Int. J. Appl. Math. and Comp. Sci., 1999, Vol.9, No.3, 519–546

DYNAMIC NEURAL NETWORKS FOR PROCESS MODELLING IN FAULT DETECTION AND ISOLATION SYSTEMS †

JÓZEF KORBICZ*, KRZYSZTOF PATAN*
ANDRZEJ OBUCHOWICZ*

A fault diagnosis scheme for unknown nonlinear dynamic systems with modules of residual generation and residual evaluation is considered. Main emphasis is placed upon designing a bank of neural networks with dynamic neurons that model a system diagnosed at normal and faulty operating points. To improve the quality of neural modelling, two optimization problems are included in the construction of such dynamic networks: searching for an optimal network architecture and the network training algorithm. To find a good solution, the effective well-known cascade-correlation algorithm is adapted here. The residuals generated by a bank of neural models are then evaluated by means of pattern classification. To illustrate the effectiveness of our approach, two applications are presented: a neural model of Narendra's system and a fault detection and identification system for the two-tank process.

Keywords: fault detection, dynamic neural networks, non-linear modelling, learning algorithms, FL-classifier, two-tank system.

1. Introduction

With increasing demands on the reliability and safety of technical processes, various Fault Detection and Isolation (FDI) approaches have been proposed. The most frequently applied quantitative model-based approach (Chen and Patton, 1999; Gertler, 1999; Isermann, 1997; Korbicz and Cempel, 1993; Mangoubi, 1998; Patton *et al.*, 1989; Pieczyński, 1999) relies on the idea of analytical redundancy that makes use of an analytical mathematical model of the system. However, in practice it is difficult to meet the demands of such a method due to the inevitable model mismatch, noise, disturbances and inherent nonlinearity. In this case, the use of knowledge-model-based techniques, i.e. artificial intelligence (Frank and Köppen-Seliger, 1997) either in the framework of diagnosis expert systems (Fathi *et al.*, 1993; Korbicz *et al.*, 1993) or in

† This work was supported by the European Community under an INCO Copernicus project IQ^2FD and in part by the Polish State Committee for Scientific Research.

* Department of Robotics and Software Engineering, Technical University of Zielona Góra, ul. Podgórna 50, 65–246 Zielona Góra, Poland, e-mail: {J.Korbicz, K.Patan, A.Obuchowicz}@irio.pz.zgora.pl.

combination with a human operator is the only feasible way. Up to now the most known artificial-intelligence-based fault-diagnostic methods and techniques (Jain and Martin, 1999) are fuzzy-logic sets (Frank, 1994; Marcu, 1996), neural networks (Ficola *et al.*, 1997; Koivo, 1994; Korbicz, 1997; Terra and Tinos, 1999) and genetic algorithms for solving various optimization problems in FDI systems (Chen and Patton, 1999; Obuchowicz and Politowicz, 1997). Taking into account the attractiveness and advantages of various analytical and artificial-intelligence-based approaches, in recent years the integration problem of quantitative and qualitative techniques has been studied especially intensively (Korbicz and Patton, 1998). To make the FDI scheme able to integrate quantitative and qualitative information the knowledge-based approach is the most appropriate (Fathi *et al.*, 1993). Recently a novel approach for achieving this, based on B-Spline neural networks, has been proposed by Benkhedda and Patton (1997). Such networks have the ability of neural networks combined with the linguistic description of fuzzy logic. The main disadvantage of the quantitative model-based FDI applications is the requirement that the model of the diagnosed process be as accurate as possible. In this respect, the linear systems theory provides numerous methods in order to get a mathematical model, but they fail where nonlinear processes should be applied.

One of the most important classes of FDI methods especially dedicated for nonlinear processes is the use of artificial neural networks. They are useful when there are no mathematical models of the diagnosed system, so analytical models and parameter-identification algorithms cannot be applied. But a lot of problems occur in identification of dynamic systems, because the known standard neural networks (Freeman and Skapura, 1991) can be applied to model static systems. To overcome this diffulty in FDI and control systems, many solutions have been proposed. In most cases, multilayer perceptron network models with artificially introduced time delay line are studied (Janczak and Korbicz, 1999; Narendra and Parthasarathy, 1990; Zhou and Bennett, 1997). Similar results are obtained by using recurrent neural networks for nonlinear system modelling (Köppen-Seliger and Frank, 1999; Saludes and Fuente, 1999). An alternative neural network architecture for fault detection systems — the GMDH one — is proposed by Korbicz and Kuś (1999). It differs from the known neural network techniques in that its structure is defined during the training process. Moreover, the structure complexity depends on the required accuracy of process modelling (Pham and Xing, 1995). One of the most interesting solutions to the dynamic system identification problem is the application of various dynamic neuron models (Ayoubi, 1994; Kuschewski *et al.*, 1993; Marcu and Mirea, 1997; Patan *et al.*, 1998). The relatively complex dynamic neuron models allow one to design an effective feedforward multilayer network of dynamic neurons with less complex architecture than standard Multi-Layer Perceptrons (MLP) with time delay line (Narendra and Parthasarathy, 1990). Because the dynamic network has a multilayer feedforward architecture, the Extended Dynamic Back-Propagation (EDBP) algorithm can be defined for network training (Ayoubi, 1994; Patan *et al.*, 1998).

The aim of this work is to propose a neural-based FDI system for dynamic nonlinear processes. The main emphasis is placed upon the designing of a neural residual generator using the Back-Propagation Dynamic MLP. Recent investigations in this

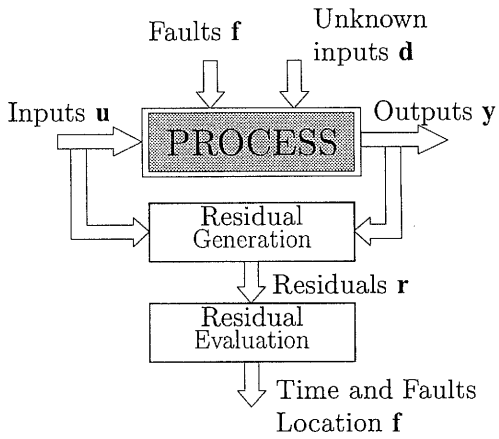

Fig. 1. General scheme for fault diagnosis.

field (Korbicz *et al.*, 1998) have shown that the neural-residual generator based on the dynamic MLP can be an effective tool for fault detection in dynamic systems. However, the choice of a neural network architecture is not a trivial problem. The network architecture of dynamic neurons as well as a set of training algorithm parameters influence the identification quality significantly. The efficiency of structure choice depends on the training process quality. This process seems to be an optimization problem which is intrinsically related to a very rich topology. In fact, two optimization problems are included in the construction of the dynamic network: searching for an optimal network architecture and a network training algoritm. To solve such problems, algorithms of global optimization, e.g. evolutionary algorithms or simulated annealing, should be implemented (Galar, 1995; Michalewicz, 1996). Moreover, the well-known cascade-correlation algorithm (Fahlman and Lebiere, 1990) can be applied to design the dynamic network. The cascade correlation is both an architecture and a supervised learning algorithm for artificial neural networks.

Finally, the effectiveness of our neural dynamic network approach is demonstrated by applying it to modelling Narendra's system (Narendra and Parthasarathy, 1990) and the two-tank system fault diagnosis.

2. Fault Diagnosis Scheme

The fault diagnosis procedure for a dynamic system consists of two separated stages: residual generation and residual evaluation (Fig. 1). In other words, the automatic fault diagnosis can be viewed as a sequential process involving the symptom extraction and the actual diagnostic task. As usual, the residual generation process is based on the comparison between the measured and predicted system outputs. As a result, the difference or so-called residual is expected to be near zero under normal operation but on the occurence of fault a bias from zero should appear. In turn, the residual evaluation module is dedicated to the analysis of the residual signal in order to determine whether a fault has occured and to isolate the fault to a particular system unit.

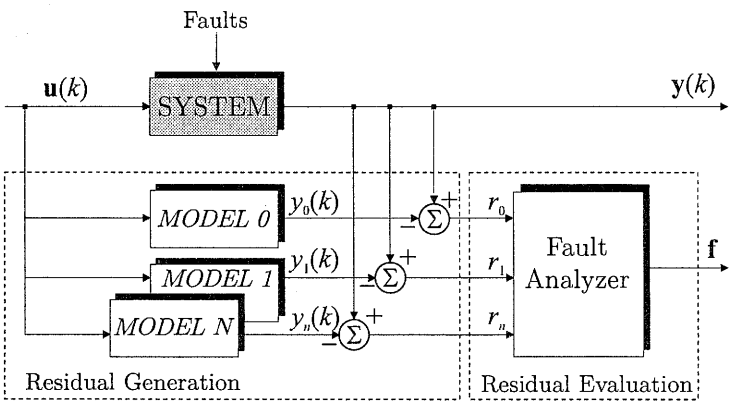

Fig. 2. The FDI scheme based on a neural network model.

One of the most known approaches to residual generation is a model-based concept. In the general case, this concept can be realized using different kinds of models: analytical, knowledge-based and data-based ones (Köppen-Seliger and Frank, 1999). Unfortunatly, the analytical model-based approach is usually restricted to simpler systems described by linear models. When there are no mathematical models of the diagnosed system or the complexity of a dynamic system increases and the task of modelling is hard, analytical models cannot be applied in the FDI system or cannot give satisfying results. In this case data-based models, such as neural networks, fuzzy sets or their combination (neuro-fuzzy networks) can be considered. Figure 2 illustrates how the FDI system is realized through the use of neural networks. In the bank of neural models (*MODEL* 0 ... *MODEL N*) each model should be designed for normal system conditions (*MODEL* 0) and for abnormal ones, i.e. for separated faults (*MODEL* 1 ... *MODEL N*).

Using such a bank of models the residuals can be determined by comparison of the system output $y(k)$ and the output of neural models $y_0(k), y_1(k), \ldots, y_N(k)$. In this way, the residual vector $\boldsymbol{r} = [r_0 \; r_1 \; \ldots \; r_N]$ which characterizes each fault can be obtained. Then the residual vector $\boldsymbol{r}$ is processed by the fault analyzer. Various approaches and techniques (thresholds, statistical and classification methods) can be involved for residual evaluation. Among these approaches the fuzzy and neural classification are attractive from different viewpoints (Köppen-Seliger and Frank, 1999; Marciniak and Korbicz, 1999). The objective of this paper is to design a bank of neural models with dynamic neurons. Moreover, the problem of comparision of fuzzy, neural and analytical classifiers will be studied for the two-tank fault diagnosis system as well.

3. Feedforward Multilayer Network with Dynamic Neurons

The main feature of dynamic Artificial Neural Networks (ANNs) is that they have memory. This means providing the mapping network with dynamic properties that

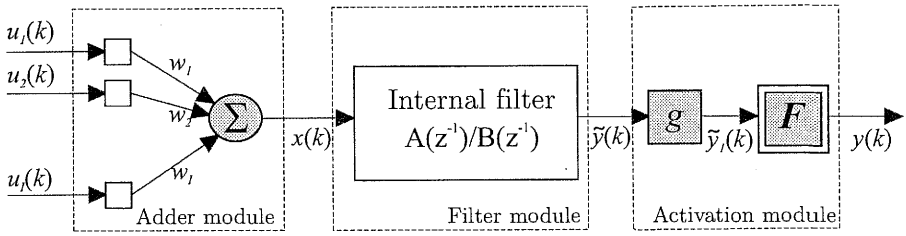

Fig. 3. Dynamic neuron with internal generalized filter.

make it responsive to time-varing signals, i.e. neural networks with internal dynamics. To introduce internal dynamics into the standard MLP network architecture, two solutions can be applied. As the neuron models are static, the introduction of multiple recurrent connections with delays between layers gives us a possiblity to change the static MLP network into a dynamic one (Köppen-Seliger and Frank, 1999; Zhou and Bennet, 1997). An alternative solution can be obtained by designing the MLP network with dynamic neuron models (Ayoubi, 1994; Marcu and Mirea, 1997; Patan and Korbicz, 1997). This approach will be presented below.

3.1. Dynamic Neuron Model

The extended structure of the dynamic neuron model proposed by Ayoubi (1994) is considered here. This structure of the dynamic neuron model is an extention of the static model by adding an Infinite Impulse Response (IIR) filter. Such a model with internal filter introduces some dynamics to the neuron transfer function. Due to the internal filter the general neuron activity depends on its internal states and therefore the neuron processes past values of its own activity $\widetilde{y}(k)$ and inputs $u_i(k)$ for $i = 1, 2, \ldots, I$. As is shown in Fig. 3, the dynamic neuron structure includes three submodules: adder, internal filter and activation module.

The mathematical description of each module is as follows:

Adder

$$x(k) = \boldsymbol{w}^T \boldsymbol{u}(k) = \sum_{i=1}^{I} w_i u_i(k) \tag{1}$$

where $\boldsymbol{w} = [w_1 \ w_2 \ \ldots \ w_I]^T$ denotes the input weight vector, $\boldsymbol{u}(k) = [u_1(k) \ u_2(k) \ \ldots \ u_I(k)]^T$ is the input vector, and $x(k)$ is the result of the weighted summation.

Internal filter

$$\begin{aligned} \widetilde{y}(k) = &-a_1\widetilde{y}(k-1) - \cdots - a_n\widetilde{y}(k-n) + b_0 x(k) + b_1 x(k-1) \\ &+ \cdots + b_n x(k-n) \end{aligned} \tag{2}$$

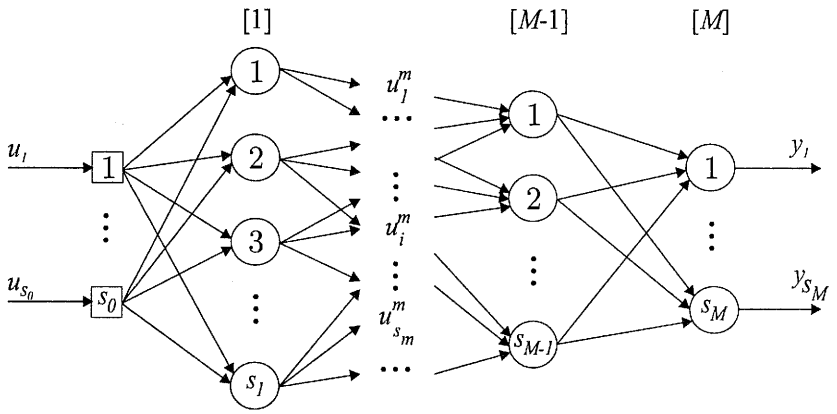

Fig. 4. Dynamic MLP network architecture; □ denotes the input element, ○ stands for the dynamic neuron.

where $x(k)$ and $\widetilde{y}(k)$ are the filter input and output, respectively, k is the discrete time, $\boldsymbol{a} = [a_1 \ \ldots \ a_n]^T$ and $\boldsymbol{b} = [b_0 \ b_1 \ \ldots \ b_n]^T$ are the feedback and feedforward synaptic vector weights, respectively.

Activation module

$$y(k) = F\big(\widetilde{y}_1(k)\big) = F\big(g\widetilde{y}(k)\big) \tag{3}$$

where $F(\cdot)$ is a non-linear activation function that produces the neuron output $y(k)$, and g is the slope parameter of the activation function. In a dynamic neuron, the slope parameter can change its value and therefore can be defined during the learning process similarly to weights $\boldsymbol{w}$, $\boldsymbol{a}$ and $\boldsymbol{b}$.

3.2. Network Architecture

It is well-known that taking into account the dynamic neuron model (1)–(3), one can design a more complex structure, i.e. a neural network. On the other hand, from neural network theory it follows that the crucial problem is the learning algorithm for the new designed network. Therefore, by using the known feedforward MLP network structure, the dynamic MLP can be obtained by replacing the static neurons with dynamic ones (Fig. 4). In comparision with the Elman recurrent networks (Saludes and Fuente, 1999) and the external recurrent MLP (Zhou and Bennett, 1997), the Dynamic MLP (DMLP) under consideration does not require past values of the input vector (process measurements). Owing to internal dynamic neuron properties, the DMLP processes the modelled system measurements at the current time instant k. Therefore the dimension of the network input space is substantially reduced in comparison with the Elman and recurrent MLP networks.

From (Hornik *et al.*, 1989) it is known that multilayer feedforward networks and Radial Basis Function (RBF) networks are universal approximators of static

non-linearities. In turn, Jin *et al.* (1999) proved that the outputs of a class of continuous-time dynamic recurrent networks with an approximate initial stay may be used to approximate uniformly the output trajectories of the considered multi-input and multi-output nonlinear system over finite-time intervals. For the DMLP network considered here it can be proved by applying the Leontaritis and Bilings theorem (1985) that it is a universal identifier. They proved that under some assumptions, any non-linear, discrete and time-invariant system can always be represented by a simplified version of the NARMAX model (Non-linear Auto Regressive Moving Average with eXogenous inputs).

4. Learning Methods and Architecture Optimisation for a Dynamic MLP

Let us consider an M-layered network with dynamic neurons described by differentiable activation functions $F[\cdot]$. In Fig. 4, s_m denotes the number of neurons in the m-th layer and $u_s^m(k)$ is the output of the s-th neuron of the m-th layer at discrete time k ($m = 1, \ldots, M$; $s = 1, \ldots, S_m$). The activity of the s-th neuron in the m-th layer is defined by

$$u_s^m(k) = F\left[g_s^m \left(\sum_{i=0}^{n} b_{is}^m \sum_{p=1}^{S_{m-1}} w_{sp}^m u_p^m(k-i) - \sum_{i=1}^{n} a_{is}^m \widetilde{y}_j^m(k-i) \right) \right] \quad (4)$$

The main objective of the learning process is to adjust all the unknown network parameters $\boldsymbol{v} = [\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}]$, where $\boldsymbol{w} = [w_{sp}^m]_{m=1,\ldots,M;\ s=1,\ldots,S_m;\ p=1,\ldots,S_{m-1}}$ is the weight matrix, $\boldsymbol{a} = [a_{is}^m]_{m=1,\ldots,M;\ s=1,\ldots,S_m;\ i=1,\ldots,n}$ and $\boldsymbol{b} = [b_{is}^m]_{m=1,\ldots,M;\ s=1,\ldots,S_m;\ i=1,\ldots,n}$ are the filter parameter matrices, and $\boldsymbol{g} = [g_{sj}^m]_{m=1,\ldots,M;\ s=1,\ldots,S_m}$ is the slope parameter matrix, based upon a given training set of input-output pairs.

It is well-known that the learning method for the feedforward MLP network is the error backpropagation. However, if an internal recurrence is presented (Fig. 3, Filter module), the localised calculation of the gradient becomes difficult, because the present output of the network $y(k)$ depends on the past outputs $\widetilde{y}(k-1)$, $\widetilde{y}(k-2), \ldots, y(k-n)$. In order to solve this problem, the well-known dynamic backpropagation algorithm (Baldi, 1995) with extension for the network of dynamic neurons (Patan and Korbicz, 1997) will be discussed. Taking into account some drawbacks of such optimization schemes (in the case of feedforward and recurrent neural networks these learning algorithms may get stuck in local minima during the gradient descent (Bianchini *et al.*, 1994)) an alternative approach will be proposed. The Evolutionary Search with Soft Selection (ESSS) algorithm and forced direction of mutation (Obuchowicz and Korbicz, 1998) seems to be a good solution to overcome the problems with the EDBP algorithm.

4.1. Extended Dynamic Back-Propagation Algorithm

In both static and dynamic neural networks, the objective is to determine an adaptive algorithm or a rule which adjust the parameters of the network based on a given set

of input-output pairs. The idea of error backpropagation is widely applied for that purpose in static contexts and with extension to dynamic ones. To define an extended dynamic BP algorithm, the standard approach (Freeman and Skapura, 1991) can be applied. Assuming that unknown parameter vectors $\boldsymbol{w}, \boldsymbol{a}, \boldsymbol{b}$ and $\boldsymbol{g}$ are considered as elements of a parameter vector $\boldsymbol{v}$, the learning process involves the determination of the vector $\boldsymbol{v}^*$ which minimizes the performance index $J_v(k)$ based upon the error function $\boldsymbol{e}(k)$:

$$J_v = \left\|\boldsymbol{e}(k)\right\|^2 = \left\|\boldsymbol{y}^d(k) - \boldsymbol{y}(k)\right\|^2 \tag{5}$$

where $\boldsymbol{y}^d(k)$ is the desired output of the network and $\boldsymbol{y}(k)$ stands for the actual response of the network on the given input pattern $\boldsymbol{u}(k)$.

The adjustment of the parameters of the s-th neuron in the m-th layer according to the EDBP algorithm has the form (Patan *et al.*, 1998)

$$v_s^m(k+1) = v_s^m(k) + \eta \delta_s^m(k) S_{vs}^m(k) \tag{6}$$

where S_{vs}^m is the sensitivity function and

$$\delta_s^m(k) = \begin{cases} e_s(k) F'\left(\widetilde{y}_{1s}^M(k)\right) & \text{for } m = M \\ \displaystyle\sum_{l=1}^{S_{m+1}} \left(\delta_s^{m+1}(k) g_l^{m+1} b_{0l}^{m+1} w_{ls}^{m+1}\right) F'\left(\widetilde{y}_{1s}^m(k)\right) & \text{for } m = 1, \ldots, M-1 \end{cases} \tag{7}$$

In turn, the sensitivity function $S_{vs}^m(k)$ for the elements of the unknown generalised parameter $\boldsymbol{v}$ is defined as follows:

(i) sensitivity with respect to feedback parameter a_{is}^m:

$$S_{a_i s}^m(k) = -g_s^m \widetilde{y}_s^m(k-i) \tag{8}$$

(ii) sensitivity with respect to feedforward parameter b_i:

$$S_{b_i s}^m(k) = g_s^m x_s^m(k-i) \tag{9}$$

(iii) sensitivity with respect to weight parameter w_p:

$$S_{w_p s}^m(k) = g_s^m \left(\sum_{i=0}^{n} b_{is}^m u_p^m(k-i) - \sum_{i=1}^{n} a_{is}^m S_{w_p s}^m(k-i) \right) \tag{10}$$

(iv) sensitivity with respect to slope parameter g_s^m:

$$S_{gs}^m(k) = \widetilde{y}_s^m(k) \tag{11}$$

4.2. Algorithms of Evolutionary Search with Soft Selection and Forced Direction of Mutation

The idea based on soft selection has been applied in a few well-known algorithms, e.g. evolutionary strategies, evolutionary programming, genetic algorithms, simulated annealing, and evolutionary search with soft selection (Galar, 1985; Goldberg, 1989; Michalewicz, 1996).

The ESSS algorithm (Galar, 1985) possesses a good ability of saddle crossing, thus it gives a chance to find a global optimum of the performance index. The efficiency of the ESSS algorithm is poor when the performance index changes in time, as is the case e.g. for the error of DMLP network outputs. In order to overcome these disadvantages, a new algorithm of the so-called ESSS with Forced Direction of Mutation (ESSS-FDM) was proposed by Obuchowicz and Korbicz (1998). The structure and main elements of this algorithm are presented below.

The structure of the ESSS-FDM algorithm

Input data

η is the population size, $t_{\max}$ stands for the maximum number of iterations (epochs), σ denotes the standard deviation of the normal distribution, and μ is the momentum parameter. Moreover, $\Phi : \mathbb{R}^n \to \mathbb{R}_+$ is a non-negative fitness function, n is the number of features, and v_0^0 is the initial point of searching (the set of the DMLP parameters).

Step 1. Initialization

(i) $P(0) = \{v_1^0, v_2^0, \ldots, v_\eta^0\}, \quad (v_k^0)_i = (v_0^0)_i + N(0,\sigma), \quad \begin{matrix} i = 1,2,\ldots,n \\ k = 1,2,\ldots,\eta \end{matrix}$

(ii) $q_0^0 = \Phi(v_0^0)$

Step 2. Repeat

(i) *Estimation*

$$P(t) \to \Phi(P(T)) = \{q_1^t, q_2^t, \ldots, q_\eta^t\}, \quad q_k^N = \Phi(v_k^t), \quad k = 1,2,\ldots,\eta$$

(ii) *Choice of the best element in the history*

$$\{v_0^t, v_1^t, v_2^t, \ldots, v_\eta^t\} \to v_0^{t+1}, \quad q_0^{t+1} = \max\{q_k^t\}, \quad k = 0,1,\ldots,\eta$$

(iii) *Selection*

$$\Phi\big(P(t)\big) \to \{h_1, h_2, \ldots, h_\eta\}, \quad h_k = \min\left\{h : \frac{\sum_{l=1}^{h} q_l^t}{\sum_{l=1}^{\eta} q_l^t} > \zeta_k\right\}$$

where $\{\zeta_k\}_{k=1}^{\eta}$ are random variables uniformly distributed on the interval $[0, 1)$.

(iv) *Modification*

$$P(t) \to P(t+1)$$

$$\big(v_k^{t+1}\big)_i = \big(v_{h_k}^t\big)_i + N(0, \sigma), \quad i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, \eta$$

$$m_i^t = \mu\sigma \frac{E_i\big(P(t)\big) - E_i\big(P(t-1)\big)}{\big\|\big(P(t)\big) - E\big(P(t-1)\big)\big\|}, \quad E_i\big(P(t)\big) = \frac{1}{\eta}\sum_{k=1}^{\eta}(v_k^t)_i$$

where $N(\boldsymbol{m}, \sigma)$ is a normally-distributed random variable with expectation $\boldsymbol{m}$ and standard deviation σ.

Until $t > t_{\max}$.

In the ESSS-FDM procedure, selected vectors (individuals) are mutated by adding to each component a normally-distributed random variable with a given variance σ and an expectation $\boldsymbol{m}^t \neq 0$, in contrast to the ESSS algorithm, where $\boldsymbol{m}^t = 0$. The direction of the vector $\boldsymbol{m}^t$ is parallel to the latest trends of population drift. Similar techniques are known in the backpropagation process (momentum technique) (Freeman and Skapura, 1991). The exogenous parameter μ, which is called the momentum, determines the proportion between the standard deviation σ and the length of the vector $\boldsymbol{m}^t : \mu = \|\boldsymbol{m}^t\|/\sigma$.

4.3. Architecture Optimization

The choice of an optimal architecture for an ANN is an important problem in neural-based FDI systems. Recently, a variety of structure optimisation algorithms have been proposed. In general, they can be divided into three classes (Doering *et al.*, 1997):

i) *bottom-up approaches:* starting with a relatively small structure, these procedures increase the number of hidden neurons and thus the power of the growing network;

ii) *top-down approaches:* these procedures start with a network structure that is supposed to be sufficiently complex to model the relation between input and output variables; after training they try to reduce the number of hidden neurons as much as possible;

iii) *discrete optimisation methods:* each network structure is assigned an evaluation value and the network structure space is searched by a given algorithm.

To design an optimal architecture of a cascade network of dynamic neurons, the well-known cascade-correlation algorithm, i.e. the so-called Fahlman algorithm (Fahlman and Lebiere, 1990), will be adopted here.

4.3.1. Cascade-Correlation Algorithm

The cascade-correlation approach is an architecture design and a supervised learning algorithm for artificial neural networks. Instead of adjusting the parameters of the network of a fixed topology, the basic idea of the cascade-correlation algorithm is to reduce iteratively the error of the output units by inserting hidden units that correlate well (or anti-correlate) with the error signal. By freezing the network while optimizing the new hidden unit candidate the algorithm avoids the moving target problem of the standard BP algorithm. This algorithm realizes simultaneously two optimisation problems related to neural networks: the choice of optimal parameters and architecture design. The network of dynamic neurons obtained is the so-called Cascade Network of Dynamic Neurons (CNDN).

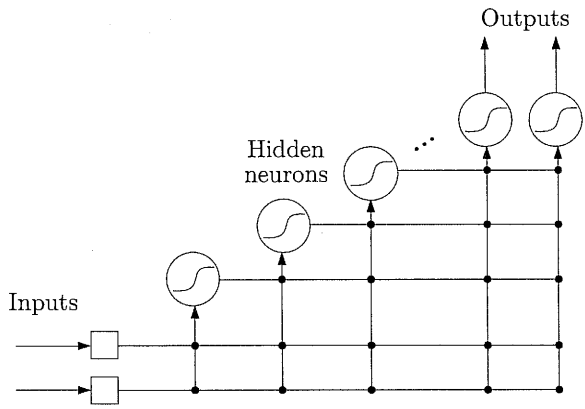

Fig. 5. The cascade network of dynamic neurons.

This algorithm starts without hidden neurons. The direct input-output connections are trained on-line using the gradient descent method. If the network performance is satisfactory, then the procedure is stopped. Otherwise, to reduce the residual errors, new hidden neurons are added to the network. If a neuron is added to the network, its input weights and IIR parameters are frozen, and all the output weights are once trained using the gradient descent method. This procedure repeats until the error is acceptable.

The neuron creation process begins when the candidate receives trainable input connections from all of the external network inputs and from all pre-existing hidden units. The output of this candidate is not connected to the active network yet. The adjusting of the candidate by input weights and its IIR parameters is performed to

maximize the following performance index:

$$\Psi = \sum_{i=1}^{S_M} \left| \sum_{p=1}^{P} (V_p - \overline{V})(E_{pi} - \overline{E}_i) \right| \tag{12}$$

where S_M is the number of output units, P is the number of training patterns, $\overline{V} = (1/P)\sum_{p=1}^{P} V_p$, $\overline{E}_i = (1/P)\sum_{p=1}^{P} E_{pi}$, V_p denotes the response of the candidate to an input u_p and E_{pi} is the output error for u_p. In order to maximize Ψ, the gradient descent method is used.

In Fig. 5 an example of the cascade network with two inputs and two outputs is shown. The small circles denote the adaptable weights between neurons and the rectangles stand for preprocessing inputs. This is a feedforward series-parallel structure. Each neuron receives signals from all inputs and all hidden neurons. Such a structure has some advantages over standard feedforward networks. The first advantage is preventing the moving target problem. This problem often occurs in standard feedforward networks where each neuron adapts its parameters in a constantly changing environment receiving only small input and output network data. In fact, instead of quickly adjusting its parameters, the hidden neurons oscillate around the constantly moving target. In a cascade network each hidden neuron is trained separately. Thus, it receives all input and output learning data and that is why it can adjust its parameters in a correct way. Another advantage is the optimal character of this structure. Hidden neurons are added to the network architecture one by one until the output network error is acceptable. In this way, a neural network which is optimal in the sense of modelling quality can be designed. Let us note that the proposed network is neither optimal in the sense of the number of hidden neurons nor the number of parameters.

5. Fault Analyzer

Another important module in the FDI system under consideration (Fig. 2) is the residual evaluation one. It selects the best fitting group of neural prediction models for either normal operation or one of the learnt faulty situations. This decision-making process can also be considered as a classification problem (Leonhardt and Ayoubi, 1997). For fault classification a few methods can be utilised: geometric, neural and fuzzy ones. All of them use a reference pattern for learning.

5.1. Nearest Neighbour Classifier

This classifier belongs to a simple group of classification algorithms which is based on geometric distance computations. The task is to match each pattern of the residual vector $\boldsymbol{r}_i$, $i = 1, 2, \ldots, P$ with one of the preassigned classes of faults $\boldsymbol{f}_j$, $j = 1, 2, \ldots, N$ to which it has the shortest distance. In a Nearest-Neighbour (NN)-classifier the geometric distance d_i is defined by

$$d_i = \sum_{j=1}^{m} \left(r_j - r_{ij}^{\text{ref}}\right)^2 \tag{13}$$

where $\boldsymbol{r}_i = [r_1 \ r_2 \ \cdots \ r_m]$ is a fault pattern vector, $i = 1, 2, \ldots, N$ indexes each fault, and $\boldsymbol{r}_i^{\text{ref}}$ denotes the reference pattern vector. An NN-classifier works based on the best 'guess' or estimate of the class type, which is obtained from comparing the current query pattern with all the patterns in the knowledge base, and selecting the pattern which is the closest. This classifier works well in many cases and when the class clasters overlap as well. The generation of the NN-algorithm is known as the k-nearest-neighbour algorithm (Looney, 1997).

5.2. Neural Network Classifier

In comparison with a geometrical classifier a more robust decision is achieved using a neural network as the pattern classifier (Marcu and Mirea, 1997; Terra and Tinos, 1999). Before applying the neural network to evaluate the residuals generated by a bank of neural models, the network has to be trained for this task. To do that a residual data base $\boldsymbol{r}_i$, $i = 1, 2, \ldots, N$ and the corresponding fault signature $\{\boldsymbol{r}_i, f_i\}$ data base have to exist.

Various neural network architectures can be applied to the neural classifier realization. One of the most often used neural architectures is the multilayer perceptron (Marcu and Mirea, 1997; Zhou and Bennett, 1997). This neural network with non-linear neurons (e.g. sigmoids) is static, and it is proved to be a universal approximator of any non-linear, but static function. Its training can be performed using the well-known backpropagation algorithm. Such a network maps the patterns from the feature space into a decision space. One of the major difficulties in application of neural networks to fault-detection schemes is the lack of analytical information about the performance, stability and robustness of the network. Other network architectures can be utilised for the classification problem as well. For example, Terra and Tinos (1999) employed an RBF-type network, and the Restricted-Coulomb-Energy (RCE) network was implemented by Köppen-Selliger and Frank (1999).

Taking into account the fact that a common neural network of a finite size does not often load a particular mapping or its generalization is poor, a parallel structure of neural experts was proposed by Marciniak and Korbicz (1999). The basic idea of this approach is to develop $\boldsymbol{n}$ independently trained networks with relevant features to classify a given output pattern by utilising combination methods to decide the collective classification.

5.3. Fuzzy Classifier

According to a general concept (Köppen-Seliger and Frank, 1999; Marcu, 1996), the fuzzy residual evaluation is a process that transforms quantitative knowledge (residuals $r_1, r_2, \ldots, r_N$) into qualitative one (fault indications $f_1, f_2, \ldots, f_N$). The principle of fuzzy residual evaluation consists of three steps: fuzzification, inference and fault indication (defuzzification) (Frank and Köppen-Seliger, 1997).

The fuzzification of the residuals is a mapping of the representation using non-fuzzy values into a representation by fuzzy sets via membership functions. The output is a fuzzy degree of membership, which is always from the interval $[0, 1]$. By using

membership functions, the input space can be divided into a certain number of linguistic variables (e.g. 'small', 'medium', 'Large'). The evolution of membership functions can be assigned on the basis of heuristic process knowledge, statistical distribution functions, subjective knowledge, or by learning with the aid of neural networks.

The task of fault decision is to infer f_i of the set $\boldsymbol{F}$ of the possible faults from a set $\boldsymbol{R}$ of residuals. It is important to note that the residuals r_i, $i = 1, 2, \ldots, N$ are defined by their fuzzy sets, and the relationships between the residuals r_j and the faults f_i in terms of IF/THEN rules can be given. Then the rules described by using the theory of fuzzy logic form a knowledge base of the type (Köppen-Seliger and Frank, 1997)

$$\text{IF} \quad (\text{effect} = r_{j1}) \quad \text{AND} \quad \text{IF} \quad (\text{effect} = r_{j2}) \quad \text{AND} \quad \ldots \quad \text{THEN} \quad (\text{cause} = f_i)$$

where r_{jk}, $k = 1, 2, \ldots, k$ describes the k-th fuzzy set of the j-th residual. Those rules form the knowledge base of the resulting diagnostic expert system. In other words, a mapping of the residuals onto the faults with the aid of the rules in the knowledge base is the main task of the inference procedure.

A proper presentation of a faulty situation to the operator is the next step of the procedure. In general, each fuzzy fault indication signal is a singleton whose amplitude characterizes the degree of membership to only one, preassigned fuzzy set. The representation of a fault indication signal for each fault to the operator in a fuzzy format by the desired degree of membership to the set fault_i, $i = 1, 2, \ldots, n$ is assumed to be given.

6. Applications

In this section, two illustrative examples to show the effectiveness of the proposed approach in the FDI system are considered. The ESSS-FDM algorithm as a training method for a network of dynamic neurons has been implemented in a neural model of Narendra's dynamic system (Narendra and Parthasarathy, 1990). The comparison benchmarks between the dynamic network under consideration and other neural network architectures is presented as well. The next example concerns a laboratory process: a two-tank system. A diagnosing system is designed to detect and evaluate incipient faults. The modelling using the dynamic networks designed by the cascade-correlation procedure is the most important phase in implementation of the proposed fault diagnosis scheme. For fault classification three different techniques have been implemented: analytical, neural networks and fuzzy logic one.

6.1. Narendra's Dynamic System

Assume that the measured process is described by the following difference equation (Narendra and Parthasarathy, 1990):

$$y(k) = f\Big(y(k-1), y(k-2), u(k), u(k-1), y(k-2)\Big) \tag{14}$$

where the non-linear function $f(\cdot)$ is given by

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2} \tag{15}$$

To check the efficiency of the dynamic MLP as a tool for modelling dynamic systems, the network under consideration and the chosen Elman network, along with the recurrent network with external feedbacks, will be compared. The network structures and learning parameters were chosen experimentally. After finishing the learning process, all the networks were tested using the same input given by

$$u(k) = \begin{cases} \sin(2\pi k/250) & \text{for } k < 250 \\ 0.8\sin(2\pi k/250) + 0.2\sin(2\pi k/25) & \text{for } k \geq 250 \end{cases} \tag{16}$$

In order to compare modelling results, let us define the performance index

$$J = \frac{\sum_{p=1}^{P} \left(y^d(p) - y(p)\right)^2}{\sum_{p=1}^{P} \left(y^d(p)\right)^2} \tag{17}$$

where $y^d(p)$ and $y(p)$ denote the desired and network outputs, respectively, and P is the size of the testing inputs.

The test results of both the recurrent and Elman networks are shown in Fig. 6. The system output is marked by a solid line and the neural network output is represented by a dotted line. In Tables 1 and 2 characteristics of both the networks are indicated. In each case the backpropagation algorithm was implemented. The simulation experiment was performed on a PC with Intel PII 333 MHz processor and 128 MB RAM.

Table 1. Characteristics and parameters of the Elman network.

Network characteristics	ELMAN NETWORK		
size of network	1-15-1	1-20-1	1-50-1
number of parameters	61	81	201
number of iterations	20000	20000	20000
learning time	~ 1,05 min	~ 1,12 min	~ 1,64 min
performance index J	0,0781	0,0224	0,0469

Table 2. Characteristics and parameters of the recurrent network with external feedbacks.

Network characteristics	RECURRENT NETWORK WITH EXTERNAL FEEDBACKS		
size of network	5-10-7-1	5-15-10-1	5-20-10-1
number of parameters	145	261	341
number of iterations	20000	20000	20000
learning time	~ 1,51 min	~ 1,63 min	~ 1,62 min
performance index J	0,0371	0,0356	0,0206

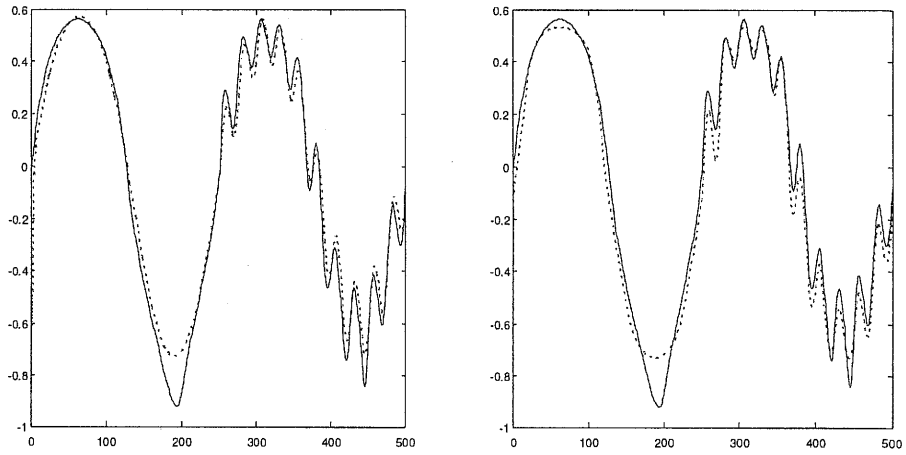

Fig. 6. Outputs of the process and neural model in the testing case: (a) Elman network of class N^2_{1-20-1}, (b) recurrent network with external feedbacks $N^3_{5-20-10-1}$.

To model Narendra's dynamic system, a dynamic MLP network belonging to class N^2_{1-5-1} was chosen (one hidden layer with five neurons). Each neuron contains a second-order IIR filter. Hence, 52 adaptable parameters have to be adjusted in the training process. The parameters of the ESSS-FDM algorithm were as follows: the size of population $\eta = 20$; the momentum $\mu = 0.0545$; the maximum number of iterations $t_{\max} = 5000$; the variance of modification $\sigma = 0.075$ for $t \leq 200$ and $\sigma = 0.015$ for $t > 200$. A set of 5000 training patterns for the on-line training process was generated. Figure 7 shows the system and neural model outputs for different chosen inputs.

The implemented three networks were trained using the on-line backpropagation learning method. It was found that in the case of the recurrent network the learning time was very small ($<$ 2 min) and the quality range of learning, expressed by the performance index J, in the best case was $\sim 0,02$. Experiments showed that it is very hard to train any recurrent network in order to obtain a smaller value of the performance index.

In the case of the dynamic MLP network trained with the ESSS-FDM algorithm, a very good quality ($J = 0.0058$) was obtained. From Fig. 7 it follows that the performance of the system modelling is high for different inputs. Unfortunately, the ESSS-FDM algorithm is more time-consuming than the EDBP one. Based on this approach a good quality model of the dynamic process can be designed. It was found that dynamic MLP network with ESSS-FDM learning algorithm can be applied in the cases where high modelling quality is required and the learning time does not matter. In other cases, the EDBP algorithm is recommended to train the dynamic MLP network. Moreover, it can be seen from Tables 1 and 2 that the size of the recurrent networks is pretty Large and after learning, during normal work the time of identification of these networks will be longer than in the case of the dynamic MLP.

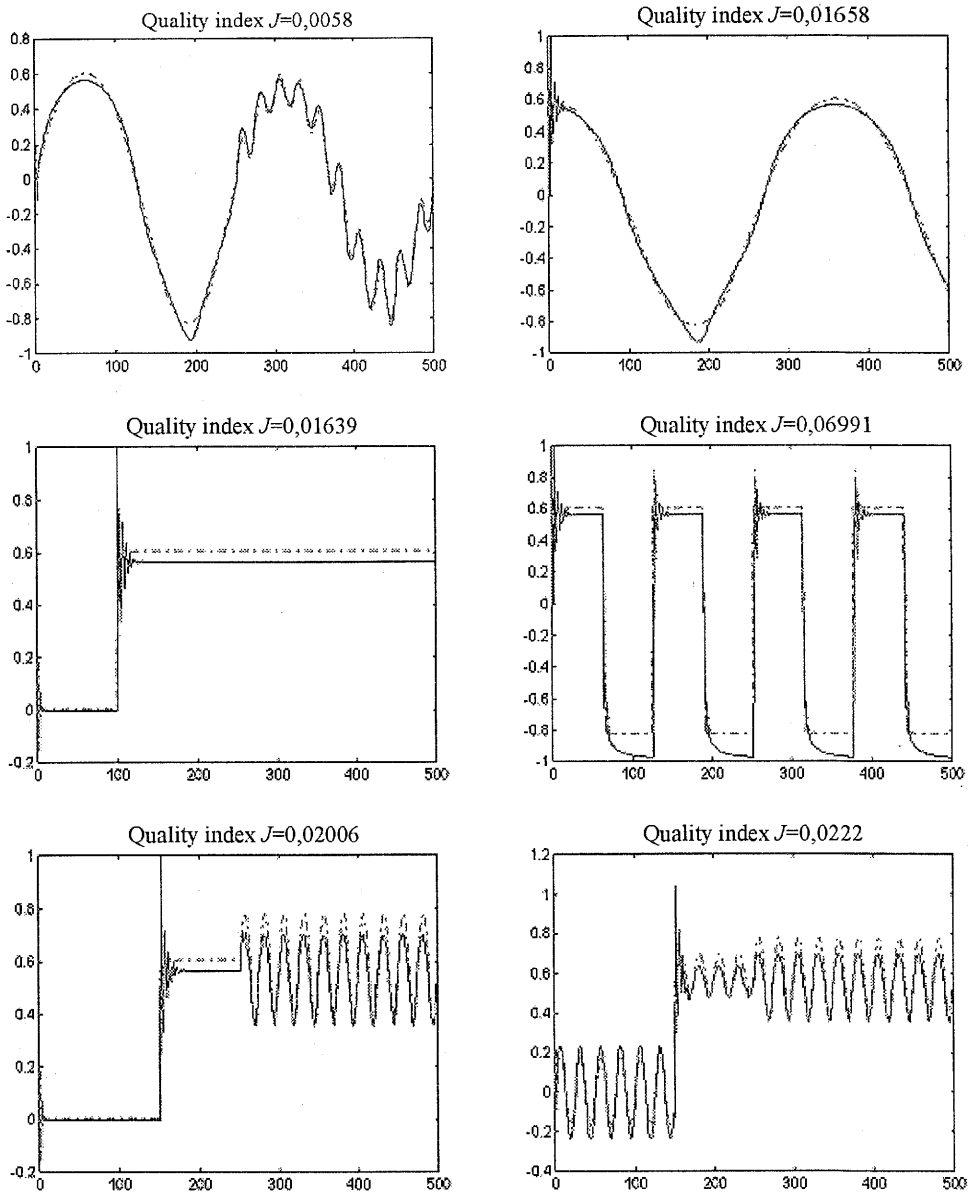

Fig. 7. The system output (solid line) and the DMLP output (dotted line) to a set of chosen inputs.

6.2. Diagnosis of the Two-Tank System

Process Description. The experimental setup *Two-Tank System* consists of two cylindrical tanks with identical cross sections being filled with water and with a delay spiral pipeline (see Fig. 8). The nominal outflow Q_n is located at Tank 2. The pump driven by a DC motor supplies Tank 1, where Q_1 is the inflow of the liquid through pump to Tank 1. Both the tanks are equipped with sensors for measuring the level of the liquid (h_1, h_2). Valves V_1, V_2, V_3, V_4 and V_E are electronic switching ones. The aim of the two-tank system control is to keep up the water level in Tank 2 constant.

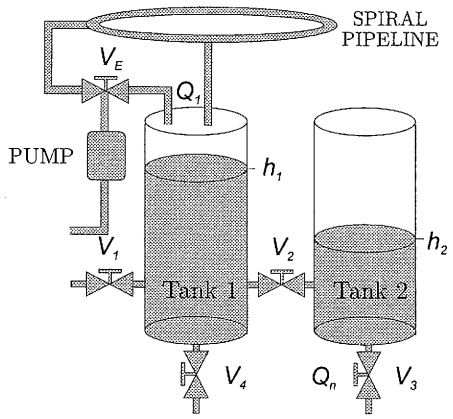

Fig. 8. Two-tank system with a delay spiral pipeline.

The connecting pipeline and tanks are additionally equipped with manual adjustable valves and outlets to simulate clogs and leaks. Four classes of process behaviour were taken into consideration in this experiment. They are as follows: f_0 – normal behaviour, f_1 – valve V_2 is closed and blocked, f_2 – valve V_2 is open and blocked, and f_3 – a leakage in Tank 1.

Residual Generation. In the proposed FDI system four classes of system behaviour (normal condition f_0 and three faults f_1–f_3) are modelled by a bank of neural observers designed by the cascade-correlation algorithm. *MODEL* 0 reflects the system under normal conditions and *MODEL* 1, *MODEL* 2 as well as *MODEL* 3 express faults f_1, f_2 and f_3, respectively.

The characteristics of the neural models are presented in Table 3. The notation N_{v-1} means that the cascade-correlation network consists of v hidden neurons and one output neuron. Good modelling results for relatively small network sizes were obtained.

Based on cascade-correlation network observers the residual signals were generated (see Fig. 9). Each model was sensitive to one of the system behaviour class and generated a residual signal near zero. The value of the residual r_{f_0} (assigned to the normal condition) significantly differs from zero when one of the faults occurred

Table 3. Characteristics of the neural models.

NETWORK PARAMETERS	CLASS OF THE SYSTEM BEHAVIOUR			
	Normal conditions	Fault No.1	Fault No.2	Fault No.3
Name of the network	*MODEL* 0	*MODEL* 1	*MODEL* 2	*MODEL* 3
Size of the network	N_{3-1}	N_{2-1}	N_{2-1}	N_{2-1}
Order of the filter	first	first	second	first

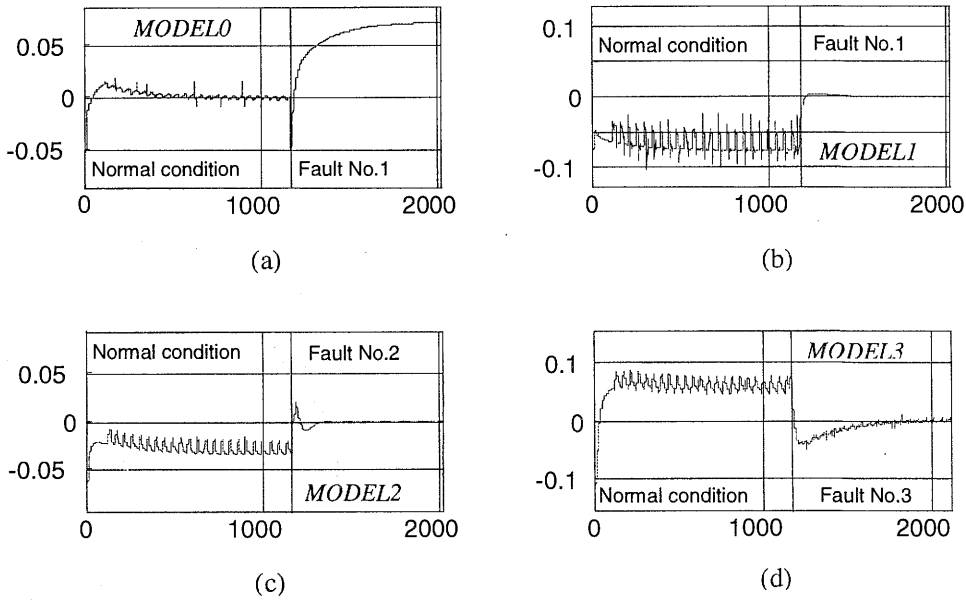

Fig. 9. Residuals behaviour for different cases.

(Fig. 9(a)). During our experiment, at the beginning the diagnosed system worked properly and only after 1000 iterations some faults occurred. The residual r_{f_i} assigned to the fault f_i generates a deviation from zero under the normal operation of the system and is almost zero when f_i occurs (Figs. 9(b)–9(d)).

Residual Evaluation. The residuals should be transformed into the fault vector $\boldsymbol{f}$ to detect and locate the faults. This operation can be seen as a classification problem. Each pattern of the symptom vector $\boldsymbol{r} = [r_0\ r_1\ r_2\ r_3]$ is assigned to one of the classes of system behaviour $\{f_0,\ f_1,\ f_2,\ f_3\}$. Such a decision can be achieved using various pattern classifiers based on artificial intelligence techniques (neural networks and fuzzy logic) and on the analytical approach (nearest neighbour).

Neural Classifier. In fact, the neural classifier should map a relation in the form $\Psi : \mathbb{R}^n \to \mathbb{R}^m : \Psi(\boldsymbol{r}) = \boldsymbol{f}$. This is a static mapping and a multilayer feedforward network can be successfully applied in the residual evaluation process. Each bit of

the created binary vector $\boldsymbol{f}$ is sensitive to a particular fault. The coding of the faults is shown in Table 4. The representations of the fault vector $\boldsymbol{f}$ presented in Table 4 match all the known kinds of system conditions. In the case of normal operation the classifier should generate $\boldsymbol{f} = [0 \quad 0 \quad 0]$. Any other vector representation (e.g. $\boldsymbol{f} = [1 \quad 1 \quad 1]$) shows that the system operates under unrecognized conditions. To perform residual evaluation, the well-known static multilayer feedforward network can be used. The applied network belongs to class $N^3_{4,5,4,3}$. The training process was carried out off-line for 20000 steps using the backpropagation algorithm with momentum and adaptive learning rate. The learning data consist of 200 patterns – fifty patterns per one class of system behaviour. The activation functions in the hidden layers were of the hyperbolic tangent type and in the output layer of the linear type. Taking into account the fact that one of the assumptions was a binary response of the classifier (Table 4), its output should be close to the nearest integer, while the classifier output differs from the desired value less than the assumed accuracy called the *confidence level* (Fig. 10).

Table 4. Coding of the faults.

Kind of fault	Fault vector $\boldsymbol{f} = [f_1 \quad f_2 \quad f_3]$		
	f_1	f_2	f_3
Normal conditions	0	0	0
Valve V_2 closed and blocked	1	0	0
Valve V_2 opened and blocked	0	1	0
Leak in Tank 1	0	0	1

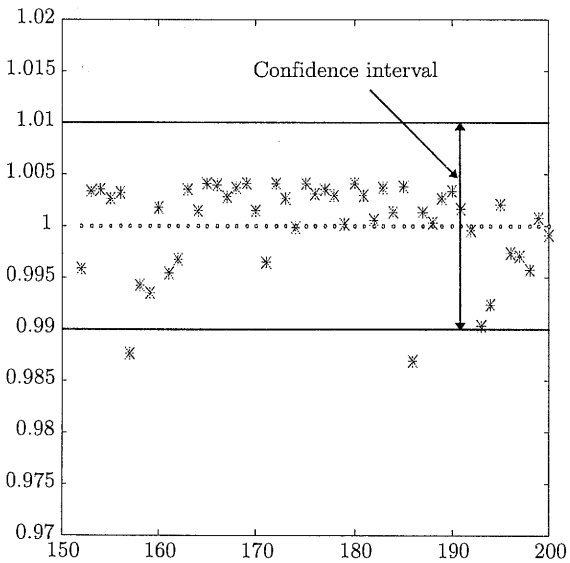

Fig. 10. Rounding of the classifier output.

Table 5 illustrates the relation between the value of the confidence level and the percentage of non-classified and badly classified system behaviours. This result relates to the case of detection of f_1.

Table 5. Relation between the confidence level and non-classified and badly classified system behaviours.

Confidence level	Non-classified behaviours	Badly classified behaviours
0.1	7.8 %	15.4 %
0.05	10.5 %	14.1 %
0.02	13.9 %	12.1 %
0.01	16.8 %	10.3 %
0.004	21.9 %	8.1 %
0.001	97.7 %	2.2 %

The applied classifier indicates a simple structure and a non-complicated training algorithm. However, it has one main disadvantage: it needs rounding the classifier output with a fixed confidence level. This leads to the undesirable effect of non-classified system behaviour. In fact, alongside with increasing the confidence level, the number of non-classified behaviours is decreased, but badly classified ones are increased too. Summarizing, such a classifier should be used in the case when each single system behaviour is assigned to a suitable set of process behaviours.

Nearest Neighbour Classifier. This analytical approach works by saving the set of exemplars or prototype patterns which were already classified in a data set. For this technique, our 'knowledge' is the set of labelled paterns denoted by $\{r_p, f_p\}_{p=1}^{P}$. In the case of a diagnostic system, the 'knowledge' base consists of 200 patterns, fifty patterns for each class of system behaviour (the same data set is in the case of the neural classifier). The output of the NN-classfier was labelled as follows: 1 – normal conditions, 2 – occurence of f_1, 3 – occurence of f_2, and 4 – occurence of f_3. In Fig. 11 classification results are presented for the case when the NN-classfier was applied.

In all simulations, faulty situations occur after about 1030 time steps. Table 6 presents the percentage of badly classified system behaviours by the NN-classifier.

Table 6. Percentage of badly classified system behaviours.

Class of the system behaviour	Badly classified behaviours
normal conditions	1.56 %
fault f_1	11.3 %
fault f_2	19.6 %
fault f_3	7.7 %

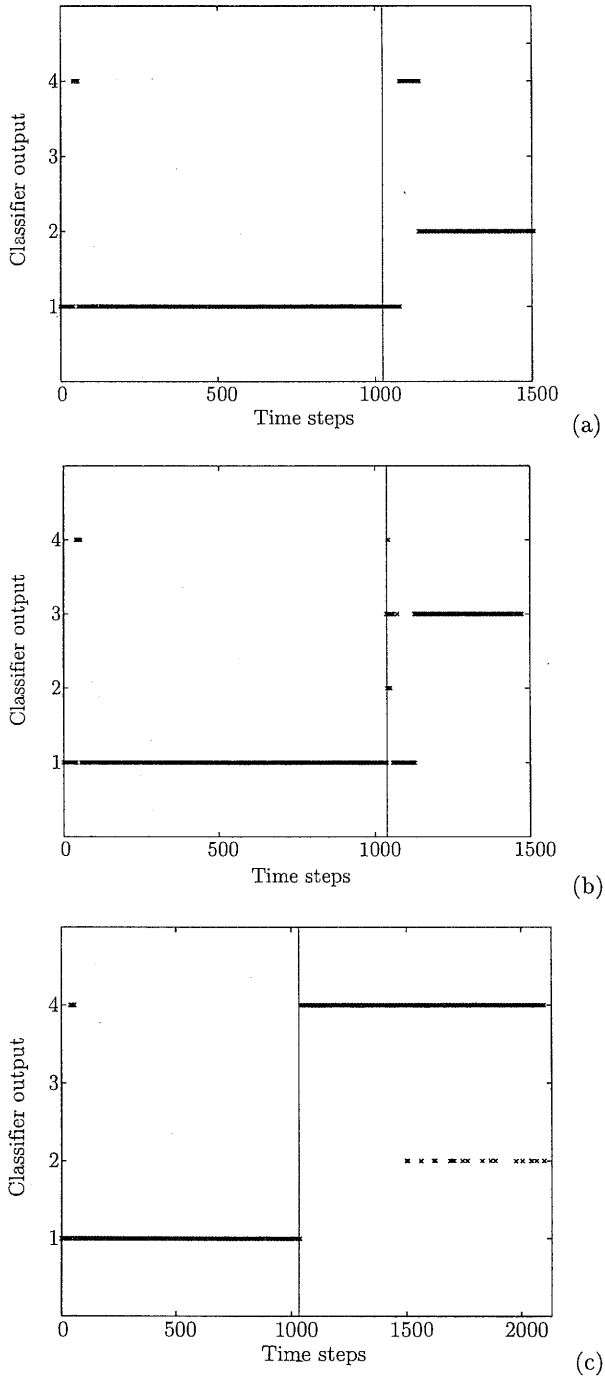

Fig. 11. Three faulty situations and their classification by the NN-classifier: (a) occurence of f_1, (b) occurence of f_2, and (c) occurence of f_3.

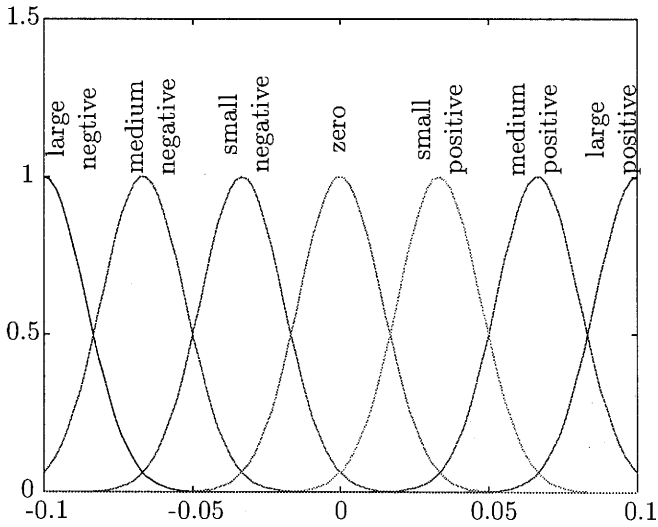

Fig. 12. Placement of the membership functions for the input.

As can be seen from Tables 5 and 6, the NN-classifier is more efficient than the neural network one. Moreover, any behaviour is ascribed to one of the known classes of system behaviours. Therefore, this classifier has another advantage over the neural one. It excludes the occurence of non-classified system behaviours. In spite of that, the number of badly classified behaviours is still considerable.

Fuzzy Logic Classifier. In our diagnostic system the Fuzzy Logic (FL) classifier has four inputs (residual signals) and three outputs (number of faults). Each input is divided into seven linguistic variables described by suitable membership functions (Fig. 12). Each linguistic variable is labelled by terms 'Large negative', 'medium negative', 'small negative', 'zero', 'small positive', 'medium positive' and 'Large positive'. All the membership functions were true Gaussian ones. In turn, the output variables were divided into two linguistic variables called 'healthy' and 'faulty' (see Fig. 13). The fuzzy inference system returns outputs of the range from 0 % to 100 %. Taking into account courses of the residual signals, fuzzy logic rules can be determined. An example of such a rule related to the diagnostic system under consideration is presented below:

R_1: IF *residual* 0 is zero AND *residual* 1 is small negative
AND *residual* 2 is medium negative AND *residual* 3 is medium positive
THEN *output* 1 is healthy, *output* 2 is healthy, *output* 3 is healthy

The defuzzification operation was carried out using the midst of the maximum method. The results obtained using the FL-classifier are presented in Fig. 14. As is seen from Fig. 14, this method recognizes faulty situations very well. For example, let us consider the behaviour of the system presented in Fig. 14(a) after 2000 time steps. The output vector [70 %, 30 %, 30 %] is obtained. This means that the probability

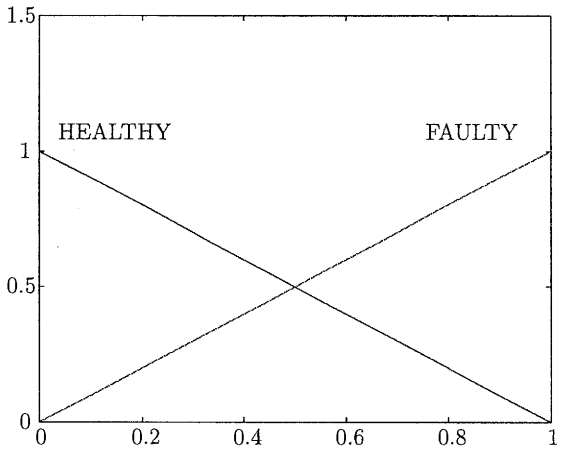

Fig. 13. Placement of the membership function for output.

of Fault 1 is 70 %, probability of Fault 2 is 30 %, and probability of Fault 3 is 30 %. With certainty we can confirm that fault f_1 occurs in the system. Using fuzzy reasoning better results than in the case of the NN-classifier can be obtained. A Large improvement is seen especially in the case of fault f_3. It spite of a high efficiency of the FL-classifier it is necessary to notice that the choice of the number of linguistic variables, as well as generating fuzzy rules, is not a trivial problem. Therefore, to solve this problem, one can combine the linguistic ability of fuzzy logic with the learning ability of neural networks. Thus, a neuro-fuzzy classifier can be obtained.

7. Conclusions

This paper has presented a scheme for fault detection and diagnosis in dynamic nonlinear systems. The key task in implementation of this scheme is to obtain the neural network of dynamic neurons with internal feedbacks. The cascade-correlation algorithm has been used for the network architecture and parameter allocation. This algorithm is more effective than the extended dynamic backpropagation one, which usually gets stuck in unsatisfactory local minima of the error function, and is not time-consuming unlike algorithms of global optimization. The inconvenience of the cascade-correlation application is related to the fact that the cascade architecture of the dynamic network is not optimal in the sense of the number of hidden neurons. The redundancy of the network parameters can lead to a high dependence between the realized network function and the actual realization of the training set.

The residuals generated by using a bank of dynamic neural nets are classified by three types of classifiers: neural, fuzzy and nearest neighbour ones. The study demonstrates that dynamic neural networks provide an efficient tool for system modelling and identification. In turn, for pattern recognition in the sense of residual evaluation, fuzzy logic is a more efficient tool.

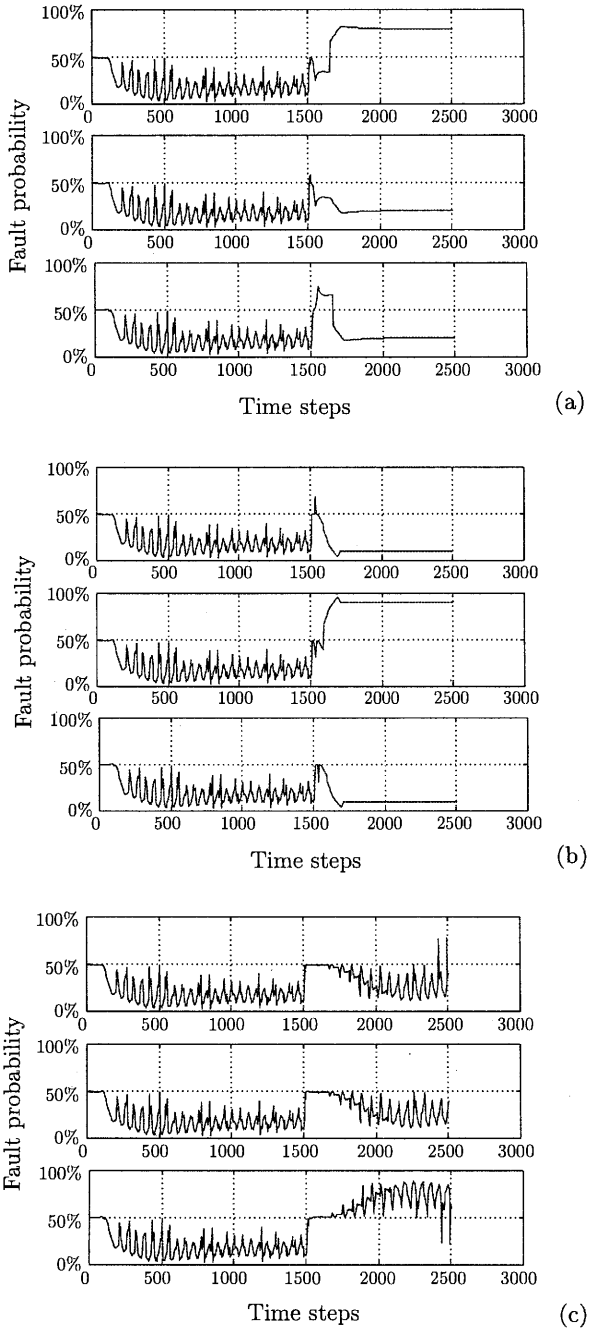

Fig. 14. Three faulty situations and their classification by the FL-classifier: (a) occurence of f_1, (b) occurence of f_2, and (c) occurence of f_3.

Further research will be focused on investigation of the behaviour of the proposed fault detection system in the case when more than one fault or combination of faults occur, e.g. valve V_2 is blocked in a certain position. Moreover, another classifier structure, i.e. neuro-fuzzy one, will be tested too to improve recognition quality of the FDI system.

References

Ayoubi M. (1994): *Fault diagnosis with dynamic neural structure and application to a turbocharger.* — Proc. IFAC Symp. *Fault Detection, Supervision and Safety for Technical Process SAFEPROCESS'94*, Espoo, Finland, Vol.2, pp.618–623.

Baldi P. (1995): *Gradient descent learning algorithm overview. A general dynamical systems perspective.* — IEEE Trans. Neural Networks, Vol.6, No.1, pp.182–195.

Benkhedda H. and Patton R.J. (1997): *Information fusion in fault diagnosis based on B-spline networks.* — Prep. IFAC Symp. *Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'97*, Hull, UK, Vol.2, pp.681–686.

Bianchini M., Gori M. and Maggini M. (1994): *On the problem of local minima in recurrent neural networks.* — IEEE Trans. Neural Networks, Vol.5, No.2, pp.167–177.

Chen J. and Patton R.J. (1999): *Robust Model-Based Fault Diagnosis for Dynamic Systems.* — Berlin: Kluwer Academic Publishers.

Doering A., Galicki M. and Witte H. (1997): *Structure optimization of neural networks with A*-algorithm.* — IEEE Trans. Neural Networks, Vol.8, No.6, pp.1434–1445.

Fahlman S.E. and Lebiere C.D. (1990): *The cascade-correlation learning architecture*, In: Advances in NIPS2 (D. Toutetzky, Ed.). — CA: Morgan kaufmann, San Mateo, pp.524–532.

Fathi Z., Ramirez W.F. and Korbicz J. (1993): *Analytical and knowledge-based redundancy for fault diagnosis in process plants.* — AIChE J., Vol.39, No.1, pp.42–56.

Ficola A., La Cava M. and Magnino F. (1997): *An approach to fault diagnosis for nonlinear dynamic systems using neural networks.* — Prep. IFAC Symp. *Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'97*, Hull, UK, Vol.1, pp.365–370.

Frank P.M. (1994): *Fuzzy supervision. Application of fuzzy logic to process supervision and fault diagnosis.* — Proc. Int. Workshop *Fuzzy Technologies in Automation and Intelligent Systems, Fuzzy Duisburg'94*, Duisburg, Germany, pp.36–59.

Frank P.M. and Köppen-Seliger B. (1997): *New developments using AI in fault diagnosis.* — Eng. Applic. Artif. Intell., Vol.10, No.1, pp.3–14.

Freeman J.A. and Skapura D.M. (1991): *Neural Networks. Algorithms, Applications, and Programming Techniques.* — New York: Addison-Wesley Publishing Company.

Galar R. (1985): *Handicapped individua in evolutionary processes.* — Biol. Cybern., Vol.51, pp.1–9.

Gertler J. (1999): *Fault Detection and Diagnosis in Engineering Systems.* — New York: Marcel Dekker, Inc.

Goldberg D.E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning.* — Reading, Massachusetts: Addison-Weley Publishing Company, Inc.

Hornik K., Stinchcombe M. and White H. (1989): *Multilayer feedforward networks are universal approximators.* — Neural Networks, Vol.2, No.5, pp.359–366.

Isermann R. (Ed.) (1997): *Supervision, Fault Detection and Diagnosis of Technical Systems.* — Special Section *Control Engineering Practice*, Vol.5, No.5.

Jain L.C. and Martin N.M. (1999): *Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms.* — New York: CRC Press.

Janczak A. and Korbicz J. (1999): *Neural network models of Hammerstein systems and their application to fault detection and isolation.* — Proc. 14th IFAC *Triennial World Congress*, Beijing, China, Vol.P, pp.85–90.

Jin L., Gupta M.M. and Nikiforuk P.N. (1999): *Dynamic recurrent neural networks for approximation of nonlinear systems.* — Proc. 14th IFAC *Triennial World Congress*, Beijing, China, Vol.K, pp.45–50.

Koivo H.N. (1994): *Artificial neural networks in fault diagnosis and control.* — Control Eng. Practice, Vol.2, No.7, pp.89–101.

Korbicz J. (1997): *Neural networks and their application in fault detection and diagnosis.* — Prep. IFAC Symp. *Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'97*, Hull, UK, Vol.1, pp.377–382.

Korbicz J. and Cempel Cz. (Eds.) (1993): *Analytical and Knowledge-Based Redundancy in Fault Detection and Diagnosis.* — Special Issue Appl. Math. Comp. Sci., Vol.3, No.3.

Korbicz J. and Kuś J. (1999): *Dynamic GMDH neural networks and their application in fault detection systems.* — Proc. *European Control Conference, ECC'99*, Karlsruhe, Germany, (accepted).

Korbicz J. and Patton R.J. (Eds.) (1998): *Integration of Quantitative and Qualitative Fault Diagnosis Methods within the Framework of Industrial Application.* — Proc. *Copernicus IQ2FD Workshop*, Kazimierz Dolny, Poland.

Korbicz J., Obuchowicz A. and Patan K. (1998): *Network of dynamic neurons in fault detection systems.* — Proc. IEEE Int. Conf. *System, Man and Cybernetics*, San Diego, USA, pp.1862–1867, CD ROM No.98CH36218.

Korbicz J., Uciński D., Pieczyński A. and Marczewska G. (1993): *Knowledge-based fault detection and isolation systems for power plant.* — Appl. Math. Comp. Sci., Vol.3, No.3, pp.613–630.

Köppen-Seliger B. and Frank P.M. (1999): *Fuzzy logic and neural networks in fault detection*, In: Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms (L.C. Jain and N.M. Martin, Eds.). — New York: CRC Press, pp.169–209.

Kuschewski J.G., Hui S. and Żak S. (1993): *Application of feedforward neural network to dynamical system identification and control.* — IEEE Trans. Contr. Syst. Technol., Vol.1, No.1, pp.37–49.

Leontaritis I.J. and Bilings S.A. (1985): *Input-output parametric models for nonlinear systems.* — Int. J. Control, Vol.41, pp.303–344.

Looney G.G. (1997): *Pattern Recognition Using Neural Networks.* — Oxford: Oxford University Press.

Marciniak A. and Korbicz J. (1999): *A multiple network structure for fault diagnosis of nonlinear systems.* — Proc. 4th Conf. *Neural Networks and Their Applications*, Zakopane, Poland, pp.431–436.

Marcu T. (1996): *Pattern recognition techniques using fuzzly labeled data for process fault detection.* — Appl. Math. Comp. Sci., Vol.6, No.4, pp.815–840.

Marcu T. and Mirea L. (1997): *Robust detection and isolation of process faults using neural networks.* — IEEE Control Systems, October, pp.72–79.

Mangoubi R.S. (1998): *Robust Estimation and Failure Detection.* — London: Springer-Verlag.

Michalewicz Z. (1996): *Genetic Algorithms + Data Structure = Education Programs.* — Berlin: Springer-Verlag.

Narendra K.S. and Parthasarathy K. (1990): *Identification and control of dynamical system using neural networks.* — IEEE Trans. Neural Networks, Vol.1, No.1, pp.12–28.

Obuchowicz A. and Korbicz J. (1998): *Evolutionary search with soft selection and forced direction of mutation.* — Proc. 7th Int. Symp. *Intelligent Information Systems*, Malbork, Poland, pp.300–309.

Obuchowicz A. and Politowicz K. (1997): *Evolutionary algorithms in optimization of a multilayer feedforward neural network architecture.* — Proc. 4th Int. Symp. *Methods and Models in Automation and Robotics, MMAR'97*, Międzyzdroje, Poland, Vol.2, pp.739–743.

Patan K. and Korbicz J. (1997): *Dynamic neural network with filters of different orders.* — Proc. 4th Int. Symp. *Methods and Models in Automation and Robotics, MMAR'97*, Międzyzdroje, Poland, Vol.2, pp.745–750.

Patan K., Obuchowicz A. and Korbicz J. (1998): *Network of dynamic neurons approach to residual generators.* — Proc. 5th Int. Symp. *Methods and Models in Automation and Robotics, MMAR'98*, Międzyzdroje, Poland, Vol.2, pp.645–650.

Patton R.J., Frank P.M. and Clark R.N. (Eds.) (1989): *Fault Diagnosis in Dynamic Systems. Theory and Application.* — New York: Prentice Hall.

Pham D.T. and Xing L. (1995): *Neural Networks for Identification, Prediction and Control.* — Berlin: Springer-Verlag.

Pieczyński A. (1999): *Computer Diagnostic Systems of Industrial Processes.* — Zielona Góra: Technical University Press, (in Polish).

Saludes S. and Fuente M.J. (1999): *Neural-networks-based fault detection and accomodation in a chemical reactor.* — Proc. 14th IFAC *Triennial World Congress*, Beijing, China, Vol.P, pp.169–174.

Terra M.H. and Tinos R. (1999): *Fault detection and isolation in Puma 560 manipulator via neural networks.* — Proc. 14th IFAC *Triennial World Congress*, Beijing, China, Vol.P, pp.175–180.

Zhou J. and Bennett S. (1997): *Dynamic system fault diagnosis based on neural network modelling.* — Prep. IFAC Symp. *Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS'97*, Hull, UK, Vol.1, pp.54–59.

Received: 14 January 1999
Revised: 17 August 1999