FAULT DETECTION AND ISOLATION FOR DYNAMIC PROCESSES USING RECURRENT NEURAL NETWORKS

Piotr PRZYSTAŁKA

Silesian University of Technology
Department of Fundamentals of Machinery Design
18a Konarskiego Str., 44-100 Gliwice, pp@polsl.pl

Summary

The paper focuses on the problem of fault detection and isolation for dynamic processes using selected recurrent neural networks. The main objective is to show how to employ some discoveries of the chaos theory for modeling processes by means of globally and locally recurrent neural networks. Both types of neural models are used in fault detection and isolation block. The performance of the FDI system is examined using two types of neural models: Jordan/Elman tower neural networks and networks with dynamic neural units. The paper contains numerical examples that illustrate the merits and limits of these two approaches.

Keywords: fault detection and isolation, Jordan and Elman neural networks, locally recurrent neural networks, chaos theory.

DETEKCJA I LOKALIZACJA USZKODZEŃ PROCESÓW DYNAMICZNYCH Z UŻYCIEM SIECI REKURENCYJNYCH

Streszczenie

Treść artykuł wiąże się z problemem detekcji i lokalizacji uszkodzeń dla szerokiej gamy procesów dynamicznych z użyciem wybranych rekurencyjnych sieci neuronowych. Głównym celem jest pokazanie w jaki sposób mogą zostać zastosowane niektóre z odkryć teorii chaosu do modelowania procesów z użyciem globalnych i lokalnych struktur neuronowych. Oba typy modeli neuronowych zostały użyte w bloku detekcji i lokalizacji uszkodzeń. Sprawność układu diagnostycznego porównana została dla modeli procesów z zastosowaniem: sieci wielo-kontekstowych Jordana/Elmana i sieci z neuronami dynamicznymi. W artykule zamieszczono przykłady numeryczne wskazujące na zalety i wady obu podejść.

Słowa kluczowe: detekcja i lokalizacja uszkodzeń, sieci Jordana i Elmana, sieci lokalnie rekurencyjne, teoria chaosu.

1. INTRODUCTION

The increasing complexity of technical means is known to be one of the most important problems in the modern control system design and analysis. Chemical refineries, electrical furnaces, aircrafts and even small inspection robots are complex systems and in some cases cannot be precisely described by classical mathematical models [3, 9, 10]. On the other hand, modern technical systems can be controlled despite faults affecting its components [2]. Due to these facts, fault diagnosis methods and fault-tolerant control design using soft computing techniques are gaining more and more attention in recent years [7, 8].

There are a large number of real-world control problems that can not be solved by conventional (hard) computing methods and it is well-founded to use the human mind-based reasoning which is included in soft computing [23]. Nevertheless, soft computing is not always a sufficient solution for our task. In numerous industrial applications hard computing plays a major role. As it can be observed, soft computing techniques are usually combined with traditional hard computing approaches in industrial products instead of using them separately [7].

In recent years, artificial neural networks, especially recurrent ones have attracted considerable research interest in the fields of control and diagnostic systems [1, 9]. On the one hand, results and data from industrial applications confirm human safety and economic efficiency of such approaches [7, 10], but on the other, there is still the need to elaborate much more general neural models that might be used for modeling both deterministic and stochastic processes simultaneously [8, 17].

Chaos together with the theory of relativity and quantum mechanics is considered as one of the three monumental discoveries of the twentieth century [24]. The peculiarities of chaotic systems can be given as follows: strong dependence of the behavior on initial conditions, the sensitivity to the changes of system parameters, presence of strong harmonics in the signals, fractional dimension of space state trajectories, at least one positive Lyapunov exponent

that characterizes the rate of separation of infinitesimally close trajectories [24].

The paper contains numerical examples illustrating the performance of recurrent neural networks in two kinds of problems: time-series prediction and fault diagnosis of a simplified three-tank benchmark system. The first task is to create a neural model of a process given by the differential equations. The main objective is to show how to employ some discoveries of the chaos theory for training considered neural networks. In this paper, two parts of this approach are used: applications of some chaotic systems to improve learning algorithms for recurrent neural networks and phase-space reconstruction as the first step in the modeling of dynamic processes. The second task is to build the fault detection and isolation block for a benchmark problem based on recurrent neural networks. Finally, the last section summarizes the important features of these approaches and the paper concludes with the future work.

2. RECURRENT NEURAL NETWORKS

Generally speaking, recurrent neural networks can be viewed as universal approximators for spatio-temporal data [11, 12]. They can be classified into two categories: globally (totally or partially) recurrent and locally recurrent networks. In the first class, there are structures with feedback connections between simple static neurons of different layers or/and these of the same layer. The second one encompasses neural structures similar to static feedforward topologies, however they include dynamic neural units with internal feedback connections [9].

2.1. Jordan/Elman tower neural networks

Jordan and Elman networks were first introduced in [4, 5] and were classified as partially recurrent networks [9, 19]. These structures can learn sequential patterns and are usually employed to learn the grammatical structure of a set of sentences. However there are also their applications in the domain of system identification theory [18]. Jordan/Elman tower networks used herein differ from standard Jordan/Elman topologies in that they may have more than one state vector (Fig. 1). It is well-founded that such multi-state architectures have better learning abilities on certain tasks than standard Jordan/Elman ones of similar weight complexity [19, 20].

Outputs of the considered types of Jordan and Elman networks are defined as follows:

$$\mathbf{y}(k) = \mathbf{LW}^2\boldsymbol{\sigma}^1\left(\mathbf{IW}^1\mathbf{u}(k) + \mathbf{p}(k) + \mathbf{b}^1\right) + \mathbf{b}^2 \quad (1)$$

a) for multi-context Jordan networks:

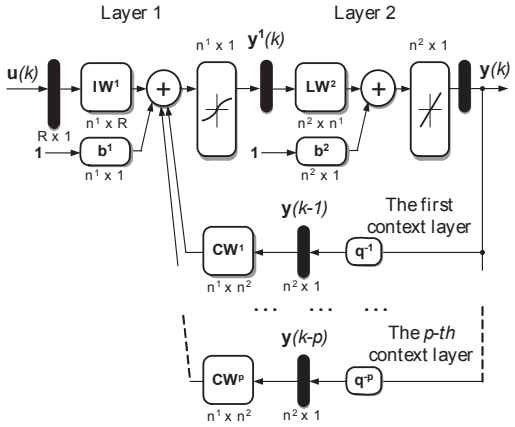
$$\mathbf{p}(k) = \mathbf{CW}^1\mathbf{y}(k-1) + \ldots + \mathbf{CW}^p\mathbf{y}(k-p) \quad (2)$$

b) for multi-context Elman networks:

$$\mathbf{p}(k) = \mathbf{CW}^1\mathbf{y}^1(k-1) + \ldots + \mathbf{CW}^p\mathbf{y}^1(k-p) \quad (3)$$

where **IW**, **LW** are input and layer weight matrixes, $\mathbf{y}(k\text{-}i)$ and $\mathbf{y}^1(k\text{-}i)$ represent output and hidden layer states, $\mathbf{CW}^i$ denoting the weight matrix in the *i-th* context layer, **b** is a bias vector, **σ** is a non-linear function.

(a) multi-context Jordan net



(b) multi-context Elman net

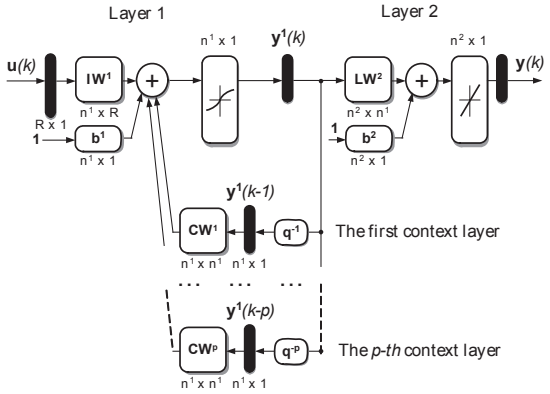


Fig. 1. Partially recurrent networks [19, 20]

2.2. Locally recurrent neural network

Typical locally recurrent architecture is obtained by introducing dynamic elementary processors into the structure of a feedforward network [9]. Such models are frequently called locally recurrent globally feedforward neural networks. Some of the most important strategies for developing dynamic units and locally recurrent topologies, and their applications in areas of technical diagnostics can be found in [1, 8, 9, 10, 11].

The topology of the network used herein is illustrated in Fig. 2. This structure has been introduced and discussed by the author in his previous papers [13, 14, 15]. In the first layer there are simple static neurons with a non-linear output function. The second hidden layer includes dynamic neurons with a non-linear output function. Such units are achieved by introducing linear dynamic

systems into its structure. The last layer consists of simple static units but the output function is linear.

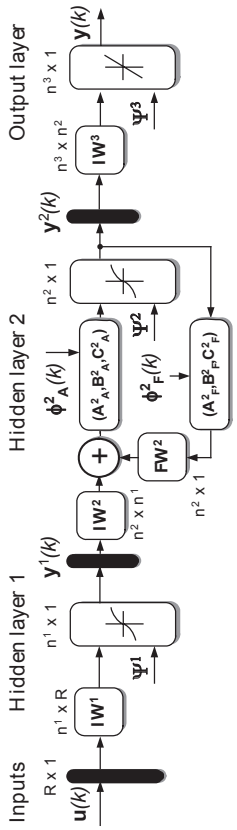


Fig. 2. Locally recurrent network

The output of the *i-th* layer of a locally recurrent network is computed using a non-linear or linear transform operator:

$$\mathbf{y}^{i}(k) = \boldsymbol{\sigma}^{i}\left(\xi_{2}^{i}(k), \boldsymbol{\Psi}^{i}\right) \tag{4}$$

Internal states of neurons in the activation block of the *i-th* layer can be written using the polynomial notation in a vector form:

$$\mathbf{A}_{A}^{i}\left(q^{-1}\right)\circ\xi_{2}^{i}(k)=\mathbf{B}_{A}^{i}\left(q^{-1}\right)\circ\xi_{1}^{i}(k)+\mathbf{C}_{A}^{i}\left(q^{-1}\right)\circ\phi_{A}^{i}(k) \tag{5}$$

and also for the feedback block:

$$\mathbf{A}_{F}^{i}\left(q^{-1}\right)\circ\xi_{3}^{i}(k)=\mathbf{B}_{F}^{i}\left(q^{-1}\right)\circ\mathbf{y}^{i}(k)+\mathbf{C}_{F}^{i}\left(q^{-1}\right)\circ\phi_{F}^{i}(k) \tag{6}$$

An associative input activation of neurons of the *i-th* layer is given by the following expression:

$$\xi_{1}^{i}(k)=\mathbf{IW}^{i}\mathbf{p}^{i}(k)+\mathbf{FW}^{i}\circ\xi_{3}^{i}(k) \tag{7}$$

where $\mathbf{IW}^{i}$ is the matrix of external input weights, $\mathbf{FW}^{i}$ is the vector of feedback weights, $\xi_{n}^{i}(k)$ denotes the vector with activation or feedback states of neurons in the *i-th* layer, $\phi_{A/F}^{i}(k)$ represents the vector of the random process.

3. NEURAL MODELLING

The main objective of training process is to minimize some loss function. Due to this purpose and also to prevent other troubles such as the stability of a model and the local minima problem, a hybrid scheme is used for both types of neural structures. A global algorithm is able to reach the region near an optimum point relatively quickly by minimizing the combined function:

$$E(\boldsymbol{\Theta})=\sum_{m}\left(\mathbf{y}\left(\mathbf{u}^{m},\boldsymbol{\Theta}\right)-\mathbf{t}^{m}\right)^{2}+\beta\sum_{i}\Theta_{i}^{2} \tag{8}$$

where β is a black box parameter, $\mathbf{y}(\cdot)$ is a mapping from the input activities $\mathbf{u}$ to the output activities $\mathbf{y}$, $\mathbf{t}$ is the target output, Θ_i is a network parameter. The second term in Eq. 8 is an additive weight-depend energy.

As one can read in the previous paper of the author [14, 15] for the global optimization several algorithms may be used, for instance: Monte Carlo method, evolutionary algorithm, simulated annealing. Then the solution from the global optimization algorithm is used as an initial point for a local method that is faster and more efficient for local search. It may be realized by minimizing the function (8) for β=0 by means of one of the following gradient-based algorithms: the Levenberg-Marquardt method for relative small networks or the BFGS method for larger networks [16].

The framework proposed by [6] is implemented in order to calculate the gradients and Jacobians for Elman/Jordan tower networks. There are two approaches considered: backpropagation-through-time is used in the BFGS formula, whereas real-time recurrent learning is adapted for Jacobian calculation in the Levenberg-Marquardt method. For the locally recurrent network proposed by the author the gradient vector and Jacobian matrix are derived using a numerical differentiation method (e.g. using gradient-based stochastic optimization algorithm) [14]. Therefore there is no need to have the gradient or Jacobian information calculate analytically, but on the other hand, the algorithm works slow [15, 16].

To obtain good training and generalization abilities the optimal structure of a network should be found. Heuristic rules are usually used in the first step to determine an initial structure of the network [13]. Next, it is possible to apply direct search methods to change complexity of the network during the process of optimization of neural models by increasing or reducing the number of hidden layers (context layers), the number of units in layers, changing structures of dynamic systems embedded in the units, etc. It is strongly connected with a process of model evaluation. In this paper, several measures are employed such as:

a) Mean Absolute Percentage Error:

$$MAPE=\frac{1}{jm}\sum_{j}\sum_{m}\left|\frac{y_{j}\left(\mathbf{u}^{m}\right)-t_{j}^{m}}{t_{j}^{m}}\right| \tag{9}$$

b) Akaike Information Criterion:

$$AIC = \log\left(E_T(\mathbf{\Theta})\right) + 2\frac{d}{m} \tag{10}$$

c) Minimal Description length Criterion:

$$MDL = m\log\left(E_T(\mathbf{\Theta})\right) + d\log(m) \tag{11}$$

where j is the number of outputs, m is the length of the data set, d is the number of estimated parameters, $E_T(\cdot)$ is the loss function (Eq. 8) that is computed for a testing data set (β=0).

3.1. Chaotic rules in global optimization algorithms

In order to improve the computational efficiency of the training process two modifications in global optimization algorithms are proposed by the author. A chaotic mutation operator similar to this used in [21] is introduced and a chaotic modification of the basic simulated annealing is adapted [22].

Chaotic mutation operator

In this paper, the chaotic mutation function adds chaotic numbers taken from a Hénon distribution to each entry of the parent vector. The Hénon map is given by the following equations:

$$\begin{cases} x_{j+1} = 1 - ax_j^2 + y_j \\ y_{j+1} = bx_j \end{cases} \tag{12}$$

where a=1.4, b=0.3. The variance of this distribution is determined by the parameters Scale (λ_0) and Shrink (μ) at each iteration. The Scale parameter determines the variance at the first generation. The Shrink parameter controls how the variance shrinks as generations go by.

$$\lambda_k = \lambda_{k-1}\left(1 - \mu\frac{k}{J}\right) \tag{13}$$

where J is the maximum number of iterations. The corresponding networks parameters are obtained by:

$$\mathbf{\Theta}_{k+1} = \mathbf{\Theta}_k + \lambda_k \mathbf{Y}_k \tag{14}$$

where $\mathbf{Y}_k = \{y_j\ (j=1,2,\ldots,d)\}$.

Chaotic simulated annealing

The key idea of CSA [22] is to replace the Gaussian distribution by a chaotic sequence in the conventional simulated annealing. It is realized by applying the following well-known logistic map:

$$C_{k+1} = 4C_k(1 - C_k) \tag{15}$$

There are three steps needed to adapt this algorithm for training neural models:

a) chaotic initialization: C_0^i (i=1,2,…,d)

b) initialize the temperature T_0

c) conventional annealing iterations: For each of the initial value C_0^i, generate C_j^i (j=1,2,…,J) by Eq. 15, where J is the maximum number of chaotic annealing iterations. The corresponding networks parameters are obtained by: $\Theta_j^i = (1\text{-}\alpha)C^{i*} + \alpha C_j^i$, where α is a constant.

3.2. Pseudo phase-space reconstruction

Phase space reconstruction is usually the first step in the analysis of complex dynamic systems. In a large number of cases during observing real multi-dimensional systems, there is no possibility to measure all of the variables simultaneously. Suppose we can observe only a one-dimensional time series $g(k)$. The most common approach to attractor reconstruction is the method of delays. An method converts the time series $g(k)$ into vectors $\mathbf{Y}(k)$ using time delay Δ:

$$\mathbf{Y}(k) = \left[g(k) \quad g(k+\Delta) \quad \ldots \quad g(k+\eta\Delta)\right]^T \tag{12}$$

where η=D-1, D is the embedding dimension, Δ is the reconstruction delay. The quality of the reconstruction strongly depends upon these two parameters. The reconstruction delay is usually determined using mutual information, whereas for the second parameter one may make use of the false nearest neighbor analysis [15, 24].

4. SIMULATION STUDIES

4.1. Time-series prediction

In the following, the Duffing oscillation model is given in the form of a Takagi-Sugeno fuzzy model [24]:

$$R_D = \begin{cases} If\ x_1(t)\ is\ M_1,\ then\ \dot{\mathbf{x}}(t) = \mathbf{A}_1\mathbf{x}(t) + \mathbf{B}_1 u(t), \\ If\ x_1(t)\ is\ M_2,\ then\ \dot{\mathbf{x}}(t) = \mathbf{A}_2\mathbf{x}(t) + \mathbf{B}_2 u(t). \end{cases}$$

where $\mathbf{x}(t) = [x_1(t)\ x_2(t)]^T$ and $u(t) = 12\cos(t)$, $\mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix}$, $\mathbf{A}_2 = \begin{bmatrix} 0 & 1 \\ -d^2 & -0.1 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and fuzzy membership functions: $M_1(x_1(t))=1\text{-}x_1(t)/d^2$, $M_2(x_1(t))=x_1(t)/d^2$, d=50.

To obtain the time series values at integer points, the fourth-order Runge-Kutta method to find the numerical solution to the above Duffing system was applied (fixed-step size equals 0.01). Moreover, an additive noise was added (SNR=27dB) to this time series and a dawn-sample procedure was executed keeping every fifth sample starting with the first. The time-series prediction problem is formulated as follow: for known values of time series up to the point in discrete time, shall we say, k to predict the value at some point in the future, shall we say, k+H:

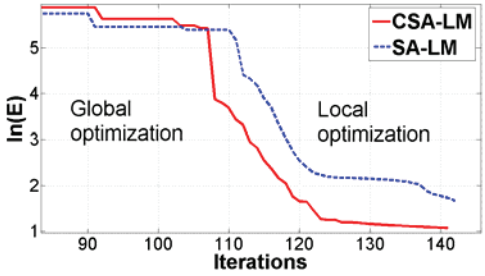
$$\left\langle g(k)\, g(k-\Delta)\ \ldots g(k-\eta\Delta)\right\rangle \rightarrow g(k+H)$$

where: the delay parameter is selected using the mutual information H=Δ=9, whereas the embedding dimension is chosen using the false nearest neighbor analysis D=5 (see Sec. 3.2 for more details).

The first 1000 data values were used for training dynamic neural models while other 1000 data samples were applied in the testing stage (for Duffing orbit x_1 only). For this task several

structures of globally and locally recurrent neural networks were examined in order to model chaotic behavior of the system. In the Fig. 3 there are presented results obtained for different hybrid training schemas of the exemplary Jordan neural network (four inputs, four hidden neurons, two context layers and one output). There were two hybrid schemas used: the SA-LM schema (110 iter. of the simulated annealing with T_0=200 and 35 iter. of the Levenberg-Marquardt method) and the EA-LM schema (30 iter. of the evolutionary algorithm with λ_0=0.8, μ=0.75 and 35 iter. of the LM method, a description of other parameters is omitted here but can be found in [14]). The averaging results show that chaotic modifications of conventional global algorithms lead to improve the performance of the training process. Similar effects were achieved for other recurrent structures.

(a)



(b)

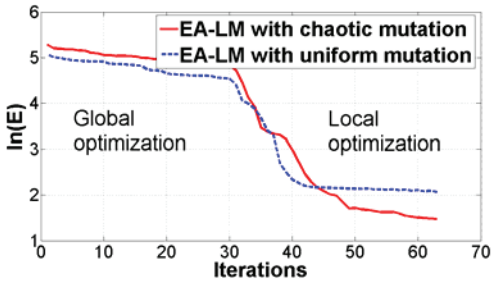


Fig. 3. Hybrid trainings of the Jordan neural network via different schemas

In Tab. 1 selected results are included. These attempts were done for the same EA-LM schema as above. As one can observe the best results are obtained for Jordan and LRNN networks and worse for Elman structures.

Tab. 1. Selected results obtained for different recurrent neural structures (testing stage)

Net	Structure	Param.	MAPE	AIC	MDL
Jordan	4-4$^{(1)}$-1	29	2.187	1.204	1325
Jordan	4-4$^{(1,2)}$-1	33	2.257	1.162	1302
Jordan	4-5$^{(1,2)}$-1	41	1.928	0.922	1106
Jordan	4-5$^{(1,2,3)}$-1	46	2.710	1.661	1856
Elman	4-4$^{(1)}$-1	41	3.864	2.315	2473
Elman	4-4$^{(1,2)}$-1	57	10.91	4.297	4498
LRNN	4-4$^{(1,2,-)}$-1	37	6.129	4.473	4573
LRNN	4-5$^{(1,2,-)}$-1	46	2.551	1.442	1641
LRNN	4-3-2$^{(2,1,-)}$-1	32	2.675	1.839	1962
LRNN	4-4-4$^{(2,2,-)}$-1	61	2.403	1.816	2089

4.2. The three-tank benchmark problem

Let us consider the three coupled tanks depicted in Fig. 4 [13]. The main aim of the control system is to keep the water level in the tank T_3 constant, while a water requirement q_{30} is changed randomly with an uniform distribution. The measured signals are: streams of the medium q_1, q_2 that flow into the first and second tank, control signals U_1 and U_2, levels in tanks L_1, L_2, L_3 and, additionally, discrete signals h_{3L} and h_{3H} from two capacitive proximity switches signaling whether the medium level in the tank T_3 is above or below the position of the sensor.

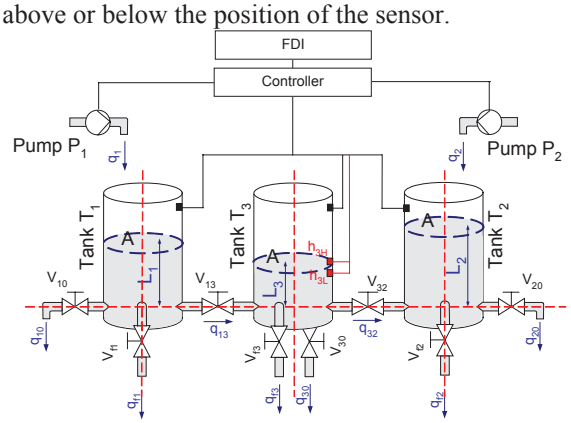


Fig. 4. The three-tank system

For this example many different types of faults like clogs and leakages may be acquired (see Tab. 2). In this paper only one fault is analyzed. Process faults f_1, f_2 and f_3 are investigated: undesirable leakages from tanks appear after 33 min. for the next 33 min. Faults f_7 and f_6 are realized by closing valves V_{13} and V_{32} in the middle of the simulation. Sensor faults are created by subtraction a 30% signal level from their output on the time window as in previous cases.

In general, it is very unlikely to have a chance to acquire the faulty data from industrial installations. However, in some cases, it is possible to create a precise mathematical model (i.e. using analytical methods) of a process taking into account fault-tolerant philosophy [9]. Such models are used in order to generate data for training neural models (off-line). The main aim of this paper is to compare different types of neural networks structures in case of the fault-tolerant control benchmark test and the problem of availability of data is not discussed in detail.

Tab. 2. Set of faults for a three-tank system

Fault	Fault description
f_0	nominal conditions
f_1	undesirable leakage from the tank T_1
f_2	undesirable leakage from the tank T_2
f_3	undesirable leakage from the tank T_3
f_4	Fault of the measuring channel 1
f_5	Fault of the measuring channel 2
f_6	Fault of the measuring channel 3
f_7	Clogging of the valve V_{13}
f_8	Clogging of the valve V_{32}

Fault detection and isolation system

The main idea of the FDI system (Fig. 5) is the same as in the previous paper of the author [13]. The group of three neural models ($NN1_1$, $NN1_2$, $NN1_3$) is used for residual generation, whereas the neural classifier NN2 is used for mapping the space of statistic features of residuals into the space of faults. In the features estimation block five statistic measures for each residual are applied: mean error (me), mean absolute error (mae), mean squared error (mse), standard deviation of errors (sde), variance of errors (ve). Statistic measures of residual signals are computed using a moving time-window of size Δk=200.

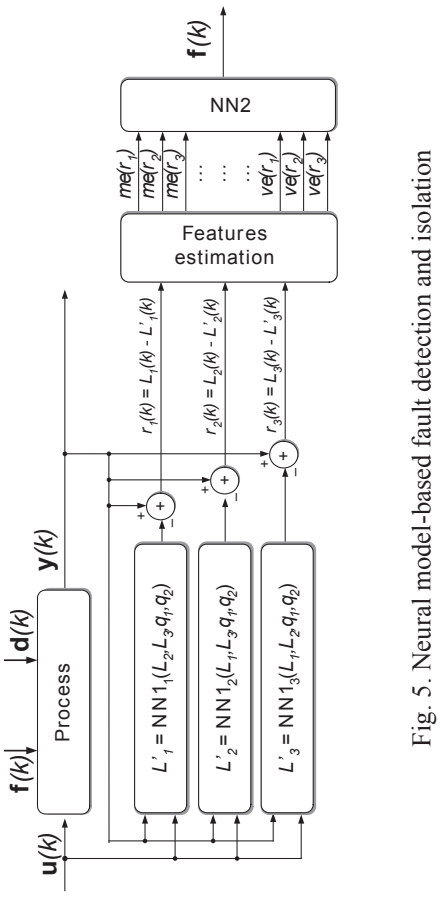


Fig. 5. Neural model-based fault detection and isolation

Three classes of neural networks were considered for residual generation block: L – locally recurrent net, J – multi-context Jordan net, E – multi-context Elman net. Each of them was trained using suitable faultless data (4000 samples). Structures of respective networks were determined basing on heuristic rules discussed in the literature [13]. The processing error for the training set was defined as the sum of squares errors. On the other hand, the error for the testing set was chosen as the mean absolute percentage error. In the training stage, networks parameters were adjusted for a maximum number of epochs or until the goal error was less than 10^{-2}. It leaded to decrease *mape* below 2% for each neural model.

Locally recurrent neural networks $\left(4-5_{(0)}^{(-,2,-)}-1\right)$

The notation in brackets means that the network consists of three processing layers with four inputs, five non-linear dynamic neurons and one linear output neuron. Only one hidden layer with hyperbolic tangent output function was sufficient to identify $NN1_i$ models accurately (*mape* <1%). Neural models were trained by means of the EA-LM hybrid scheme with chaotic mutation operator (30 generations of the EA and 35 iterations of the LM).

Multi-context Jordan neural networks $\left(4-6^{(1,2)}-1\right)$

The network consists of four processing layers with four inputs, six non-linear neurons, one linear output neuron and two context layers (q^{-1}, q^{-2}). The training process was carried out using the same learning-pattern set and also the same EA-LM hybrid schema as in the previous case.

Multi-context Elman neural networks $\left(4-6^{(1,2)}-1\right)$

These structures are similar to the ones discussed previously, however, the internal states of the network are computed using signals from the hidden layer instead of these from the output.

Fault detection and isolation were realized by means of the multilayer perceptron network (M). The structure 15-10-10-8 with hyperbolic tangent output functions in hidden layers and linear function in the output one was enough for mapping the space of statistical features of residuals into the space of faults. Training patterns were obtained for no-fault and fault states (256 examples). Scaled conjugate gradient back propagation method was used for updating network parameters (the maximum number of epochs was equal to 200).

Table 3 presents results (%) obtained for testing data sets. Four FDI schemas are included: L-M – locally recurrent nets for residuals generation and the multilayer net for mapping statistical features of residuals into the space of faults, J-M multi-context Jordan nets and the multilayer net, E-M - multi-context Elman nets and the multilayer net.

The best results the author has received for the L-M schema. As one can observe, the performance of the described FDI system also increased when compared to the results obtained in the paper [13].

Tab. 3. Results of fault detection and isolation

FDI schema	FAULT								
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
L-M	100	100	100	100	100	100	100	100	100
J-M	100	100	93	100	93	100	100	93	93
E-M	100	100	43	100	100	100	100	93	100
L-M	*98*	*75*	*72*	*62*	*98*	*96*	*98*	*100*	*100*

***** *italic designation* denotes results for neural models described in [13]

5. CONCLUSION AND OUTLOOK

Recurrent neural networks are well-known to be popular black-box type models for system identification and time series prediction. Partially recurrent and locally recurrent networks are frequently used in the area of the technical diagnostics within FDI systems. In this paper, it was shown that the applications of such architectures aided by training algorithms with chaotic modifications lead to the improvement of the performance of FDI systems. It can be stated that all FDI schemes provide the fault detection efficiency greater than 95%. However, fault isolation by means of the L-M scheme is more effective.

Finally, it can be assumed that much more complicated neural structures and training algorithms should be continuously developed to determine all potential advantages of this technique.

6. FUTURE WORK

The author plans to adapt this approach in the fault-tolerant control system of the inspection robot that is developed in the Department of Fundamentals of Machinery Design, Silesian University of Technology at Gliwice, carried out within the Multi-Year Programme *"Development of innovativeness systems of manufacturing and maintenance 2004-2008"*.

ACKNOWLEDGMENTS

The author would like to express his thank to Prof. W. Moczulski for his valuable comments and suggestions to improve the paper.

The paper has been prepared with financial support of Ministry of Science and Higher Education in the work of the grant No. N N514 3412 33.

REFERENCES

[1] Ayoubi M.: *Nonlinear dynamic systems identification with dynamic neural networks for fault diagnosis in technical processes*. Humans, Information and Technology, pages 2120–2125, October 1994.

[2] Blanke M., Kinnaert M., Lunze J. and Staroswiecki M.: *Diagnosis and fault-tolerant control*. Springer-Verlag Berlin Heidelberg, 2003.

[3] Caccavale F. and Villani L.: *Fault Diagnosis and Fault Tolerance for Mechatronic Systems: Recent Advances*. Springer Tracts in Advanced Robotics. Springer Berlin / Heidelberg, 2003.

[4] Elman J.: *Finding structure in time*. Cognitive Science, 14(2):179–211, 1990.

[5] Jordan M.: *Serial Order: A Parallel Distributed Processing Approach*. Technical report. California Univ., San Diego, La Jolla. Inst. For Cognitive Science, 1986.

[6] De Jesus O. and Hagan M.: *Backpropagation Algorithms for a Broad Class of Dynamic Networks*. IEEE Transactions on Neural Networks, pages 14–27, January 2007.

[7] Kamiya A., Seppo J., Rajkumar R. and Shigenobu K.: *Fusion of soft computing and hard computing for large-scale plants: a general model*. Applied Soft Computing, 5(3):265–279, 2005.

[8] Korbicz J.: *Robust fault detection using analytical and soft computing methods*. Bulletin of the Polish Academy of Sciences : Technical Sciences, 54(1):75–88, 2006.

[9] Korbicz J., Kościelny J., Kowalczuk Z. and Cholewa W. (Eds.): *Fault Diagnosis. Models, Artificial Intelligence, Applications*. Springer Berlin / Heidelberg, 2004.

[10] Patton R., Frank P. and Robert C.: *Issues of Fault Diagnosis for Dynamic Systems*. Springer-Verlag Berlin and Heidelberg, 2000.

[11] Patan K.: *Stability Analysis and the Stabilization of a Class of Discrete-Time Dynamic Neural Networks,* IEEE Transactions on Neural Networks, May 2007.

[12] Patan K.: *Approximation of state-space trajectories by locally recurrent globally feed-forward neural networks*. Neural Networks, pages doi:10.1016/j.neunet.2007.10.004, 2007.

[13] Przystałka P.: *Model-based fault detection and isolation using locally recurrent neural networks*. Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, June 22-26, 2008, (to be printed).

[14] Przystałka P.: *Hybrid learning algorithm for locally recurrent neural networks.* Fault diagnosis and fault tolerant control, Academic Publishing House EXIT, pages 255–262, September 2007.

[15] Przystałka P.: *Heuristic modeling using recurrent neural networks: simulated and real-data experiments*. Computer Assisted Mechanics and Engineering Sciences, 14(4):715–727, November 2007.

[16] Przystałka, P.: *Heuristic modeling of objects and processes using dynamic neural networks.* Diagnostyka, 2/38:15–18, 2006.

[17] Schlang M., Lang B., Poppe T., Runkler T. and Weinzierl K.: *Current and future development in neural computation in steel processing*. Control Engineering Practice, 9(9):975–986, 2001.

[18] Tomanek A., Przystałka P. and Wyczółkowski R.: *Optimization of Jordan and Elman neural networks through distributed computing environment.* Methods of Artificial Intelligence, pages 69–70, November 2007.
[19] William W.: *A Comparison of Architectural Alternatives for Recurrent Networks*. Proceedings of the fourth Australian Conference on Neural Networks, pages 189–192, February 1993.
[20] William W.: *Stability of learning in classes of recurrent and feedforward networks*. Proceedings of the sixth Australian Conference on Neural Networks, pages 142–145, February 1995.
[21] Xiaohua Y., Zhifeng Y., Xinan Y. and Jianqiang L.: *Chaos gray-coded genetic algorithm and its application for pollution source identifications in convection diffusion equation.* Elsevier B.V. pp. 1676-1688, 2008.
[22] Yao D., Zeng M. and Yongjie Li: *Chaotic simulated annealing algorithm applied to ERP dipole localization*. Proceedings of the International Conference on Communications, Circuits and Systems, pp 908-911, May 2005.
[23] Zadeh, Lotfi A.: *Fuzzy Logic, Neural Networks and Soft Computing*. Communications of the ACM, 37(3):77–84, March 1994.
[24] Zhong L., Halang W. and Chen G.: *Integration of Fuzzy Logic and Chaos Theory*. Studies in Fuzziness and Soft Computing. Springer-Verlag Berlin / Heidelberg, 2006.

Piotr PRZYSTAŁKA is currently a PhD student at the Department of Fundamentals of Machinery Design at Silesian University of Technology in Gliwice. He received the M.Sc. degree in design and exploitation of the machines from the same department in 2004. He deals with modeling linear, non-linear and chaotic dynamic systems and the applications of artificial neural networks in fault-tolerant control systems.