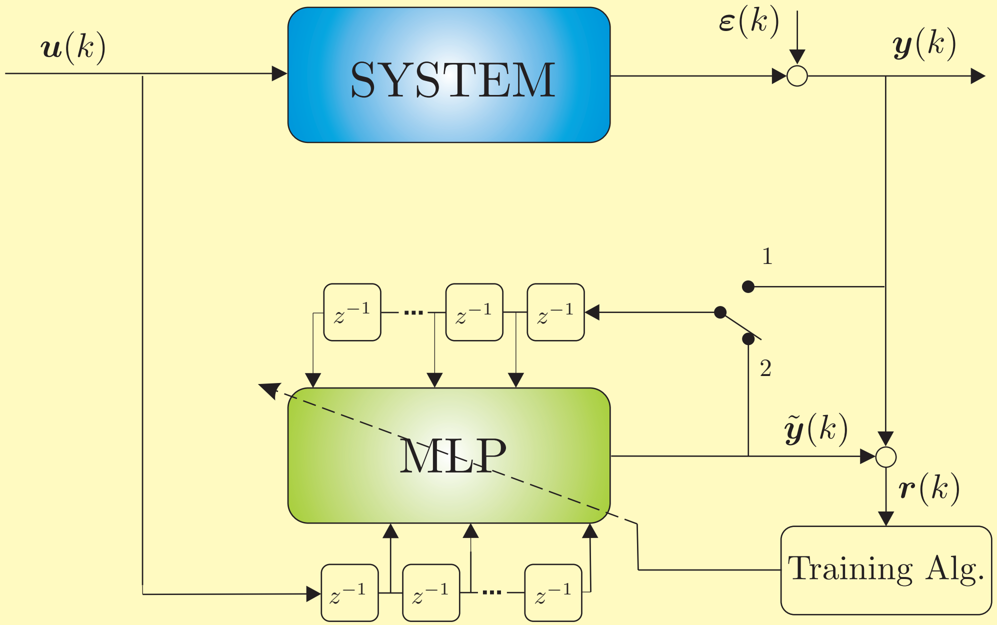
☛ MODELING OF SYSTEM DYNAMICS VIA ANNs

➜ Dynamic neural networks

- Globally recurrent networks – feedback is allowed between neurons of different layers or between neurons of the same layer:
  - MLP with external Time Delay Lines (TDL) (Gupta *et al.*, 2003)
  - Williams-Zipser neural network (Williams and Zipser, 1989)
  - Hopfield's neural network (Hopfield, 1982)
  - Elman's neural network (Elman, 1990)
  - recurrent MLP (Parlos *et al.*, 1994)
- Locally recurrent networks – feedback is only inside neuron models. These networks have a structure similar to that of static feed-forward ones, but consist of dynamic neuron models.

➜ Globally recurrent networks: MLP with external time delay lines

Neural residual generator with external TDL (Gupta *et al.*, 2003)

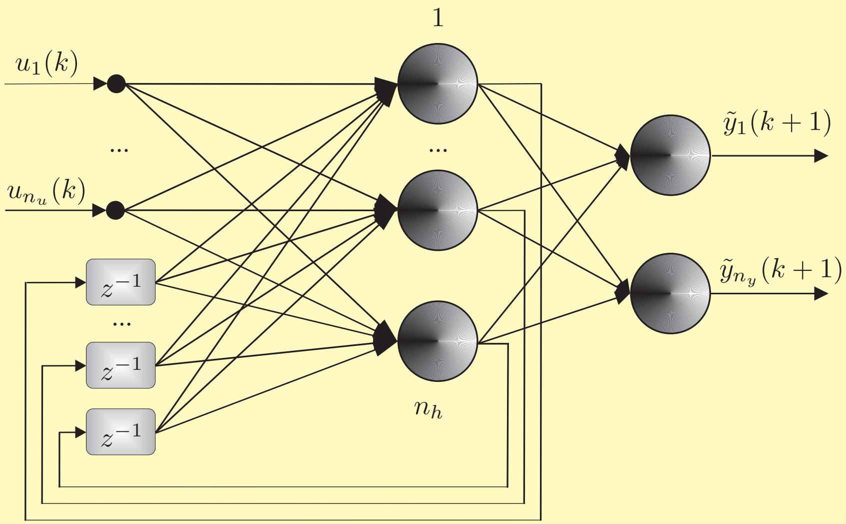


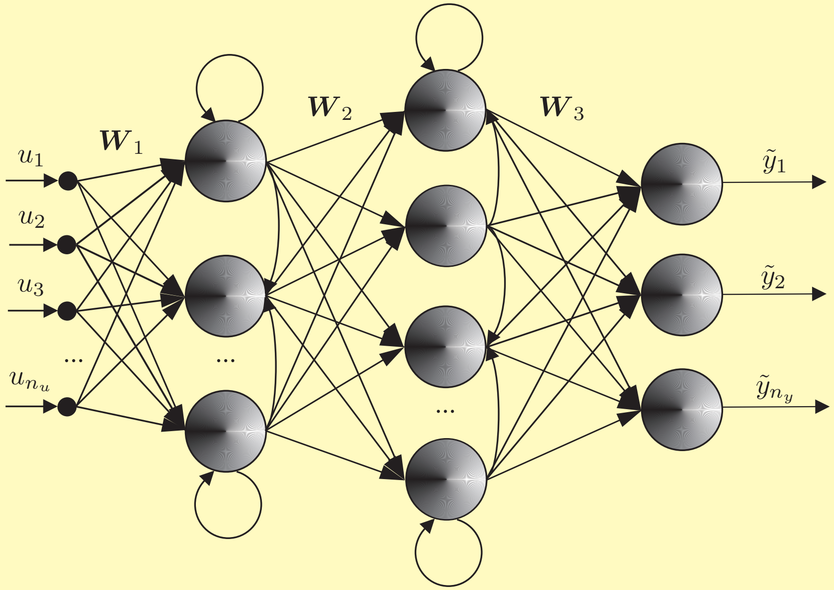
Input-output representation:

$$\tilde{y}(k) = \tilde{f}(y(k-1), \ldots, y(k-n_a), u(k), u(k-1), \ldots, u(k-n_b))$$
$$\tilde{y}(k) = \tilde{f}(\tilde{y}(k-1), \ldots, \tilde{y}(k-n_a), u(k), u(k-1), \ldots, u(k-n_b)),$$

where f and $\tilde{f}$ are non-linear functions of the network and diagnosed process

➜ Globally recurrent networks: Elman's neural network and recurrent MLP





Global recurrence: drawback – the stability problem

➜ Locally recurrent networks: dynamic neuron models

- Local activation feedback (Frasconi, 1992):

$$\tilde{y}(k) = \xi\Big(\varphi(k)\Big), \qquad \varphi(k) = \sum_{i=1}^{n_p} w_i x_i(k) + \sum_{j=1}^{n_b} b_j \varphi(k-j)$$

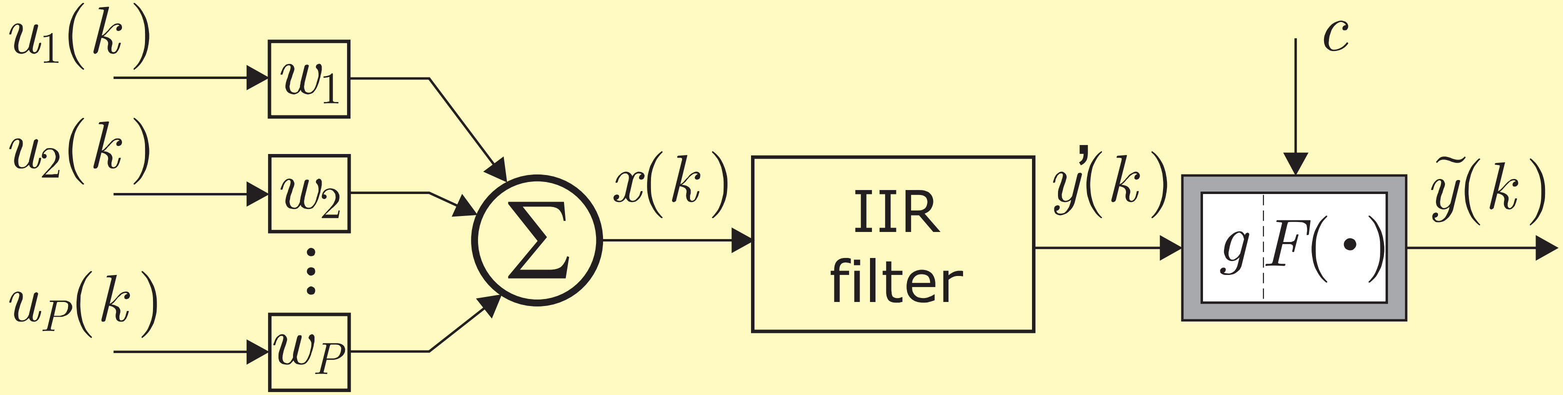
- Local output feedback (Gori, 1989):

$$\tilde{y}(k) = \xi\left(\sum_{i=1}^{n_p} w_i x_i(k) + \sum_{j=1}^{n_c} c_j \tilde{y}(k-j)\right)$$

- Local synapse feedback (Back, 1991):

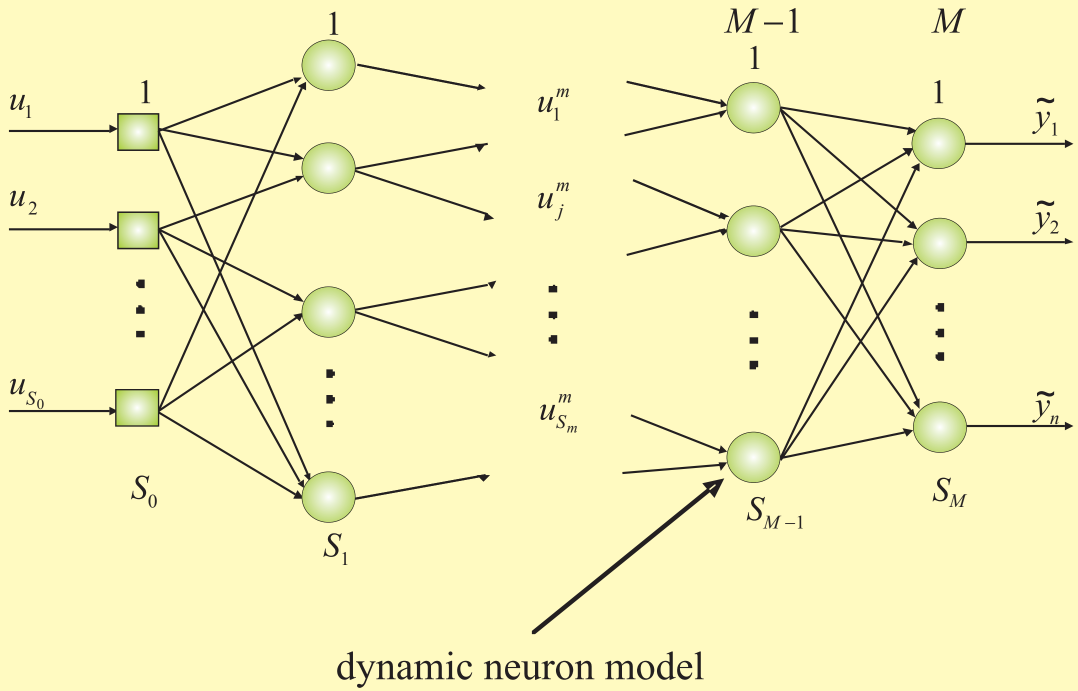
$$\tilde{y}(k) = \xi\left(\sum_{i=1}^{n_p} G_i(z^{-1}) x_i(k)\right), \qquad G_i(z^{-1}) = \frac{\sum_{j=0}^{n_b} b_j z^{-1}}{\sum_{j=0}^{n_a} a_j z^{-1}}$$

➜ Locally recurrent networks: a dynamic neuron model with the IIR filter (Korbicz, Patan and Obuchowicz, 1999)



- Adder module: $x(k) = \sum_{p=1}^{P} w_p u_p(k)$
- IIR filter module: $y'(k) = -\sum_{i=1}^{n} a_i y'(k-i) + \sum_{i=0}^{n} b_i x(k-i)$
- Activation module: $\tilde{y}(k) = F\big(g, y'(k), c\big)$

➜ Locally recurrent networks: a dynamic multilayered neural network



Network adaptable parameters:

$$\boldsymbol{v} = [\boldsymbol{w}_i^T, (\boldsymbol{a}_j^i)^T, (\boldsymbol{b}_j^i)^T, (g_{sj}^i)^T]^T | i = 1, ..., M; j = 1, ..., s_i$$

M – number of layers

s_i – number of neurons in the i-th layer

➜ Training algorithm – Extended Dynamic Back-Propagation (EDBP) (Korbicz, Patan and Obuchowicz, 1999)

- Performance index:

$$J(k) = \left\| \boldsymbol{y}(k) - \tilde{\boldsymbol{y}}(k) \right\|^2,$$

where
$\boldsymbol{y}(k)$ is the desired output of the network,
$\tilde{\boldsymbol{y}}(k)$ is the actual response of the ANN on the given input pattern $\boldsymbol{u}(k)$.

- Update rule:

$$v_j^m(k+1) = v_j^m(k) + \eta \delta_j^m(k) S_{vj}^m(k),$$

where

$$\delta_j^m(k) = \begin{cases} e_j(k) F'\left(y_j'^m(k)\right), & \text{for } m = M, \\ \sum_{l=1}^{s_{m+1}} \left(\delta_j^{m+1}(k) g_l^{m+1} b_{0l}^{m+1} w_{lj}^{m+1}\right) F'\left(y_j'^m(k)\right), & \text{for } m = 1, \ldots, M-1 \end{cases}$$

(i) Sensitivity with respect to the feedback filter parameter a_{ij}^m:

$$S_{a_i j}^m(k) = -g_{sj}^m y_j'^m(k-i), \quad i = 1, \ldots, n$$

(ii) Sensitivity with respect to the feed-forward filter parameter b_{ij}^m:

$$S_{b_i j}^m(k) = g_{sj}^m x_j^m(k-i) \quad i = 1, \ldots, n$$

(iii) Sensitivity with respect to the slope parameter g_j^m:

$$S_{g_s j}^m(k) = y_j'^m(k),$$
$$j = 1, \ldots, s_m$$

(iv) Sensitivity with respect to the bias c_j^m:

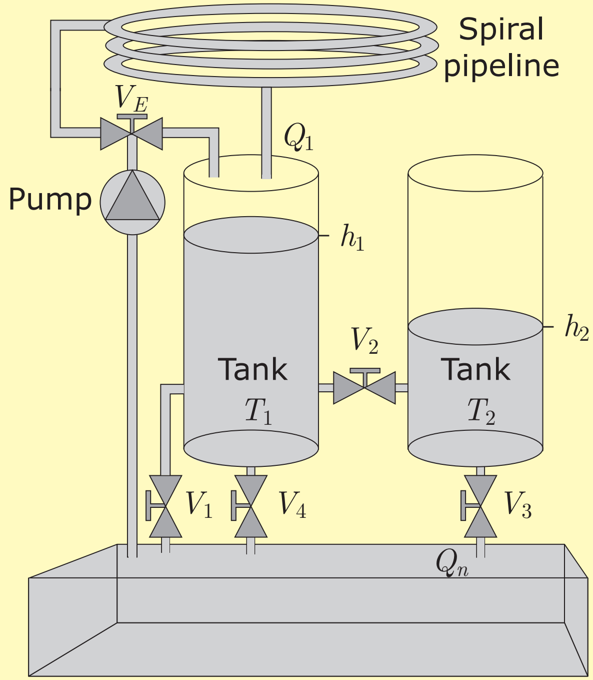
$$S_{cj}^m(k) = 1, \quad j = 1, \ldots, s_m,$$

(v) Sensitivity with respect to the weight w_{pj}^m:

$$S_{w_{pj}}^m(k) = g_j^m \left(\sum_{i=0}^n b_{ij}^m u_p^m(k-i) - \sum_{i=1}^n a_{ij}^m S_{w_p j}^m(k-i) \right),$$
$$j = 1, \ldots, s_m; \quad p = 1, \ldots, s_{m-1}.$$

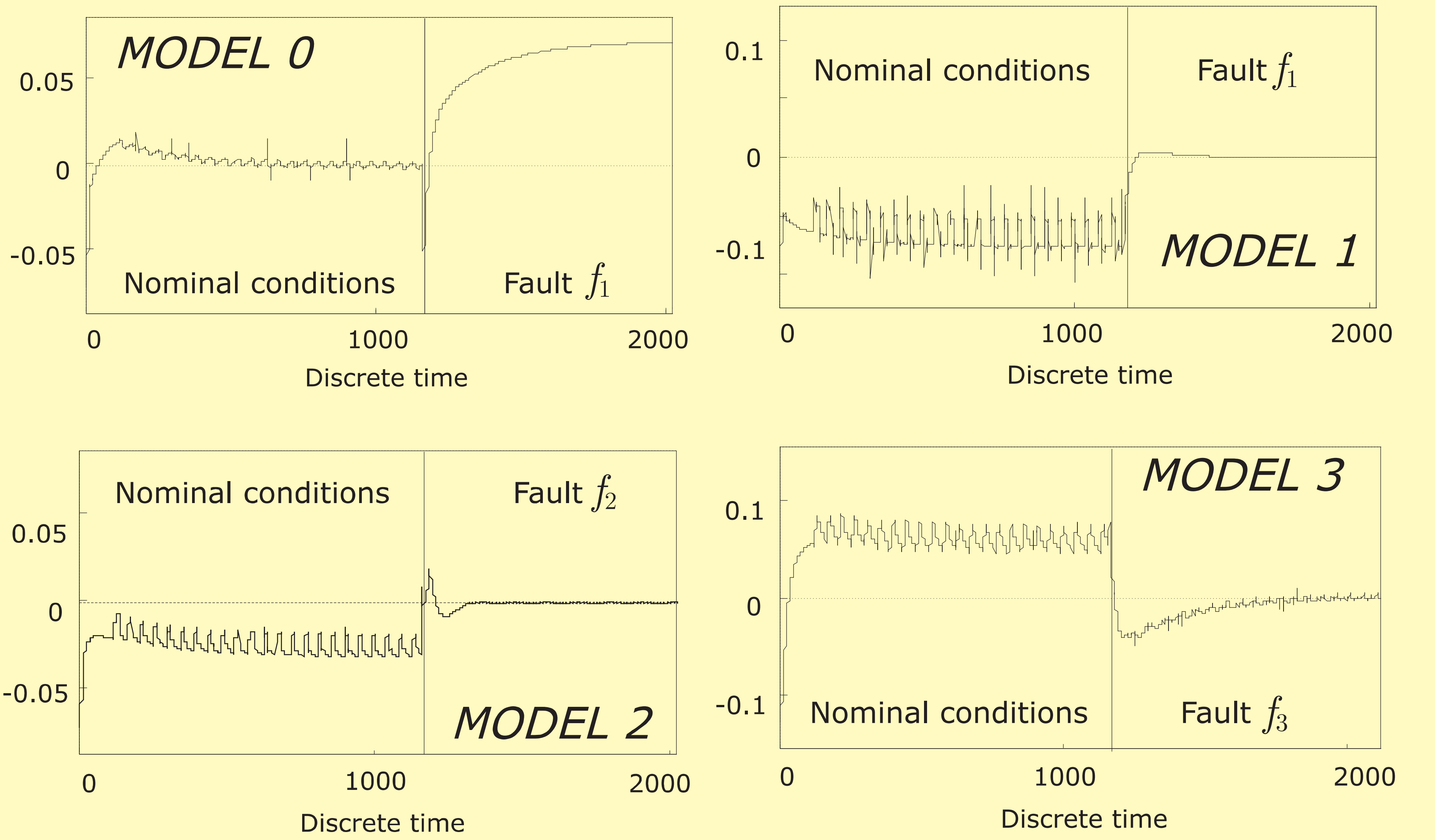
➜ Examples of fault diagnosis systems: a two-tank system

- Aim of system control: to keep a constant level of water in Tank 2
  - Q_1 – inflow of liquid through the pump to tank T_1
  - Q_n – outflow of tank T_2
  - h_1, h_2 – sensors for meas, liquid levels
  - V_1, V_2, V_3, V_4 and V_E – electronically controlled valves



- Possible faults:
  - Valve V_2 closed and blocked
  - Valve V_2 opened and blocked
  - Leak in Tank 1

➜ Residuals generated from neural models



➜ Dynamic Group Method of Data Handling (GMDH) neural networks (Ivakhnenko, 1971; Farlow, 1984)

Why GMDH?

- Successful identification depends on a proper selection of the model structure
- A GMDH approach can be successfully employed to automatic selection of the neural network structure
- The structure of the network is designed by gradually increasing its complexity
- Different techniques for parameter estimation of linear-in-parameter models can be used

Idea of the GMDH:

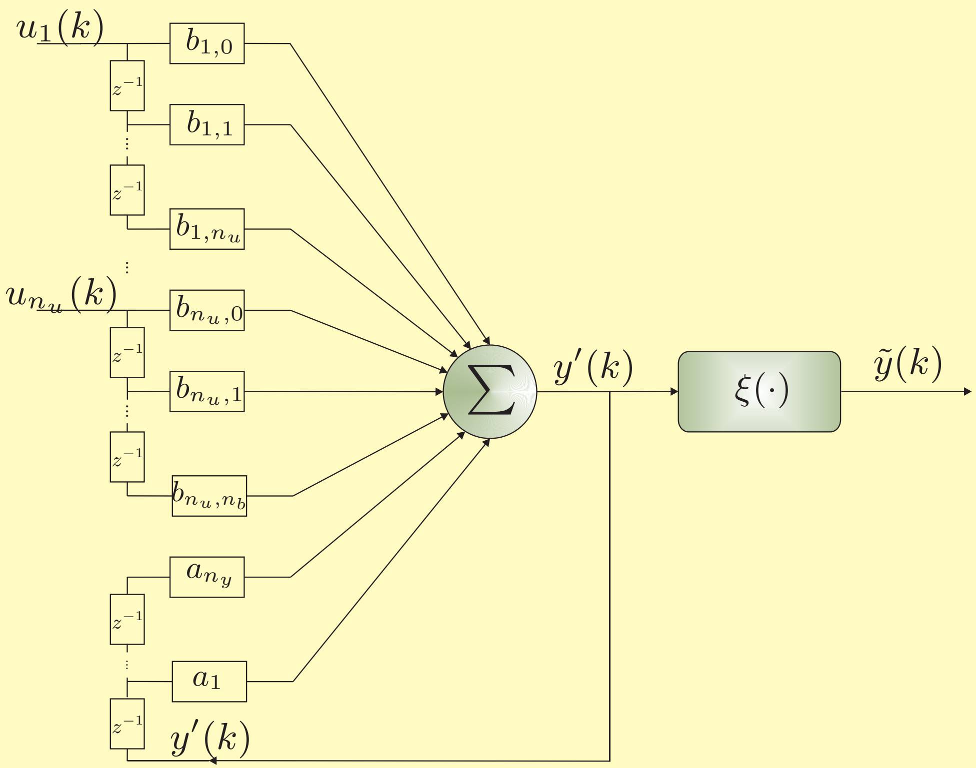
- replacing the complex model of the process with partial models (neurons) by using the rules of variable selection

➜ Neuron of the dynamic GMDH neural network (Mrugalski and Witczak, 2002)

$$\tilde{y}_n^{(l)}(k) = \xi\left(\left(\boldsymbol{z}_n^{(l)}(k)\right)^T \hat{\boldsymbol{\theta}}_n^{(l)}\right),$$

where

- $\tilde{y}_n^{(l)}(k)$ – neuron output
- $\hat{\boldsymbol{\theta}}_n^{(l)}$ – parameters vector
- $\boldsymbol{z}_n^{(l)}(k) = \boldsymbol{f}\left([u_i^{(l)}(k), u_j^{(l)}(k)]^T\right)$, $i, j = 1, \ldots, n_u$ – regressor vector
- l – layer number
- n – neuron number in the l-th layer
- $\xi(\cdot)$ – nonlinear invertible activation function

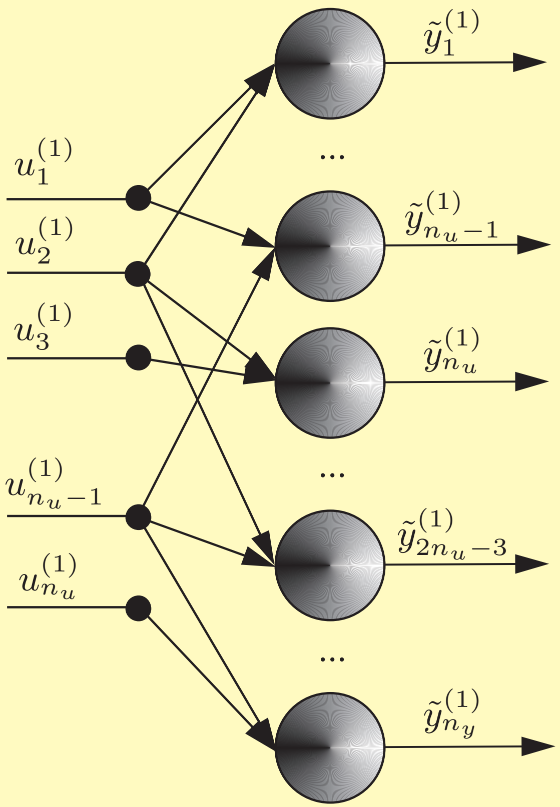


Statement:

Independently of the methods applied to $\boldsymbol{\theta}$ estimation there is the uncertainty of the neural model

➜ Synthesis of the GMDH network

The input layer of two-input neurons is given by



$$
\begin{cases}
\tilde{y}_1^{(1)} = f(u_1^{(1)}, u_2^{(1)}, \hat{\boldsymbol{\theta}}_{1,2}), \\
\tilde{y}_2^{(1)} = f(u_1^{(1)}, u_3^{(1)}, \hat{\boldsymbol{\theta}}_{1,3}), \\
\quad \ldots \\
\tilde{y}_{n_u-1}^{(1)} = f(u_1^{(1)}, u_{n_u}^{(1)}, \hat{\boldsymbol{\theta}}_{1,n_u-1}), \\
\tilde{y}_{n_u}^{(1)} = f(u_2^{(1)}, u_3^{(1)}, \hat{\boldsymbol{\theta}}_{2,3}), \\
\quad \ldots \\
\tilde{y}_{2n_u-3}^{(1)} = f(u_2^{(1)}, u_{n_u}^{(1)}, \hat{\boldsymbol{\theta}}_{2,n_u}), \\
\quad \ldots \\
\tilde{y}_{n_y}^{(1)} = f(u_{n_u-1}^{(1)}, u_{n_u}^{(1)}, \hat{\boldsymbol{\theta}}_{n_u-1,n_u}).
\end{cases}
$$

To define the unknown parameters $\hat{\boldsymbol{\theta}}_{i,j}$, the Least Mean Squares (LMS) method can be applied.

➜ Synthesis of the GMDH network (Mueller and Lemke, 2000)

Partial models evaluation:

- *Final Prediction Error – FPE*

$$\frac{n_{\mathcal{D}} + n_p}{n_{\mathcal{D}} - n_p} s_e^2$$

- *Akaike Information Criterion – AIC*

$$n_{\mathcal{D}} \log s_e^2 + 2n_p + c$$

- *Convergence criterion –* $i^2(n_{\mathcal{D}})$

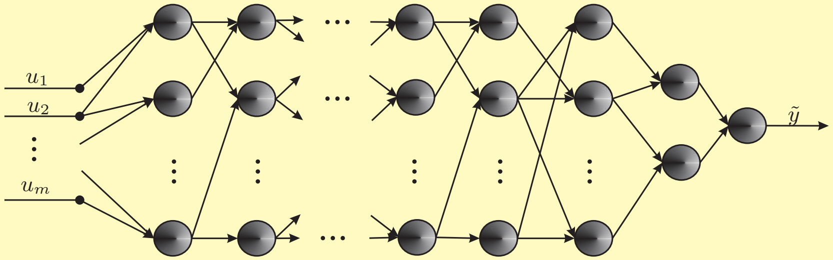
$$\sum_{k=1}^{n_{\mathcal{D}}} \left(\hat{y}_n^{(l)}(k) - y(k)\right)^2 \Big/ \sum_{k=1}^{n_{\mathcal{D}}} y(k)^2$$

  - $s_e^2 = \frac{1}{n_{\mathcal{D}}} \sum_{k=1}^{n_{\mathcal{D}}} \varepsilon(k)^2 = \frac{1}{n_{\mathcal{D}}} \sum_{k=1}^{n_{\mathcal{D}}} \left(y(k) - \hat{y}_n^{(l)}(k)\right)^2$
  - n_p – parameters number
  - y – system output
  - $\hat{y}_n^{(l)}$ – neuron output for all data sets $n_{\mathcal{D}}$

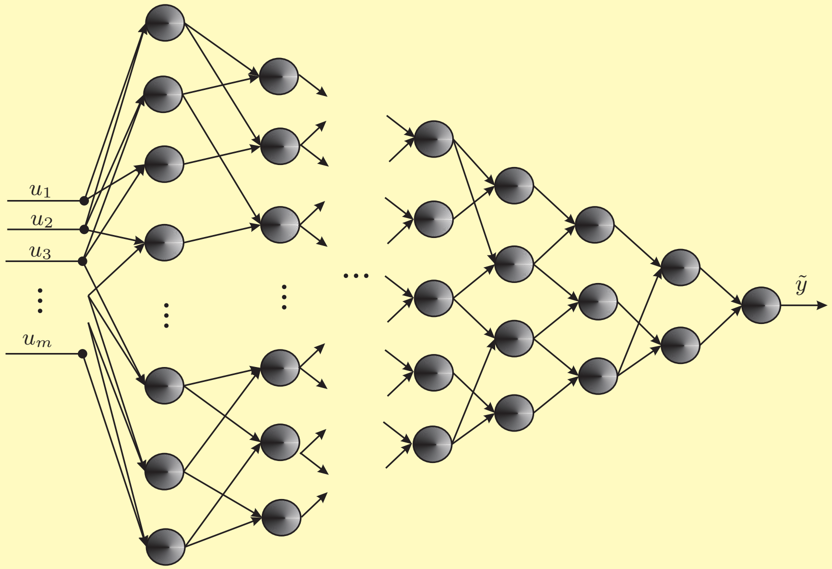
➜ Synthesis of GMDH neural networks

Selection methods of best-performing neurons – an element of the network structural optimization

- *Constant population method* is based on the selection of g neurons, for which $Q(\hat{y}_n^{(l)})$ reaches the least values

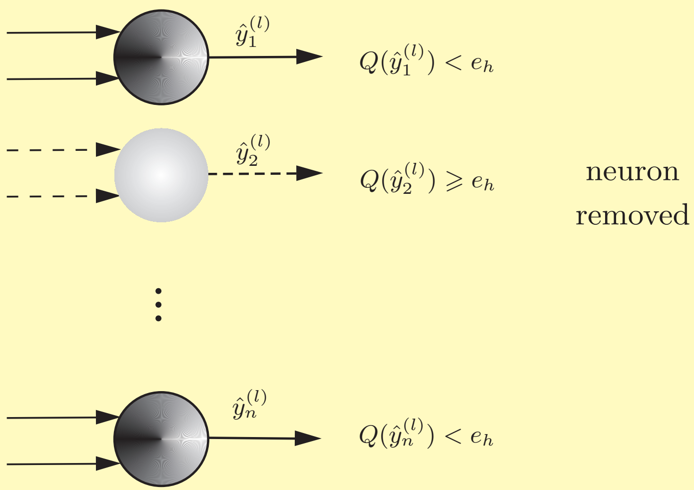


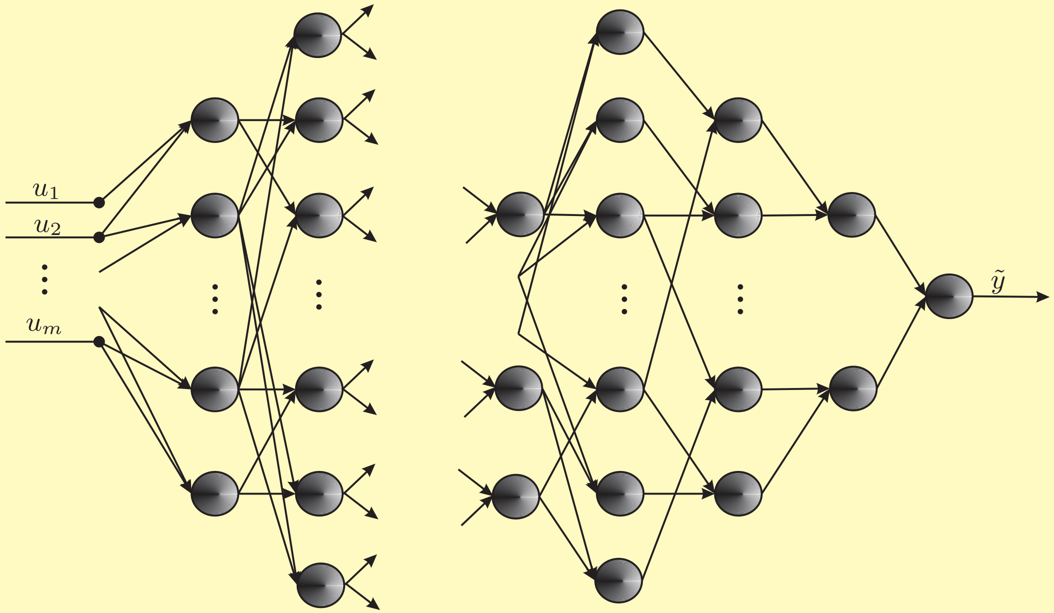
- *Decreasing population method* defines the maximum number of elements in a layer. The number of neurons in each layer decreases along with the growth of the network



➜ Synthesis of GMDH neural networks

- *The optimal population method* is based on the rejection of neurons for which the defined quality index is bigger than the arbitrarily determined threshold e_h:

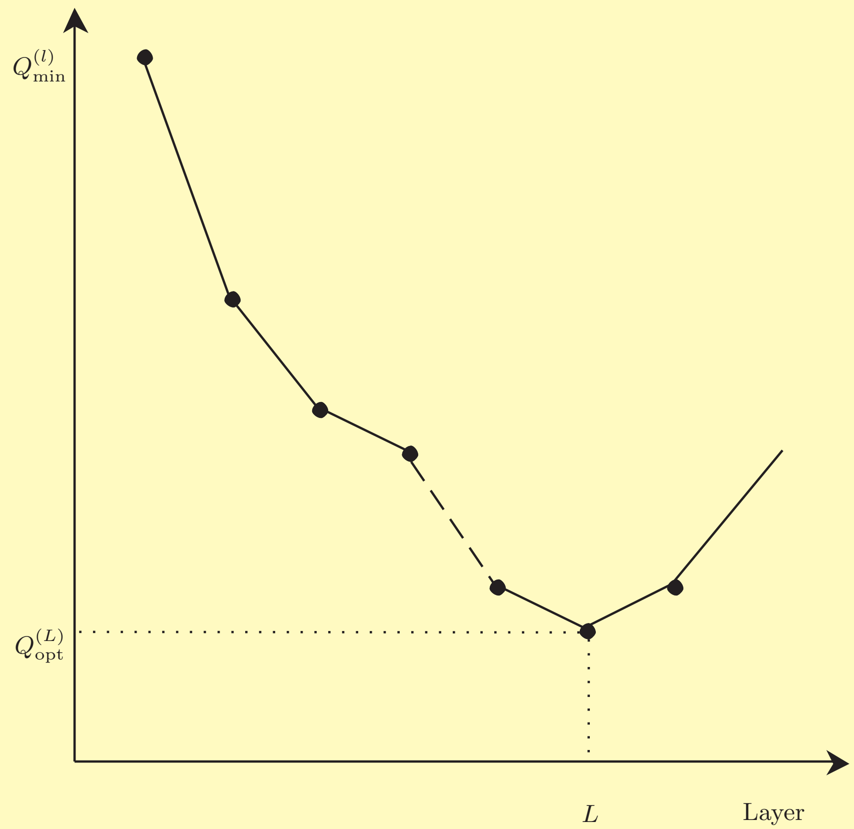




➜ Synthesis of GMDH neural networks

Mathematical description of the second layer:

$$\begin{cases} u_1^{(l+1)} = \tilde{y}_1^{(l)}, \\ u_2^{(l+1)} = \tilde{y}_2^{(l)}, \\ \qquad \dots \\ u_{n_u}^{(l+1)} = \tilde{y}_{n_y}^{(l)}. \end{cases}$$

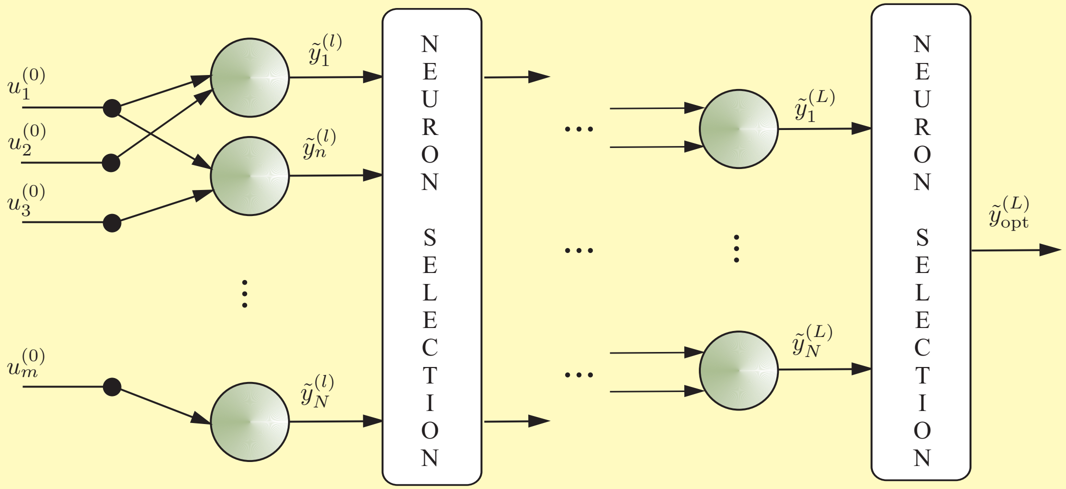


The procedures of:

- parameter identification
- partial models evaluation
- partial models selection

are repeated over till the transition error starts growing.

➜ Synthesis of GMDH neural networks

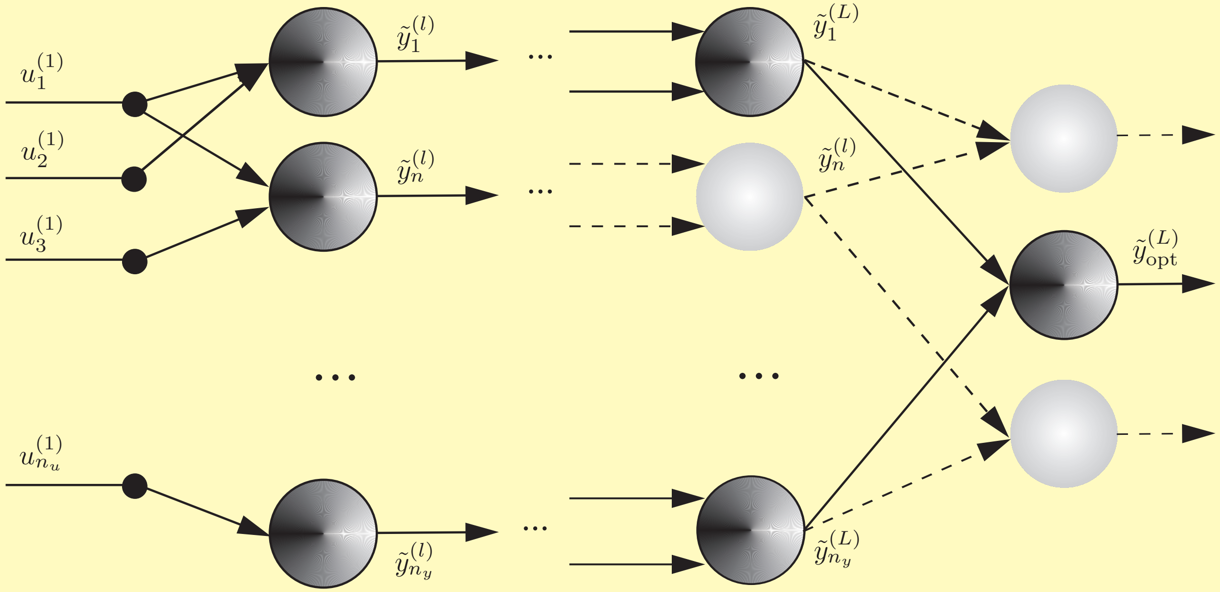


Step 1 : Determine all neurons (estimate their parameter vectors $\boldsymbol{\theta}_n^{(l)}$ with the training data set $\mathcal{T}$) whose inputs consist of all possible couples of input variables, i.e. $(n_u - 1)n_u/2$

Step 2 : Using a validation data set $\mathcal{V}$ select several neurons which are best-fitted in terms of the chosen quality index

Step 3 : If the termination condition is fulfilled then STOP, otherwise use the outputs of the best-fitted neurons (selected in *Step 2*) to form the input vector for the next layer, and then go to *Step 1*

➜ Final structure of the GMDH network

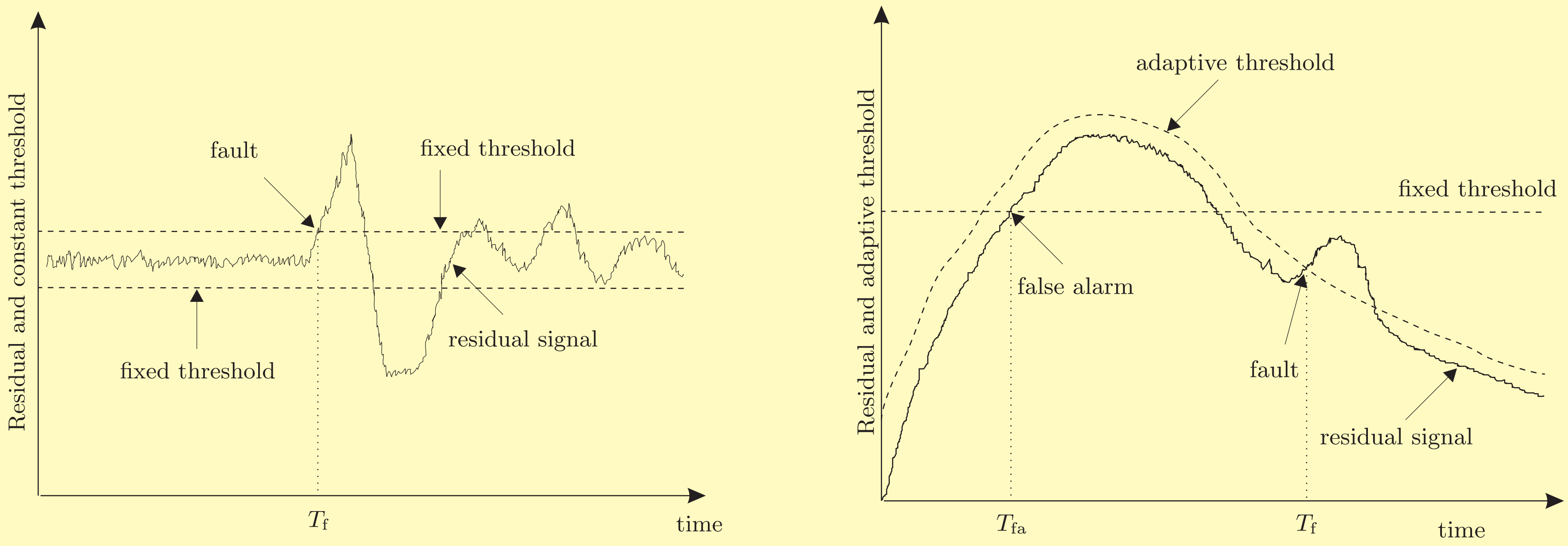


☛ ROBUST FAULT DETECTION APPROACHES

➜ Why is the uncertainty of neural models considered?

- Training algorithms = identification algorithms based on input-output observations
- Multi-layered perceptron modeling – only parameter identification (the model structure is assumed)
- GMDH modelling – structure and parameters identification
- Result of training – irrespective of the identification method used there is always the model-reality mismatch

➜ Problem of robust fault detection (Frank and Ding, 1997)



Solution: Passive approaches – providing an adaptive threshold taking into account model uncertainty

➜ Assumptions of passive model-based fault diagnosis (Papadopoulos *et al.*, 2001)

- No structural errors – the structure of the model is the same as that of the system
- Disturbances and noise acting upon the system are known
- The model is linear with respect to parameters
- A large number of data points should be available

Objective – designing a robust fault detection scheme by using artificial neural networks and a model error modelling technique

➜ Development of robust fault diagnosis (Patan, 2005)

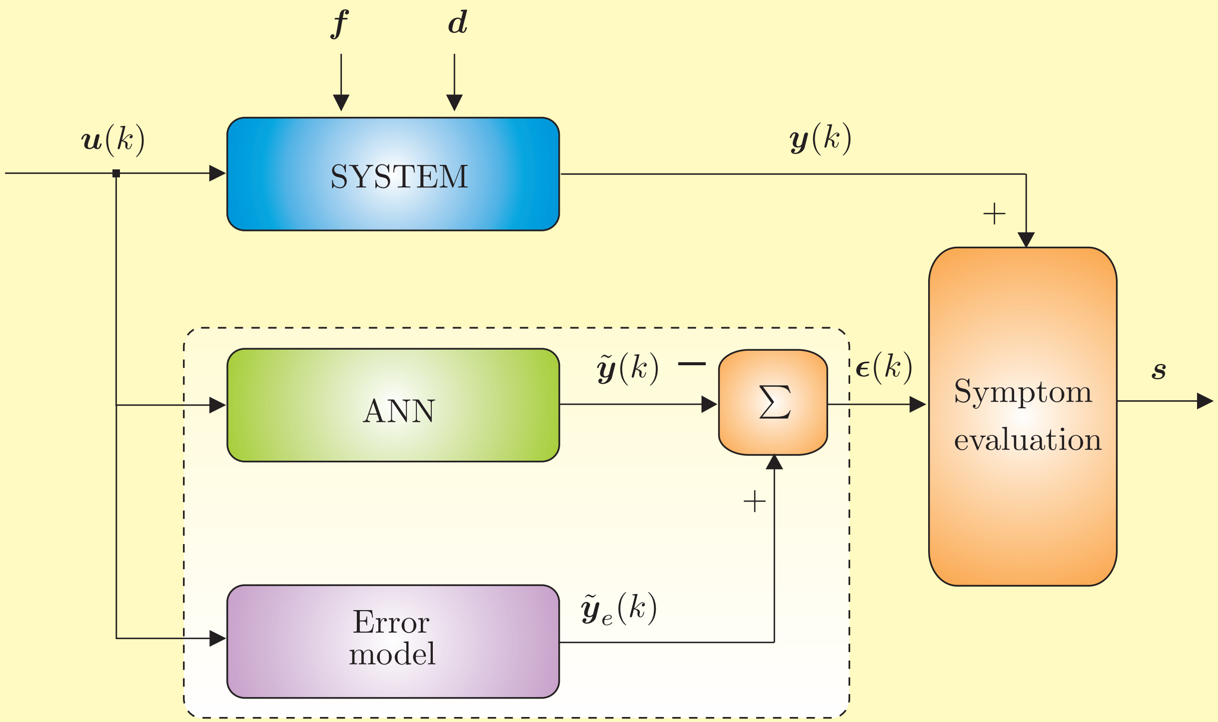
Model error modelling – the MEM procedure:

- The uncertainty of the model is estimated by analyzing residuals
- The uncertainty is a measure of unmodelled dynamics, noise and disturbances
- The proposed approach will
  - design a model of uncertainty by using MLP with tapped delay lines (neural network ARX)
  - construct uncertainty bands in the time domain (on-line fault diagnosis)
- The centre of the uncertainty region is the signal $\tilde{y} + \tilde{y}_e$, where
  - $\tilde{y}$ is the model output
  - $\tilde{y}_e$ is the error model output

➜ Development of robust fault diagnosis

MEM procedure:

1. Compute $r = y - \tilde{y}$, where y and $\tilde{y}$ are desired and model outputs
2. Collect the data $\{u_i, r_i\}_{i=1}^{N}$ and identify an error model. This model constitutes an estimate of the error due to under-modelling, and is called the Model Error Model (MEM)
3. Construct a model along with uncertainty using both nominal and model error models:



➜ Development of robust fault diagnosis

Confidence bands:

- The response of this network representing the model error model is used to form the uncertainty band:

  $r_u = \tilde{y} + \tilde{y}_e + t_\alpha v$ – the upper band

  $r_l = \tilde{y} + \tilde{y}_e - t_\alpha v$ – the lower band

  where

  - $\tilde{y}$ – output of the model
  - $\tilde{y}_e$ – output of the error model
  - t_α – $N(0,1)$ tabulated value assigned to $1-\alpha$ confidence level
  - v – the standard deviation of $\tilde{y}_e$

- Observing the system output y, one may decide whether the fault occurred or not. If y is inside the confidence bounds, the system is healthy.

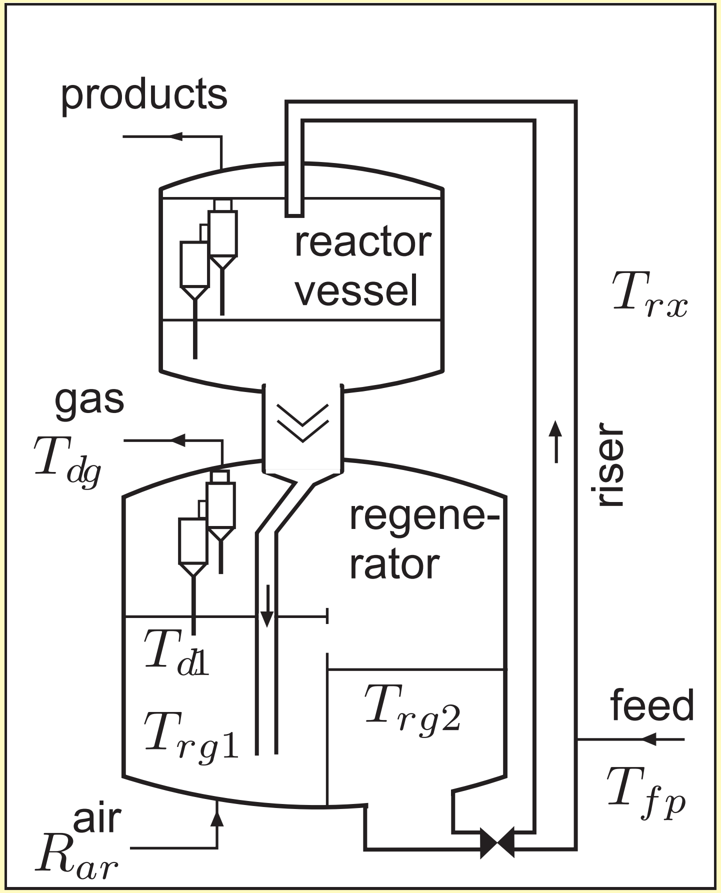
➜ Industrial example: robust fault detection with the MEM

The catalytic cracking process has been implemented in Simulink as an FCC benchmark (`http://www.enq.ufrgs.br/recope/FCC`)

Process considered:

$$T_{rx} = f(T_{rg2}, T_{fp})$$

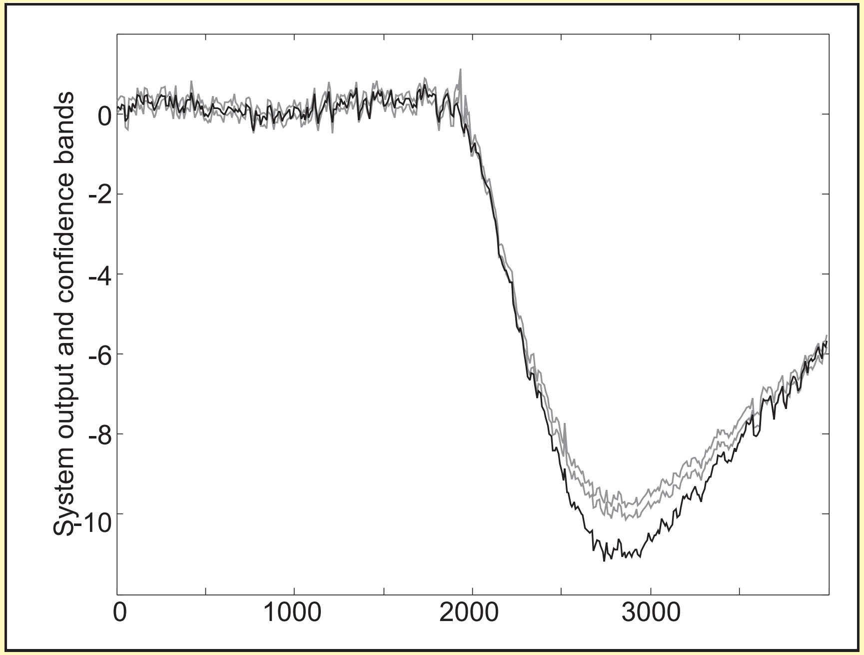
- T_{rx} – temp. of the cracking mixture
- T_{fp} – feed temp. at the riser entrance
- T_{rg2} – temp. of the dense phase at regenerator second stage

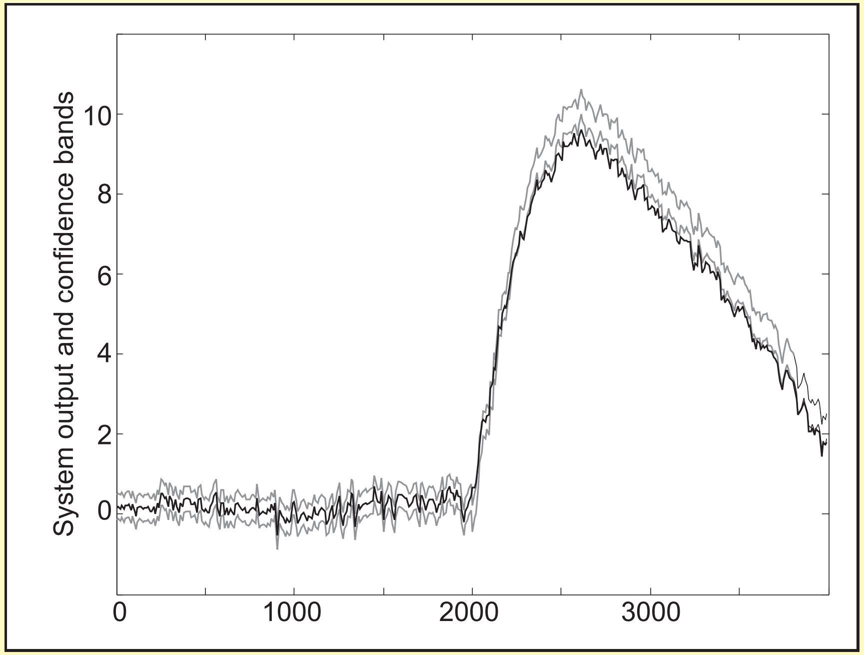


➜ Industrial example: robust fault detection with the MEM

- f_1 – 10% increase in catalyst density
- f_2 – 15% decrease in weir constant of the first and second stages
- significance level: 99% $(\alpha = 0.01)$

Fault detection: f_1 (left), f_2 (right)





➜ Experimental design for ANN-based robust fault detection (Witczak, 2005)

Statistical approach:

$$|y(k) - \hat{y}(k, \hat{\boldsymbol{\theta}})| \leqslant \mathrm{t}_{n_t - n_p}^{\alpha/2} \hat{\sigma} (1 + \boldsymbol{z}^T(t) \, \boldsymbol{F}^{-1} \, \boldsymbol{z}(t))^{1/2}$$

- **Main idea:** determining experimental conditions adapted to the final purpose of the modelling:

$$\Xi = \begin{Bmatrix} \boldsymbol{u}_1 & \ldots & \boldsymbol{u}_{n_e} \\ \mu_1 & \ldots & \mu_{n_e} \end{Bmatrix}$$

  – $\boldsymbol{u}_k \in U \subset \mathbb{R}^{n_u}$ – k-th support point
  – μ_k – weight of the k-th support point, $\sum_{i=1}^{n_e} \mu_i = 1$

- **Optimization problem:**

$$\Xi^* = \arg \underset{\Xi \in \boldsymbol{\Xi}}{\{\max, \min\}} \, \phi[\boldsymbol{F}(\boldsymbol{\theta}, \Xi)]$$

  ◦ **F** – Fisher information matrix
  ◦ $\phi(\cdot)$ – scalar function

➜ G-optimality criteria (Fukumizu, 2000; Uciński, 2005)

- Minimizing the variance of the estimated model's output:

$$\Xi_G^* = \arg\min_{\Xi \in \boldsymbol{\Xi}} \max_{\boldsymbol{u} \in \mathbb{U}} \boldsymbol{z}_k^T(\boldsymbol{u})\, \mathrm{F}^{-1}(\boldsymbol{\theta}, \Xi)\, \boldsymbol{z}_k(\boldsymbol{u})$$

  – $\boldsymbol{z}_k(\boldsymbol{u}) \in \mathbb{P} \subset R^{n_p}$ – vector of first-order sensitivity functions of the model

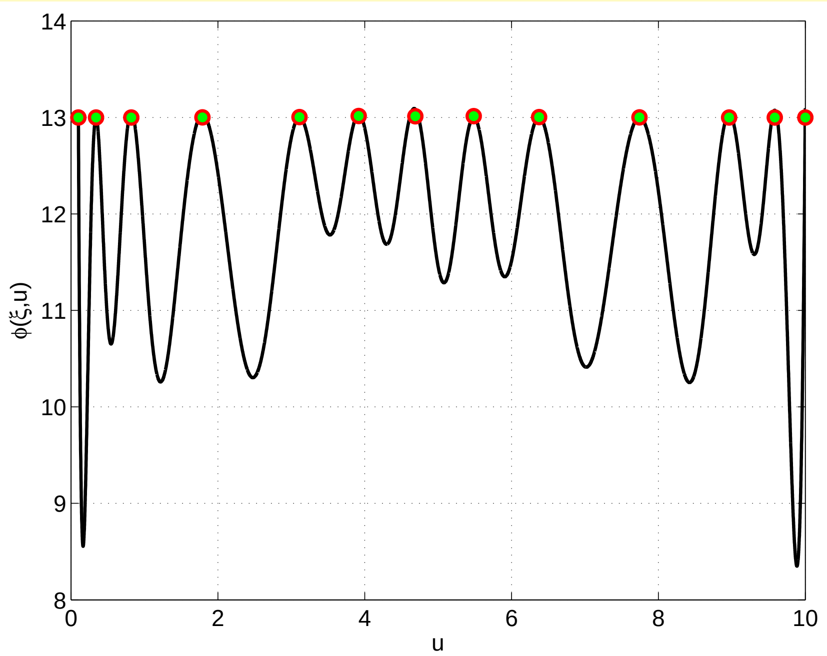
- Equivalence theorem (Kiefer and Wolfowitz, 1960):
  – equivalence between G-optimality and D-optimality criteria

- Search for D-optimality experimental design:
  – Wynn-Fedorov (1972)
  – DETMAX (Mitchell, 1974)

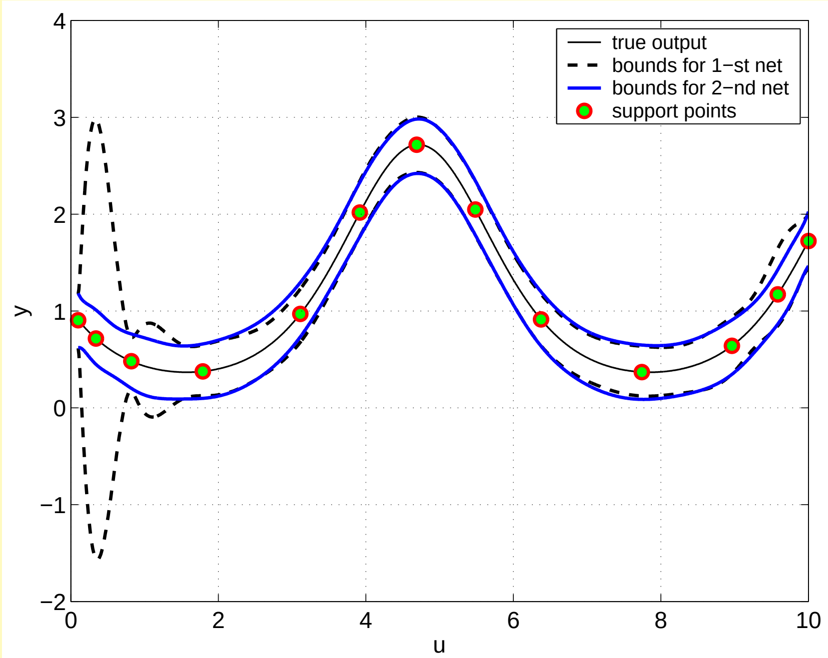
➜ Numerical simulation

- Function approximation:
  - $y_k = exp(-sin(u_k)) + \epsilon_k, \quad \epsilon \sim \mathcal{N}(0, 0.02^2)$, $u_k \in [0.1, 10]$
- Neural network used in the example:
  - one-dimensional input $u_k \in \mathbb{R}^1$
  - four hidden units with a hyperbolic tangent activation function
  - one output neuron with a linear activation function
- Algorithms used:
  - Levenberg-Marquardt method (estimation parameters)
  - Wynn-Fedorov algorithm (D-Optimal experimental design)

➜ Numerical simulation



Variance function



Output and its bounds

➜ Designing a robust fault detection scheme by using a GMDH network

Model uncertainties in the GMDH network:

- Structural errors:
  - application of the classical evaluation criteria
  - selection of inappropriate neurons during the neuron selection procedure
  - the structure of the neuron is not the same as that of the system
- Parameter estimation errors:
  - assumption that the noise nature is known
  - non-linear neuron – an invertible activation function
  - application of methods for parameter estimation of linear-in-parameter models in the case of dynamic neurons

➜ Parameter estimation methods

Statement:

The usual statistical parameter estimation methods, e.g. the least-square method, assume that data are corrupted by errors which can be modeled as realisations of independent random variables with a known or parameterised distribution.

Alternative approach:

A more realistic approach is to assume that errors lie between prior bounds.

➜ Confidence estimation of GMDH neural networks

- The problem is to obtain $\hat{\boldsymbol{\theta}}_n^{(l)}(k)$, and associated parameter uncertainty – the admissible parameter set $\mathbb{P}$
- The knowledge regarding the set of admissible parameters allows obtaining the confidence region of the model output:

$$\tilde{y}^m(k) \leqslant y(k) \leqslant \tilde{y}^M(k),$$

where $\tilde{y}^m(k)$ and $\tilde{y}^M(k)$ are the minimum and maximum admissible values of the model output consistent with the input-output measurements of the system.

➜ Bounded-error approach (BEA) (Schweppe, 1968; Walter and Pronzato, 1997)

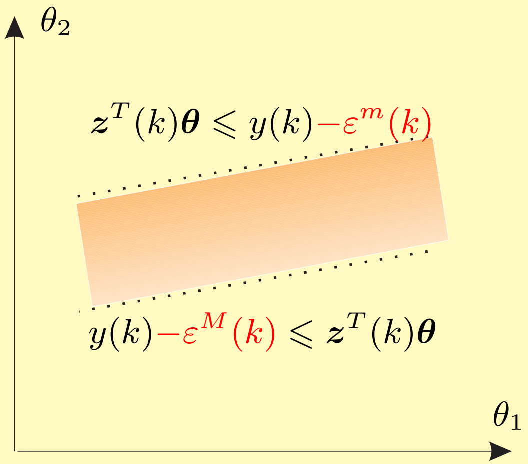
- Let us consider the following static system:

$$y(k) = \boldsymbol{z}^T(k)\boldsymbol{\theta} + \varepsilon(k),$$

where the bounds are known *a priori*:

$$\varepsilon^m(k) \leqslant \varepsilon(k) \leqslant \varepsilon^M(k).$$

- Let $\mathbb{S}(k)$ be a strip in the parameter space:

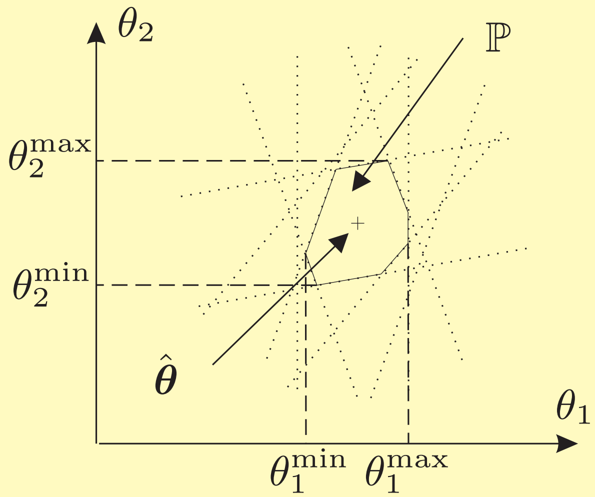


$$\mathbb{S}(k) = \left\{ y(k) - \varepsilon^M(k) \leqslant \boldsymbol{z}^T(k)\boldsymbol{\theta} \leqslant y(k) - \varepsilon^m(k) \right\}$$

$$k = 1, \ldots, n_{\mathcal{U}}$$

- The idea underlying the BEA is to obtain the admissible parameter set:

$$\mathbb{P} = \bigcap_{k}^{n_{\mathcal{U}}} \mathbb{S}(k)$$



The estimate $\hat{\boldsymbol{\theta}}$ can be obtained as follows:

$$\hat{\theta}_i = \frac{\theta_i^{\min} + \theta_i^{\max}}{2}, \quad i = 1, \ldots, n_p,$$

where

$$\theta_i^{\min} = \arg\min_{\theta \in \mathbb{P}} \theta_i, \quad \theta_i^{\max} = \arg\max_{\theta \in \mathbb{P}} \theta_i, \quad i = 1, \ldots, n_p.$$

➜ Model output uncertainty – no error in variables

- The problem of determining model output uncertainty can be solved as

$$\boldsymbol{z}^T(k)\boldsymbol{\theta}^m(k) \leqslant \boldsymbol{z}^T(k)\boldsymbol{\theta} \leqslant \boldsymbol{z}^T(k)\boldsymbol{\theta}^M(k),$$

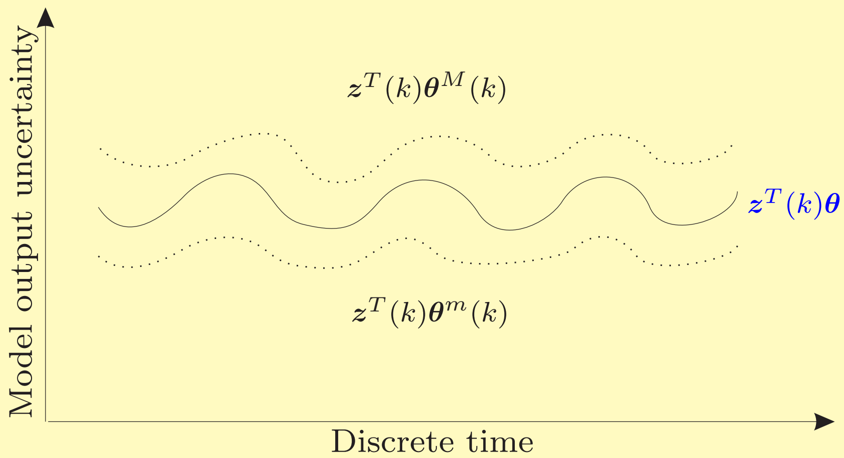
where

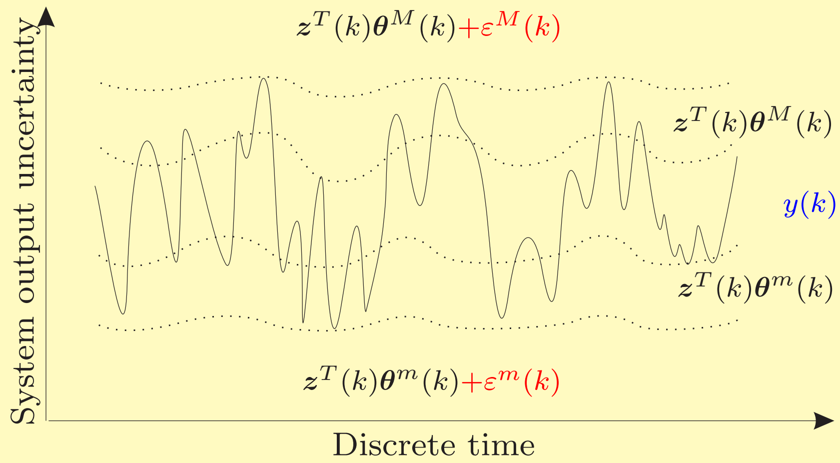
$$\boldsymbol{\theta}^m(k) = \arg\min_{\boldsymbol{\theta}\in\mathbb{V}} \boldsymbol{z}^T(k)\boldsymbol{\theta}, \quad \boldsymbol{\theta}^M(k) = \arg\max_{\boldsymbol{\theta}\in\mathbb{V}} \boldsymbol{z}^T(k)\boldsymbol{\theta}$$

$\mathbb{V}$ – the set of all vertices $\boldsymbol{\theta}^i$, $i = 1, \ldots, n_v$, describing the parameter set $\mathbb{P}$

- The system output will satisfy

$$\boldsymbol{z}^T(k)\boldsymbol{\theta}^m(k)+\varepsilon^m(k) \leqslant y(k) \leqslant \boldsymbol{z}^T(k)\boldsymbol{\theta}^M(k)+\varepsilon^M(k)$$





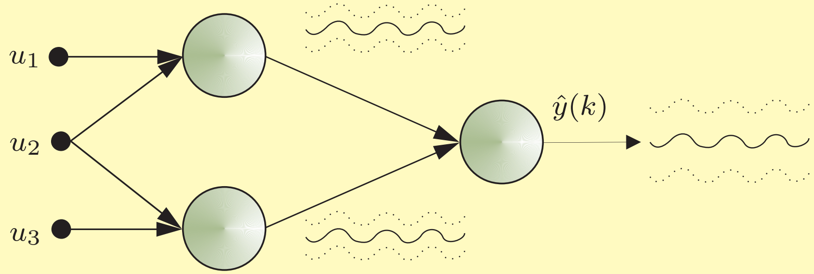
➜ Parameter estimation – an error-in-variables case

- Let us denote an unknown "true" value of the regressor by

$$\boldsymbol{z}_n(k) = \boldsymbol{z}(k) - \boldsymbol{e}(k),$$

where

- $\boldsymbol{z}(k)$ is a known measured value of the regressor
- the error in the regressor is assumed to be bounded as follows:



$$e_i^m(k) \leqslant e_i(k) \leqslant e_i^M(k), \quad i = 1, \dots, n_p.$$

- Space containing parameter estimates:

$$\varepsilon^m(k) - \boldsymbol{e}^T(k)\boldsymbol{\theta} \leqslant y(k) - \boldsymbol{z}(k)^T\boldsymbol{\theta} \leqslant \varepsilon^M(k) - \boldsymbol{e}^T(k)\boldsymbol{\theta}$$

- Bounds depend on the value and sign of each parameter p_i:

$$\theta_i = \theta_i' - \theta_i'', \quad \theta_i', \theta_i'' \geqslant 0$$

$$\varepsilon^m(k) - \left(\boldsymbol{e}^M(k)\right)^T \boldsymbol{\theta}' + \left(\boldsymbol{e}^m(k)\right)^T \boldsymbol{\theta}'' \leqslant y(k) - \boldsymbol{z}^T(k)(\boldsymbol{\theta}' - \boldsymbol{\theta}'') \leqslant \varepsilon^M(k) - \left(\boldsymbol{e}^m(k)\right)^T \boldsymbol{\theta}' + \left(\boldsymbol{e}^M(k)\right)^T \boldsymbol{\theta}''$$

➜ Model output uncertainty – an error-in-variables case

- Model output uncertainty has the following form:

$$y^m(k)(\boldsymbol{\theta}'^m(k), \boldsymbol{\theta}''^m(k)) \leqslant \boldsymbol{z}_n^T\boldsymbol{\theta} \leqslant y^M(k)(\boldsymbol{\theta}'^M(k), \boldsymbol{\theta}''^M(k)),$$

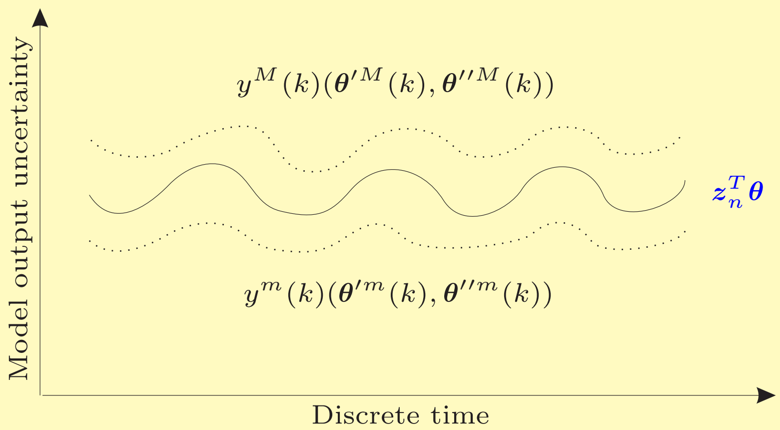
where

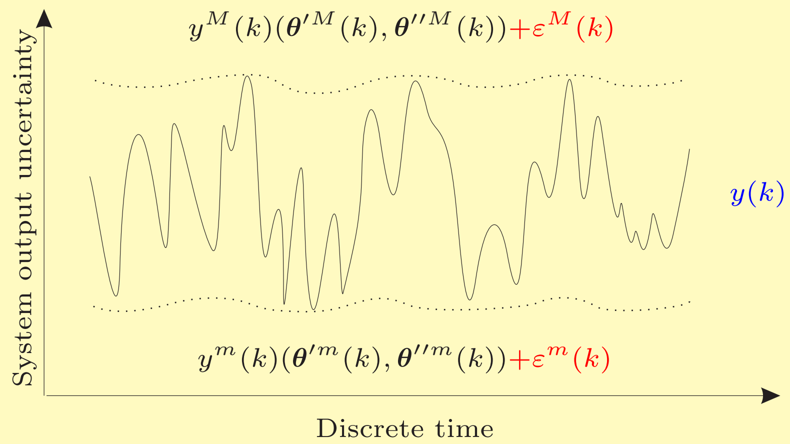
$$y^m(k)(\boldsymbol{\theta}'^m(k), \boldsymbol{\theta}''^m(k)) = \left(\boldsymbol{z}(k) - \boldsymbol{e}^M(k)\right)^T \boldsymbol{\theta}'^m(k) + \left(\boldsymbol{e}^m(k) - \boldsymbol{z}(k)\right)^T \boldsymbol{\theta}''^m(k)$$

$$y^M(k)(\boldsymbol{\theta}'^M(k), \boldsymbol{\theta}''^M(k)) = \left(\boldsymbol{z}(k) - \boldsymbol{e}^m(k)\right)^T \boldsymbol{\theta}'^M(k) + \left(\boldsymbol{e}^M(k) - \boldsymbol{z}(k)\right)^T \boldsymbol{\theta}''^M(k)$$

$$\left(\boldsymbol{\theta}'^m(k), \boldsymbol{\theta}''^m(k)\right) = \arg \min_{(\boldsymbol{\theta}', \boldsymbol{\theta}'')\in\mathbb{V}} y^m(k)(\boldsymbol{\theta}', \boldsymbol{\theta}''(k))$$

$$\left(\boldsymbol{\theta}'^M(k), \boldsymbol{\theta}''^M(k)\right) = \arg \min_{(\boldsymbol{\theta}', \boldsymbol{\theta}'')\in\mathbb{V}} y^M(k)(\boldsymbol{\theta}', \boldsymbol{\theta}''(k))$$





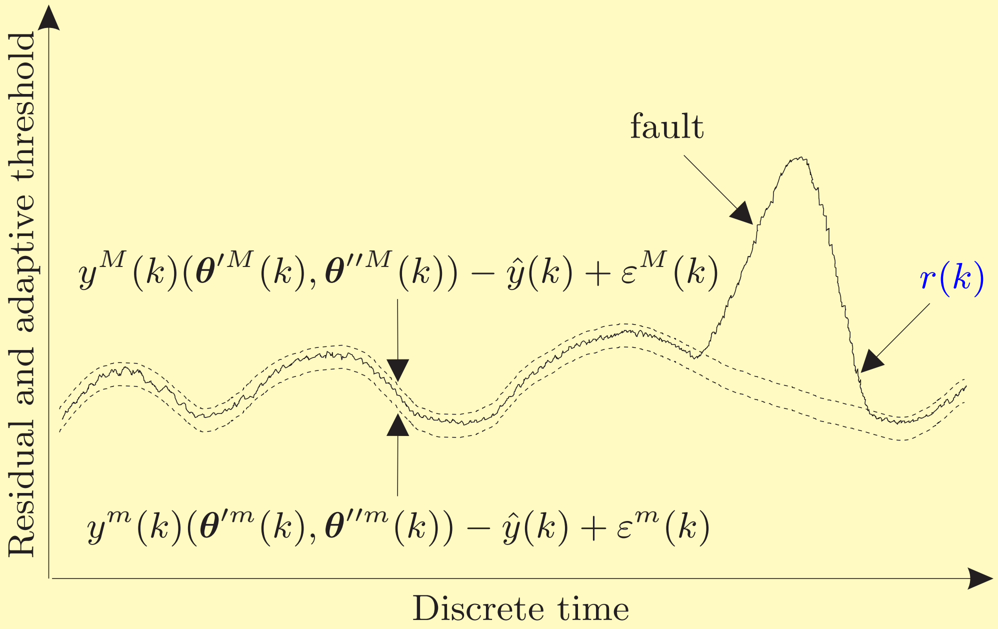
➜ Robust fault detection with a GMDH network

- The residual

$$r(k) = y(k) - \hat{y}(k)$$

- An adaptive threshold

$$y^m(k)(\boldsymbol{\theta}'^m(k), \boldsymbol{\theta}''^m(k)) - \hat{y}(k) + \varepsilon^m(k) \leqslant r(k) \leqslant y^M(k)(\boldsymbol{\theta}'^M(k), \boldsymbol{\theta}''^M(k)) - \hat{y}(k) + \varepsilon^M(k)$$



➜ Illustrative example – robust fault detection with a GMDH

- The data from GARTEUR benchmark were employed to identify the input-output model of the low-fidelity Boening 747-100/200 aircraft model.

- During flight simulation the following pilot inputs were used:

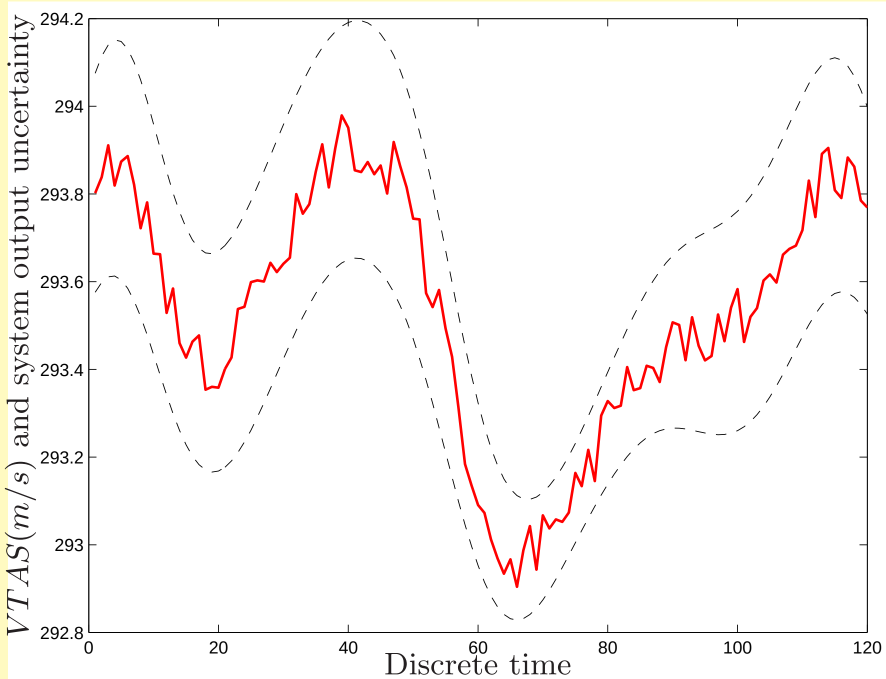
$stab$	stabilizer	Tn_1	Engine 1
δ_w	wheel	Tn_2	Engine 2
δ_p	pedal	Tn_3	Engine 3
δ_c	column	Tn_4	Engine 4

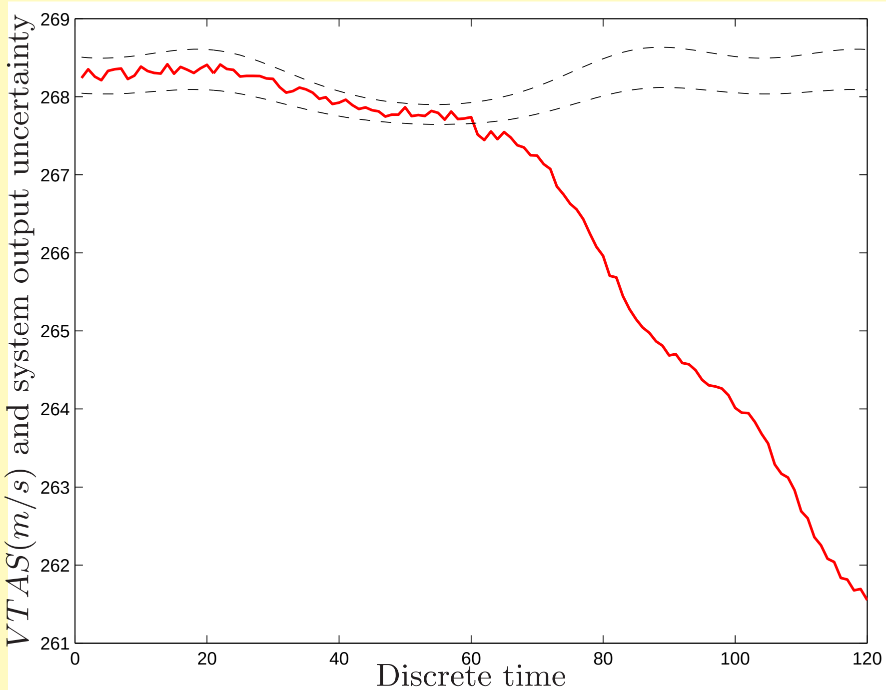
- Low fidelity longitudinal and lateral aircraft states which can be used for fault detection:

q_{body}	Pitch rate	p_{body}	Roll rate
$VTAS$	True air speed	r_{body}	Yaw rate
α	Angle of attack	β	Sideslip angle
θ	Pitch angle	ϕ	Roll angle
he	Altitude	ψ	Yaw angle
xe	x-position	ye	y-position

➜ Illustrative example – robust fault detection with a GMDH

- For the fault detection purpose, a fault scenario containing a wing damage due to engine separation was simulated.

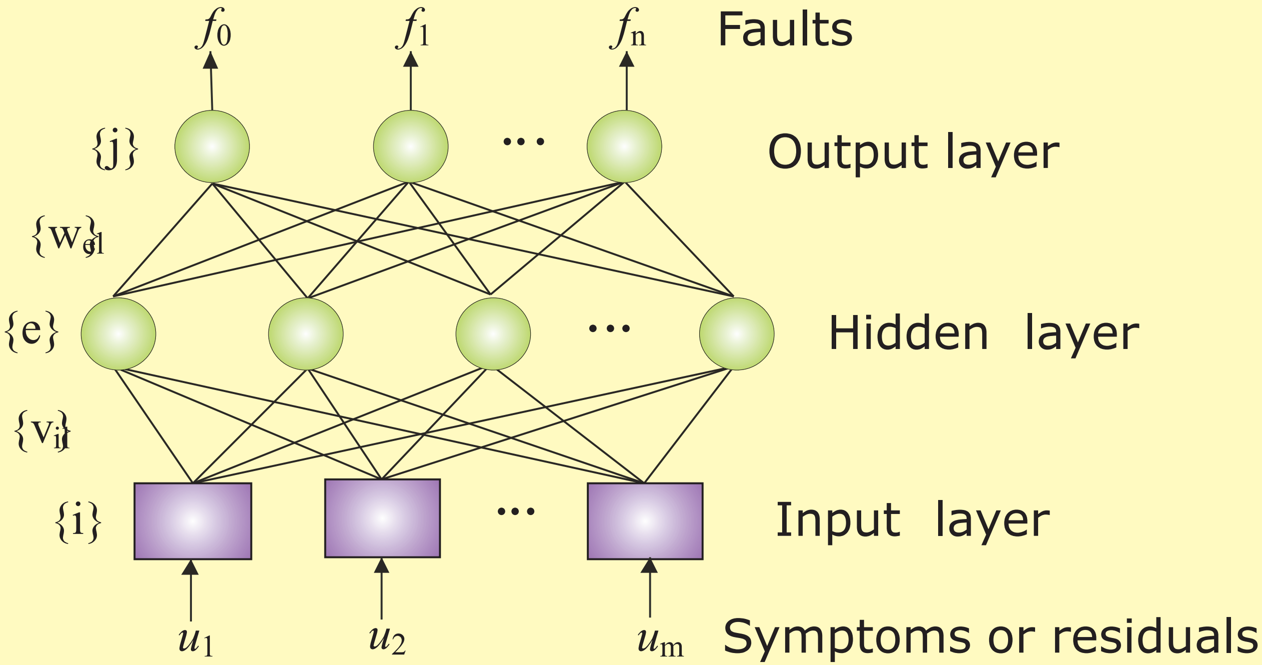




The real system response (true air speed) as well as the corresponding system output uncertainty obtained with the GMDH approach for the fault scenario

☛ ANN-BASED SYMPTOM EVALUATION

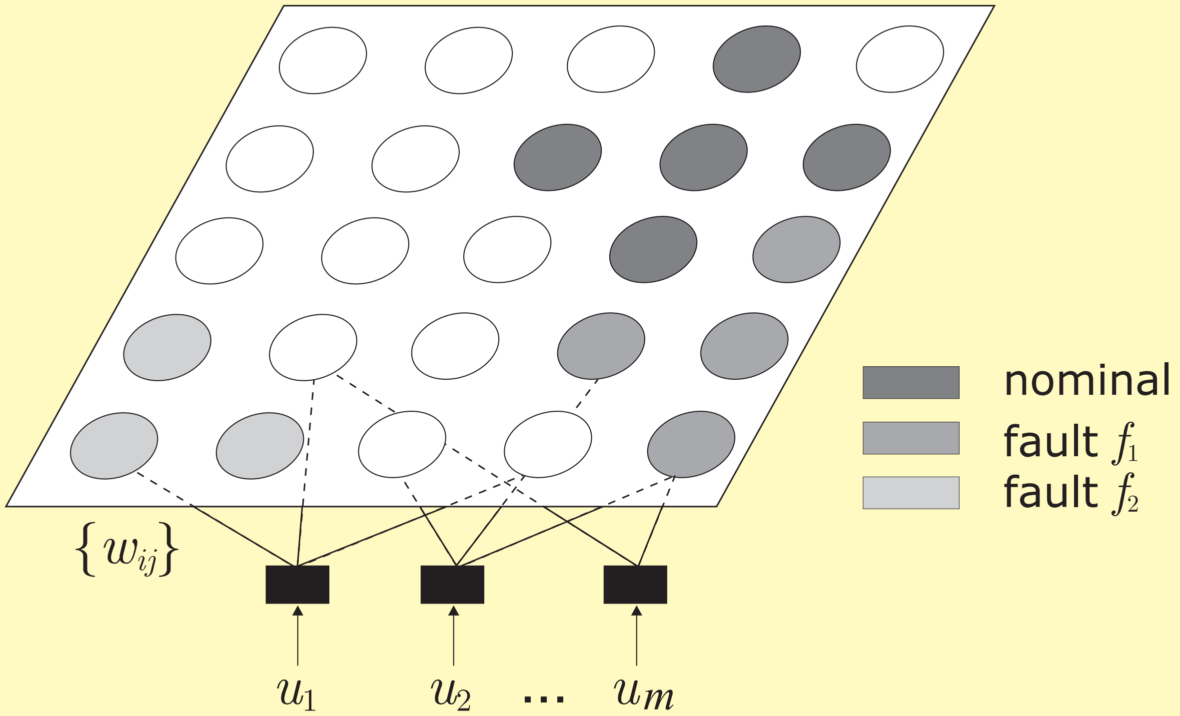
➜ Neural classifiers – a multilayered perceptron



➜ Learning problem

- The standard backpropagation algorithm and its extension can be used.
- The number of potential faults $f_1, f_2, \ldots, f_n$ and a normal state f_0 of the diagnosed process should be selected before designing a network architecture.
- The quality of the neural classifier depends on the quality of learning patterns.

➜ Structure of a two-dimensional self-organizing Kohonen map



➜ Kohonen self-organizing maps

- Unsupervised learning: target categories are developed by the network
- Unsupervised learning extends the capablities of neural networks to pattern recognition tasks where target classifications are not known
- Unsupervised learning schemes are based on the competitive learning principle, i.e. nodes compete with each other to respond to the input pattern (the so-called Winner-Takes-All principle, WTA):

$$\left\|\mathbf{u}(k) - \mathbf{w}_c(k)\right\| = \min_i \left\{ \left\|\mathbf{u}(k) - \mathbf{w}_i(k)\right\| \right\},$$

where

– $\mathbf{u}(k)$ is the input vector

– $\mathbf{w}_c(k)$ is the winner's weight vector

– $\mathbf{w}_i(k)$ is the weight vector of the i-th processing unit

➜ Kohonen learning algorithm

Step 0 : Initialize weights $w_{i,j}$

Step 1 : For each j compute

$$d_j = \sum_{i=1}^{m} (w_{i,j} - u_i)^2$$

Find an index j^* such that d_{j^*} is a minimum

Step 2 : For all units j with a specified neighbourhood of J (the winning node) and for all i

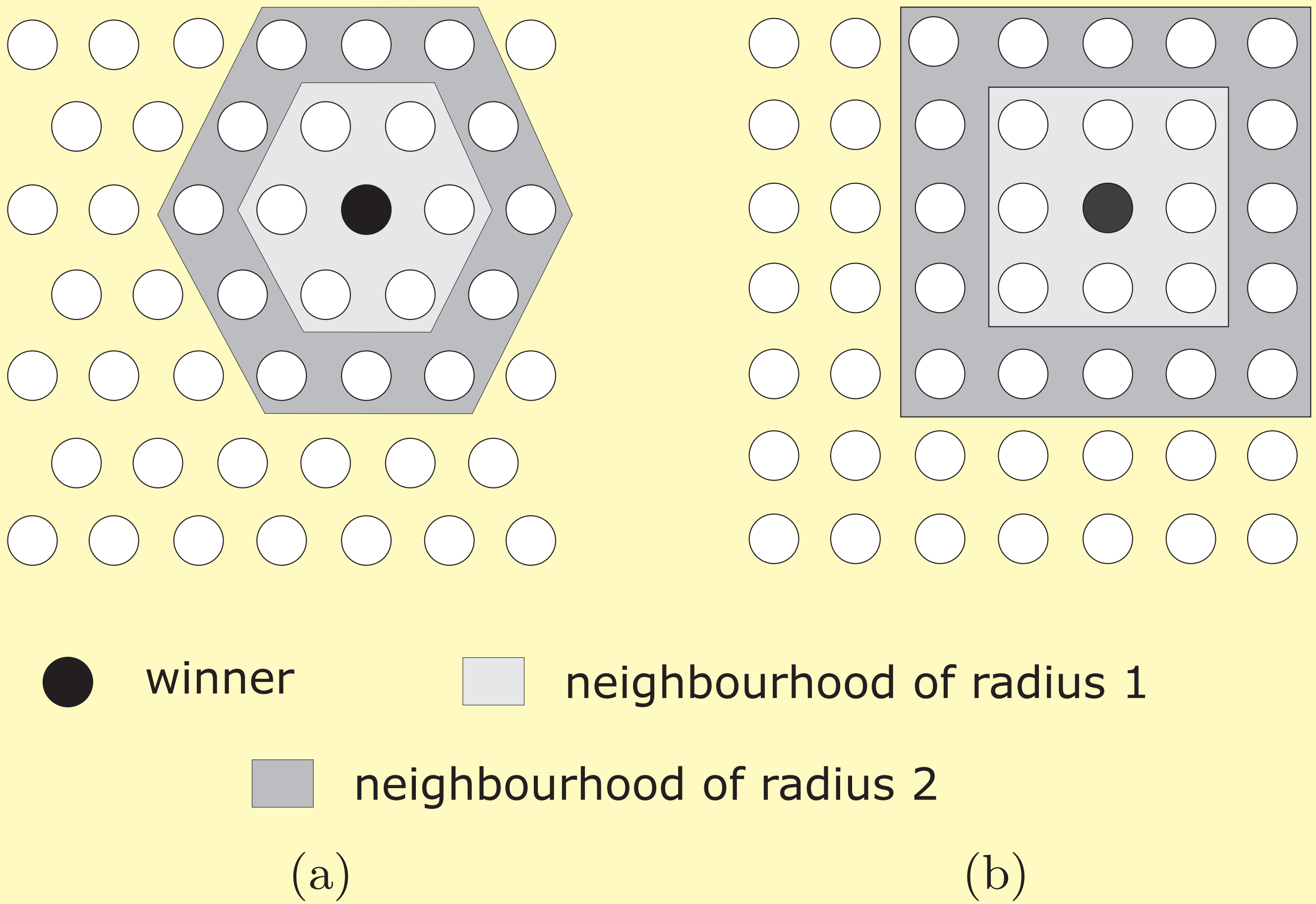
$$w_{i,j}(new) = w_{i,j}(old) + \alpha[u_i - w_{i,j}(old)]$$

- α is called the learning rate. Initially $\alpha_0 \approxeq 0.2$–0.5, but then it decreases as the training proceeds:

$$\alpha_t = \alpha_0(1 - \frac{t}{T})$$

- t is the current training step
- T is the total number of training steps

➜ Winner's neighbourhood: a hexagonal grid (**a**), a rectangular grid (**b**)



➜ Design and learning problems

- In some way the dimension of the Kohonen map depends on the number of selected faults
- The learning quality depends on the quality of patterns concerning each of the faults $f_j, j = 1, 2, ...n$ and the normal state f_0
- Results of learning – separated clusters of neurons are created

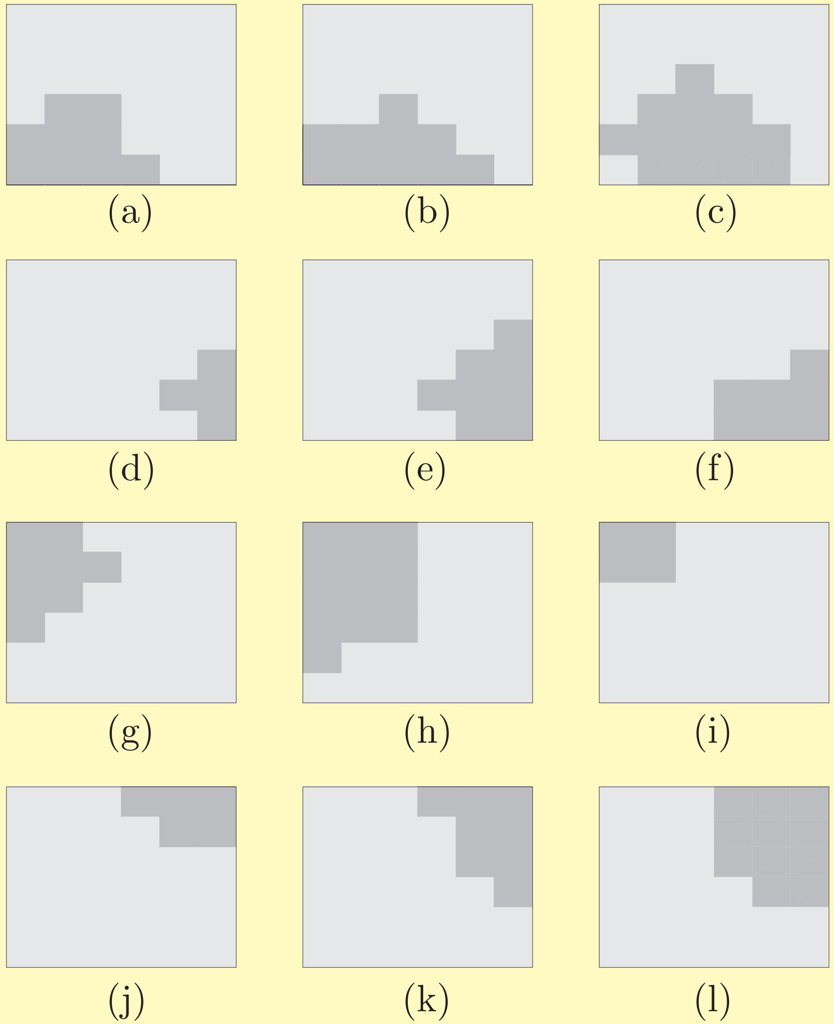
➜ Example of fault evaluation: a two-tank system

- A two-dimensional Kohonen network with the following structure was used:
  - 4 inputs (number of residuals)
  - 49 processing elements (7 neurons by 7 neurons)
- The training set consists of 200 patterns representing 4 process operation conditions – 50 patters for each condition
- A rectangular grid was trained for 20000 steps

Faults	**Fault vector f** $= [f_1\ f_2\ f_3]$		
	f_1	f_2	f_3
normal conditions	0	0	0
valve V2 closed and blocked	1	0	0
valve V2 opened and closed	0	1	0
leak in Tank 1	0	0	1

➜ Example of fault evaluation: a two-tank system

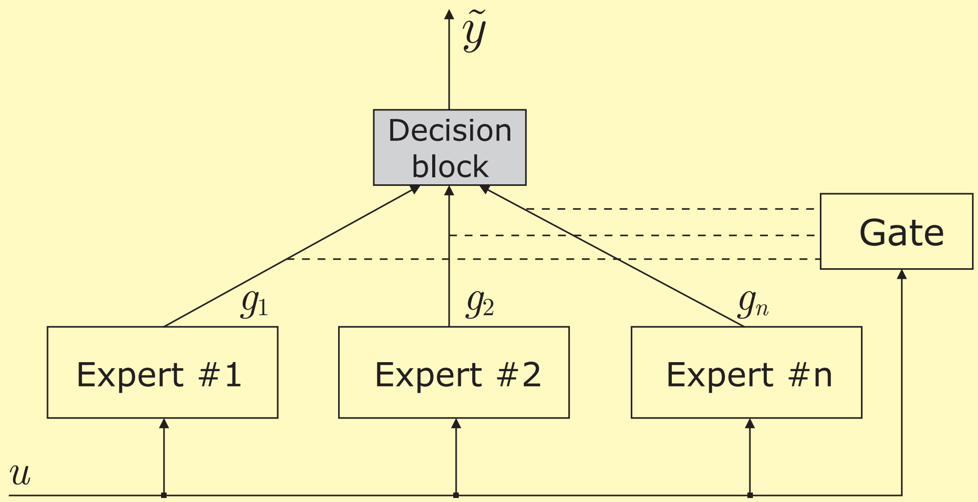
Results generated by the Kohonen network



- Fig. (a,b,c) – nominal operation conditions
- Fig. (d,e,f) – occurrence of the fault f_1
- Fig. (g,h,i) – occurrence of the fault f_2
- Fig. (j,k,l) – occurrence of the fault f_3

➜ Multiple Network Structure (MNS)

- A single ANN of a finite size does not assure the required mapping or its generalisation ability is not sufficient.
- The underlying idea of the MNS is to develop n independently trained ANNs for n working points and to classify a given input pattern by using a decision block.
- A general scheme of the MNS, the so-called *scheme with many experts*:



- The decomposition of a complex classification problem can be performed using independently trained neural classifiers (experts) designed in such a way that each of them is able to recognize only few classes.
- The decision block underdecides which expert should classify a given pattern. This task can be carried out using a suitable rule base in the following form:

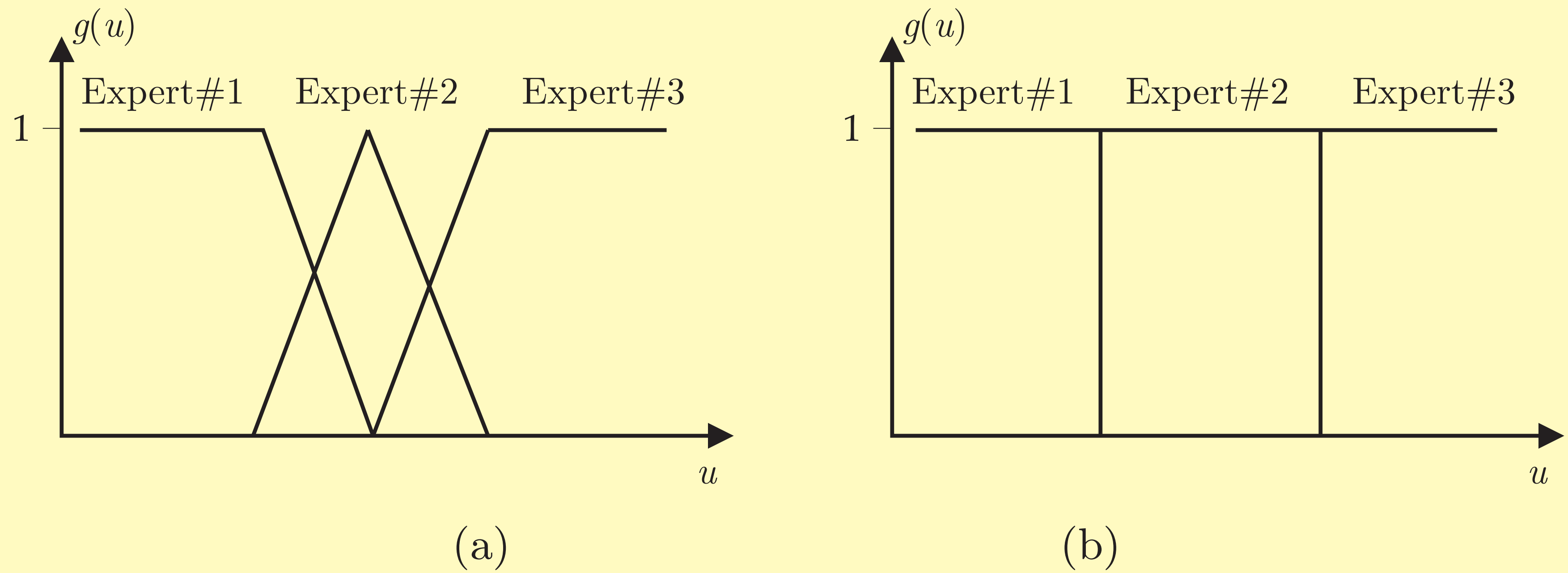
$$\textbf{if}\ \ u \in U_i\ \ \textbf{then}\ \ \text{Expert\#i}, \quad \text{for} \quad i = 1, \ldots, n, \tag{1}$$

  where

  - u is the testing sample
  - U_i is the i-th input subspace.
- The degree of membership of the sample u in a proper subspace can be verified using single features or a set of features.

- Both premises and conclusions of the rules can have crisp or fuzzy values.

  In the case of classical logic, weights assigned to experts have binary representation, and in the case of fuzzy logic they have values from the interval $(0, 1)$.



Membership function distribution: a fuzzy system (a), classical logic (b)

➜ Example of fault evaluation: a two-tank system

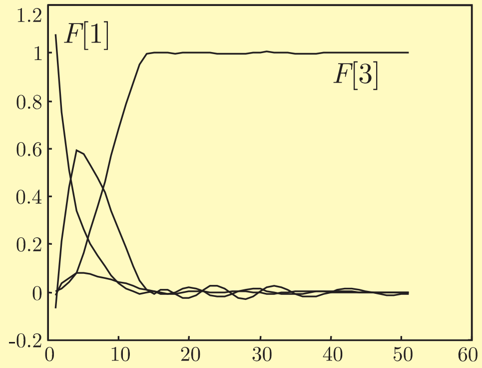
Multiple network structure:

- In order to estimate liquid levels in both tanks, as a feature extractor the ARX estimator was applied.
- Two working points were assumed: levels in Tank $T_1 = 0.5$ and $0.6\,m$.
- Each state of the system was represented by 50 learning patterns.
- The vector of states F consists of the following elements:
  F=[*nominal conditions, leakage, Valve* V_1 *closed and blocked, Valve* V_1 *opened and blocked*].
- Two ANNs were designed and trained for the examined working points.
- The ANNs consist of 90 and 81 hidden neurons.

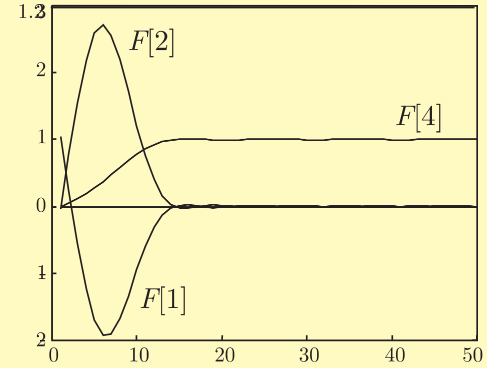
➜ Examples of fault evaluation: a two-tank system

Results generated by the multiple network structure:

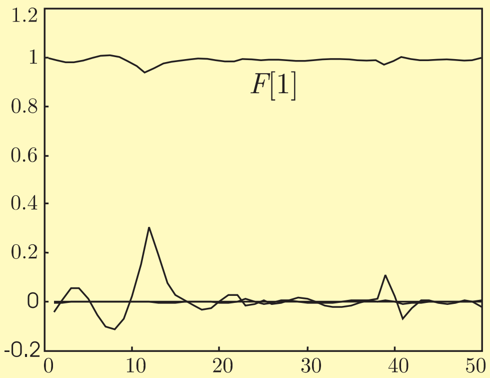
- Fig. (a) – nominal operating conditions
- Fig. (b) – Valve V_1 closed and blocked
- Fig. (c) – Valve V_1 opened and blocked
- Fig. (d) – leakage in Tank T_1
- Fig. (e) – multiple faults: leakage and Valve V_1 closed and blocked



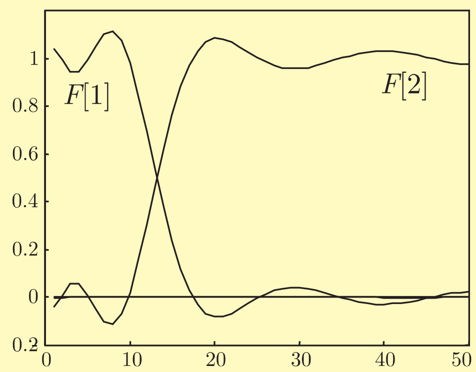
(b)



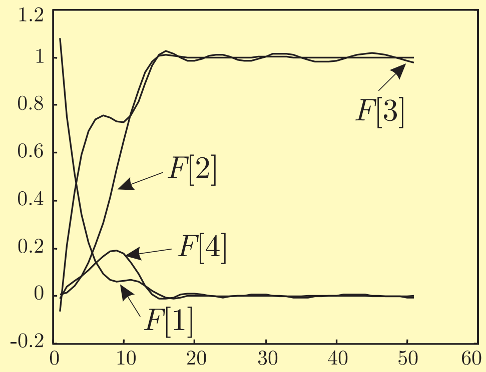
(c)



(a)



(d)



(e)

☛ CONCLUDING REMARKS

- ❐ ANNs are successfully applied to symptom generation and symptom evaluation schemes
- ❐ Application of ANNs to FDI does not require an accurate analytical model of the diagnosed process
- ❐ ANNs provide excellent modelling abilities for dynamic non-linear processes
- ❐ Result of system identification – irrespective of the identification method used there is always the model-reality mismatch
- ❐ The presented ANN-based approaches constitute excellent tools for passive robust fault detection

Thank you