DYNAMIC FUNCTIONAL – LINK NEURAL NETWORKS GENETICALLY EVOLVED APPLIED TO FAULT DIAGNOSIS

T. Marcu*, B. Köppen-Seliger, P.M. Frank, S.X. Ding

University of Duisburg-Essen, Institute of Automatic Control and Complex Systems (AKS)
Bismarckstrasse 81 (BB), D-47057 Duisburg, Germany
*Phone: +49-203-3794293; Fax: +49-203-3792928;
e-mail: {t.marcu, bks, p.m.frank, s.x.ding}@uni-duisburg.de

Keywords: fault diagnosis; dynamic neural networks; non-linear system identification; genetic algorithms; multi-objective optimisation.

Abstract

The paper addresses the development of neural observer schemes for process fault diagnosis. The design is based on a generalised functional-link neural network with internal dynamics. An evolutionary search of genetic type and multi-objective optimisation in the Pareto-sense is used to determine the optimal architecture of the dynamic network. Symptoms characterising the current state of the process are obtained based on prediction errors. The latter are further evaluated by a static artificial network. Experimental results regarding the detection and isolation of artificial sensor faults in an evaporation station from a sugar factory illustrate the approach.

1 Introduction

Artificial Neural Networks (ANNs) have been suggested as a possible data-based technique to cope with the robustness problem in Fault Detection and Isolation (FDI) [5,8,9]. The robustness of a diagnosing system implies the maximisation of detectability and isolability of faults, under the constraint of minimisation of the false alarm rate [4]. ANNs are currently used as both predictors of dynamic non-linear models for symptom generation and pattern classifiers for the evaluation of symptoms. However, in the framework of real applications [6], it is necessary to perform an updating of the designed neural diagnosing systems when there is either a variation of the operation point of the plant and (or) a change in the process configuration.

The identification of dynamic systems requires models with adequate memory. Therefore, the ANNs have to be provided with dynamic elements and appropriate learning methods [5]. A first approach refers to neural networks with external dynamics, e.g. static ANNs equipped with tapped delay lines [5,12]. A better approach is achieved by providing the ANN with internal dynamics [5]. This kind of network processes multi-inputs and does not require past values of process measurements as current inputs. Such a neural net is referred to as a dynamic network in the present paper. The experience gained using the dynamic ANN architectures [8,9,11] has evidenced the considerable computational effort that is involved in the design stage when the selection of the optimal structure of an ANN is based on a trial-and-error approach.

In the mentioned context, the paper first suggests a generalised version of several Dynamic Functional-Link Neural Networks (DFLNNs) [8,11]. That generalised network is used to approximate non-linear models of a plant. The design is formulated as a problem of multi-objective optimisation. Genetic algorithms are used to find out the optimal dynamic architecture of the network. The optimality refers to the approximation accuracy and net complexity. The neural design of an FDI system is next presented. A neural variant to the generalised observer scheme [4] is used to generate symptoms reflecting the current state of a process. The symptoms are given by the prediction errors obtained by using the DFLNN genetically evolved. The generated symptoms are further evaluated by means of a static multi-layer perceptron [2,12] used as pattern classifier. The presented methodology is illustrated with a case study based on real data supplied by industry [15]. Finally, certain conclusions regarding the suggested approach are included.

2 Generalised dynamic functional-link network

The Functional-Link Neural Network (FLNN) has a feed-forward architecture with no hidden layer(s). Instead, a number of non-linear enhancement nodes, referred to as functional links, are used to provide supplementary inputs within the network [13,14]. In the following, the functional expansion given by a sub-set of orthogonal trigonometric functions is considered. This provides a more compact representation of the function to be approximated, in the mean-square sense, than other orthogonal basis functions [14]. A so-called flat network results for which only the connection weights and bias term(s) must be learned. The Back-Propagation (BP) learning method [2,12], used for adapting the FLNN's parameters, therefore becomes very simple.

A generalised version of previously developed Dynamic FLNNs (DFLNNs) [8,11] is characterised by a variable set of functional links and integrates conveniently dynamic elements into a static FLNN. This is suggested in the following and illustrated generally by Fig. 1.

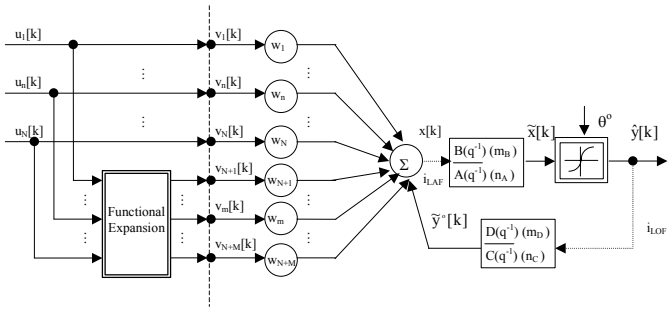

Figure 1: The structure of the generalised dynamic functional-link neural network: N inputs, M functional expansion terms, local activation feedback (i_{LAF}=1), local output feedback (i_{LOF}=1), one non-linear activation unit and one output; q^{-1} stands for the linear operator of time shifting; A,B,C,D are polynomials.

The initial inputs of the net u_n, $n=1,...,N$, are functionally expanded to constitute the actual inputs of the non-linear neuron, v_m, $m=1,...,N+M$, given by the following set:

$$\{u_n, \{i_{u_n}\cdot\cos(j_n\cdot\pi\cdot u_n)\}, \{i_{u_n}\cdot\sin(k_n\cdot\pi\cdot u_n)\}\}$$

$$j_n=1,...,S_n^{\cos};\ k_n=1,...,S_n^{\sin};$$

$$S_n^{\cos}, S_n^{\sin}\in\{1,...,S_{\max}\};\ n=1,...,N$$

$$M=\sum_{n=1}^{N}(S_n^{\cos}+S_n^{\sin})\le 2\cdot S_{\max}\cdot N$$

where, for each initial input u_n, i_{u_n} indicates the presence (value of 1) or absence (value of 0) of the functional expansion, $S_n^{\cos}$ and $S_n^{\sin}$ are the orders of functional expansion corresponding to the cosine and sine terms, respectively, and $S_{\max}$ denotes a pre-specified maximum order of the functional expansion. At least one initial input of the net has to be subject of the non-linear enhancement, i.e. $\exists\, i_{u_n}=1, n=1,...,N$. The present consideration of the functional inputs generalises the previous approaches [8,11,14], where a fixed order of functional expansion has been considered $S_n^{\cos}=S_n^{\sin}=S_{\max}$, n=1,...,N.

In order to provide the FLNN with adequate internal memory, an Auto-Regressive Moving Average (ARMA) filter can be placed: either before the non-linear activation unit of the neuron, the resulting network being a DFLNN with Local Activation Feedback (DFLNN_LAF) [8]; or on the back connection from the network output to the neuron's input, the resulting network being a DFLNN with Local Output Feedback (DFLNN_LOF) [8]; or on both places, the resulted network being a DFLNN with MIXed structure (DFLNN_MIX) [11]. For the Generalised DFLNN (GDFLNN), the implementation is realised in such a manner that one can select one of the three considered variants based on the specified architecture's parameters i_{LAF} and i_{LOF}.

The following equations describe the propagation of the actual inputs of the network, through the GDFLNN (Fig.1), providing the output $\hat{y}$ of the considered hyperbolic tangent neuron:

$$x[k]=\sum_{m=1}^{N+M}w_m\cdot v_m[k]+i_{\mathrm{LOF}}\cdot\tilde{y}^o[k]$$

$$\tilde{y}^o[k]=\sum_{j=1}^{m_{\mathrm{D}}}d_j\cdot\hat{y}[k-j]-\sum_{i=1}^{n_{\mathrm{C}}}c_i\cdot\tilde{y}^o[k-i]$$

$$\tilde{x}[k]=b_0\cdot x[k]+i_{\mathrm{LAF}}\cdot\{\sum_{j=1}^{m_{\mathrm{B}}}b_j\cdot x[k-j]-\sum_{i=1}^{n_{\mathrm{A}}}a_i\cdot\tilde{x}[k-i]\}$$

$$\text{if } i_{\mathrm{LAF}}=0\text{, then } b_0\equiv 1 \text{ and } \tilde{x}[k]\equiv x[k]$$

$$z^o[k]=\tilde{x}[k]+\theta^o$$

$$\hat{y}[k]=(e^{z^o[k]}-e^{-z^o[k]})/(e^{z^o[k]}+e^{-z^o[k]})$$

where [k] represents the sampling time instant k and the following notations are used:

- FLNN: w_m, m=1,...,N+M, represent the connection weights, θ^o denotes the bias term, and z^o is the input of the activation unit of the neuron;
- LAF filter: m_{B} represents the numerator order, n_{A} denotes the denominator order, b_j, j=0,...,m_{B} are the numerator coefficients and a_i, i=1,...,n_{A} are the denominator coefficients, $m_{\mathrm{B}}\in\{0,1,...,m_{\max}\}$, $n_{\mathrm{A}}\in\{1,...,n_{\max}\}$;
- LOF filter: m_{D} denotes the numerator order, n_{C} represents the denominator order, d_j, j=1,...,m_{D} are the numerator coefficients and c_i, i=1,...,n_{C} are the denominator coefficients, $m_{\mathrm{D}}\in\{1,...,m_{\max}\}$, $n_{\mathrm{C}}\in\{1,...,n_{\max}\}$,

where $m_{\max}$ and $n_{\max}$, respectively, are maximal orders for the polynomials of ARMA filter(s).

The parameters characterising the architecture of GDFLNN are given by the following sets:

$$\{i_{u_n}; S_n^{\cos}, S_n^{\sin}\}_{n=1,...,N};\quad \{i_{\mathrm{LAF}}; m_{\mathrm{B}}, n_{\mathrm{A}}\};\quad \{i_{\mathrm{LOF}}; m_{\mathrm{D}}, n_{\mathrm{C}}\}$$

All these parameters have been provided, for the previous approaches [8,11] characterised by certain restrictions, by a trial-and-error process in a pre-defined space.

For a given architecture, the parameters of the GDFLNN are the connection weights, the coefficients of filter(s), and the bias term. These parameters are determined with an extended, Dynamic BP (DBP) algorithm [8,9,11]. The latter minimises the sum of squared errors between the training data and the approximating values provided by the ANN.

The design of the GDFLNN is based on three sets of representative process data as follows [2,12]: a *training data set* used for model identification, different models' architectures and parameters being determined; a *data set for validation* used to select the best identified model(s); a *data set for* model *testing* used to evaluate the quality of the validated model(s). For the DBP learning, the batch learning mode [2] is applied, i.e. the net parameters are adapted after an entire pass of the training data through the network (one epoch). The mechanisms of variable parameter of learning rate and momentum term are considered as well [2].

3 Genetic evolving of GDFLNN

The specification of the optimal GDFLNN architecture and related parameters is a rather difficult task when a trial-and-error approach is considered [8,11]. Therefore, a Genetic Algorithm (GA) is used to solve this problem. Since the remaining parameters of the GDFLNN (connection weights, filters' coefficients, and bias term) have meaning only for a pre-defined dynamic architecture, they are determined by means of the extended DBP algorithm. The latter is integrated into the genetic search, when the potential solutions are evaluated for their fitness to the application specific design.

Evolutionary algorithms of genetic type are principally based on computational models of fundamental processes, such as selection, recombination and mutation [1,3,7]. An algorithm of this type begins with a set (population) of parameters' estimates (genes), called individuals (chromosomes) appropriately encoded. Each one is evaluated for its fitness in solving a given optimisation task. At each iteration (algorithm time-step), the most fit individuals are allowed to mate and bear offspring.

3.1 Neural architecture encoding

The GDFLNN is encoded using a structured formulation of the chromosome in a hierarchical fashion [7,9]. The resulted hierarchical GA has the following main features: the chromosome has a multi-level genetic structure, i.e. a directed graph, and consists of multiple level control genes and parametric genes; genes at any level can be either active (the value of '1' is assigned) or passive (the value of '0' is assigned); the inactive genes remain within the chromosome structure and can be carried forward for further generations; "high level" (control) genes activate or deactivate the "low level" (parametric) genes according to their current values; the standard genetic operations of recombination and mutation are applied independently to each level of genes.

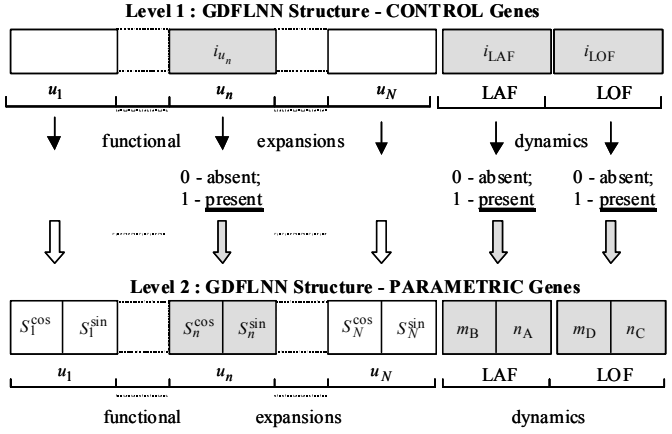

Figure 2: Hierarchical formulation of chromosome for the design of the generalised dynamic functional-link neural network.

Fig. 2 presents the hierarchical structure of the chromosome that is used for the design of GDFLNN, as described in the previous section of the paper. The highest level 1 controls the presence or absence of the functional expansion for each network's initial input, and of the dynamic elements. The involved parameters are binary coded. The lowest (second) level contains the parameters of the functional expansion, i.e. the orders of activated cosine and sine terms, and of the dynamic internal structures, i.e. the orders of activated ARMA filter(s). These parameters are coded as integer values.

The search space is defined by assigning certain maximal values for the orders of functional expansion, $S_{\max}$, and orders of ARMA filter(s), $m_{\max}$ and $n_{\max}$, respectively (section 2 of the paper). These values determine the maximal length of the chromosomes as well.

3.2 Objective functions

A multi-objective approach is adopted for the design of an optimal GDFLNN. Two categories of objectives are considered to be minimised. They characterise the approximation accuracy of the network and its complexity. Since the first distinction is of paramount importance, the associated objectives have different priorities. The following objective functions are considered in the present approach, as they refer to items of different meaning:

- O_1 = the Sum of Squared Errors (SSE) that characterise a certain architecture of GDFLNN and its training data set;
- O_2 = the sum of squared errors that characterise a certain architecture of GDFLNN (as for the objective function O_1) and its validation data set;
- $O_{2+n} = S_n^{\cos} + S_n^{\sin}; n = 1,..., N$ = the number of active functional expansions corresponding to each net's input;
- $O_{N+3} = m_{\mathrm{B}} + n_{\mathrm{A}}$ = the number of coefficients of active LAF filter;
- $O_{N+4} = m_{\mathrm{D}} + n_{\mathrm{C}}$ = the number of coefficients of active LOF filter.

The following priority assignment is considered: objectives O_1 and O_2 have the same priority of high level; objectives $O_{2+n}; n = 1,..., N$, O_{N+3} and O_{N+4} have the same priority of low level.

Multi-objective Optimisation based on GA (MOGA) seeks to optimise the components of a vector-valued cost function. The solution is not a single point, but a family of points known as the Pareto-optimal set [3,7]. Each point in this surface is optimal in the sense that no improvement can be achieved in one component of the objectives' vector without degradation in at least one of the remaining components. The process of optimisation is seen as the result of the interaction between an artificial selector, referred to as a Decision Maker (DM), and an evolutionary search process of genetic type (GA). Thus, the search process generates a new set of candidate solutions according to the utility assigned by the DM to the current set of candidates. This approach permits accommodation of goal and priority information available from the problem formulation.

3.3 MOGA Procedure

For the genetic evolving of the GDFLNN, the following procedure is adopted and sketched out in the sequel, based on the work previously reported [3, 9]:

- *Initialisation*:

Create an initial hierarchical population; Check consistency of the initial population (with remedy actions if necessary); Apply extended DBP to GDFLNN architectures corresponding to initial population; Calculate objective functions for population; Calculate goals for population; Calculate Pareto-ranking values for population; Select initial best individual(s).

- *Evolutionary loop*:

loop over a number of *MAX_GEN* generations:

- *GA Stage*: Fitness assignment to whole population; Select individuals from population (stochastic universal sampling); Recombination and mutation; Check consistency of offspring (with remedy actions if necessary); Apply extended DBP to GDFLNN architectures corresponding to offspring; Calculate objective functions for offspring; Calculate Pareto-ranking values for offspring; Insert best offspring into population replacing worst parents (Elitist Strategy);
- *DM Stage*: Adapt goals; Calculate Pareto-ranking values for new population; Select best individual(s).

- *Exit*:

Determine best individual(s) over all performed generations, including post-training analysis (linear regression and correlation) applied to the response of identified model(s) [2].

The mentioned remedy actions are applied to maintain feasible dynamic architectures of GDFLNN into a population. The corrective actions are made by means of repeated application of the mutation operator in the corresponding fields of the control and (or) parametric genes. The DBP is applied for a reasonable low number of epochs to investigate efficiently the search space. Finally, the DBP is applied for a greater number of epochs to the best individual(s) found at exit. The adaptation of goal values (used in the Pareto ranking) is achieved such that the search is uniformly directed towards the middle region of the trade-off surface of objectives.

4 Neural design of an FDI system

For the *generation of symptoms*, the neural networks replace the analytical model that describes the process. An ANN model for each system output is identified by using both inputs and outputs of the process as inputs to the network for a known class of process behavior, usually the normal operation. This way, the resulted models are used for the separate estimation of process output signals in observer-like arrangement [8,9]. Process symptoms, i.e. residuals, are then obtained by subtracting the approximations of the neural observer from the corresponding process measurements.

With respect to the application described in the next section of the paper, the GDFLNN is used to design an extended version of the Generalized Observer Scheme (GOS) [4,9]. This is applied to detect and isolate sensor and (or) actuator faults. All networks are trained using data corresponding to the normal behavior of the process. In the following, a Multi-Input Single Output (MISO) process is considered, having I measured inputs $u_{\mathrm{P},i}[k], i = 1,...,I$ and one output $y_{\mathrm{P}}[k]$, all known at sampling time k.

The considered process is first identified by one GDFLNN. This is driven by all inputs and the output of the system. The network estimates the output of the process:

$$\hat{y}_0[k] = f_0(\mathbf{u}_{\mathrm{P}}[k], y_{\mathrm{P}}[k-1]); \quad \mathbf{u}_{\mathrm{P}}[k] := [u_{\mathrm{P},i}[k]]_{i=1,...,I} \qquad (1)$$

where f_0 denotes the mapping performed by the GDFLNN. Secondly, the Neural GOS (NGOS) is developed. It consists of as many GDFLNNs as process inputs are available. Each such network of NGOS is driven by the process output and all inputs but one input. The output of the MISO process is approximated by the mappings f_j of the GDFLNNs:

$$\begin{aligned} &\hat{y}_j[k] = f_j(\mathbf{u}_{\mathrm{P},j}[k], y_{\mathrm{P}}[k-1]); \\ &\mathbf{u}_{\mathrm{P},j}[k] := [u_{\mathrm{P},i}[k]]_{i=1,...,I;i\neq j}\ ;\ j = 1,...,I \end{aligned} \qquad (2)$$

The residuals are generated by means of the extended NGOS given by the relationships (1) and (2). One step-ahead prediction errors are obtained as:

$$r_l[k] = y_{\mathrm{P}}[k] - \hat{y}_l[k], l = 0,1,...,I \qquad (3)$$

The residual r_0 is affected by all process variables. Instead, each residual $r_l, l = 1,...,I$ is sensitive to the process output and all inputs but the l-th input. In case of an input variable affected by a fault, the decoupled residual remains small, while the others are influenced. All these patterns of change are further used to locate the faults.

The next stage of *symptom evaluation* can be seen as a classification problem. This means to match each pattern of the symptom vector (3) with one of the pre-assigned classes of faulty behaviour, if available, and the fault-free class, respectively. An ANN classifier used for FDI has as inputs the residual signals and must produce pre-assigned outputs characteristic to each known class of process behaviour. A static Multi-Layer Perceptron (MLP) with sigmoid neurons and BP rule of learning is considered in the present approach [2], to assess the quality with respect to FDI (i.e., sensitivity) of residuals generated by different dynamic ANNs.

5 Application

The presented methodology was under assessment by using real process data from the evaporation station of Lublin sugar factory in Poland [15]. The investigated process is a five-stage industrial system characterised by high complexity. No process models from the sugar factory are available. Instead, the knowledge of the technological flux and measurement values in a huge database do exist. All these justify the black-box approach of applying neural networks to modelling and monitoring of the process. To implement the presented neural

approach, the following were used and extended: the MATLAB programming environment [10], the Neural Network Toolbox [2], and the standard GA Toolbox (GAT) [1].

5.1 Process Description

The evaporation process reduces the water content of sucrose juice. The liquor goes through a series of five sections. In each passage, the sucrose concentration is increased. The juice steam recovered from one stage is used as a heating source for the next section. Data from the heater and first section of the evaporation section (Fig. 3) have been considered for a case study [15].

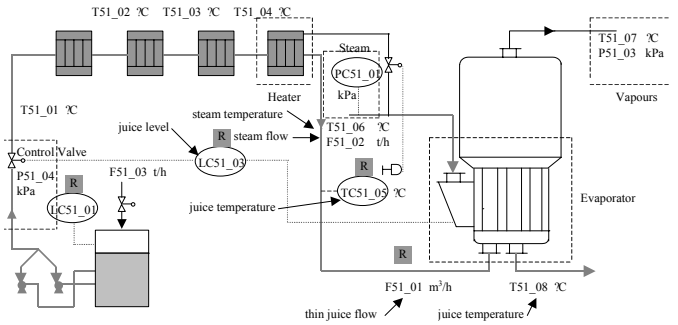

Figure 3: Heater and first section of evaporation station from a sugar factory.

The factory is equipped with a Supervisory Control And Data Acquisition system (SCADA) that allows the registration and storage of all set-point, control and process variables. The faults are rather exceptional, due to the careful inspection of installation before starting the three-months long production period. The fault-free data stored by SCADA during one month, every T_S=10 s, were used to extract the training and validation data sets for developing the residual generators.

For testing and benchmarking purpose, 3000 rows of measurements (corresponding to approximately 8 hours) from another month of plant exploitation were used. Another testing set was created by introducing artificial single faults in the first testing data set. Faults were generated by adding or subtracting 5, 10 and 15 percent of the whole possible range of variation to each measurement, respectively, in different periods of time. According to the benchmark [15], those simulated faults had to be detected and isolated.

5.2 Experimental Results

The results corresponding to FDI of the "Evaporator" Sub-system (ES, Fig.3) are only reported here due to space restrictions. Similar results were obtained for the other sub-systems of "control valve", "heater", "steam", and "vapours" (Fig.3) considered in that case study [15]. The GDFLNNs were used to design an NGOS for that ES. The inputs of the models (1) and (2) are: $u_{P,1}$ – the steam flow to the input of ES, $u_{P,2}$ – the steam temperature at the input of ES, and $u_{P,3}$ – the juice temperature after heater. The modelled output is y_P – the juice temperature after section 1 of ES. Artificial faults affect all four variables. To develop a model, isolated missing and uncertain measurements were replaced by means of linear interpolation. A spectral analysis was then performed and a low-pass filtering (by means of appropriate discrete-time Butterworth filters) was applied to reduce the noise and the amount of data used in the ANN learning. The training and validation data, each containing 3000 rows of measurements, were decimated (using each 10^{th} sampled value).

The search space for the genetic procedure was defined by setting the following parameters (section 2): S_{max}=5, and m_{max}=n_{max}=3. The parameters for different GAT functions [1] were considered as (section 3): *MAX_GEN*=30 generations, 30 individuals in a population, selection rate of 0.8 for recombination, discrete recombination rate of 0.7 (uniform crossover), mutation rate of 0.01, offspring reinsertion rate of 0.8. During the genetic search, the GDFLNNs were trained for 500 epochs. The best architecture of GDFLNN found at exit was trained again for 3000 epochs. The resulted GDFLNNs are characterised by a simple structure (e.g. not all network inputs are functionally expanded), and a performance given by SSE of order of 10^{-2} over the reduced testing data set of 300 points. However, the most sensitive residuals and best FDI performance are obtained with GDFLNNs having all inputs functionally expanded (i.e. it was necessary to restrict the search space), the resulted networks having the same accuracy performance over the testing data set without faults.

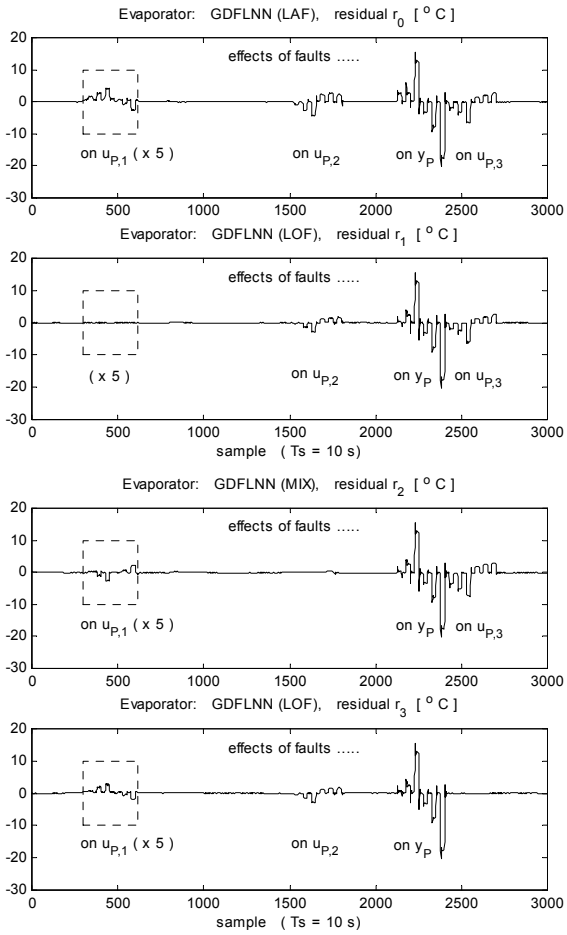

Figure 4: Evaporator, residuals obtained with the GDFLNNs.

The generated residuals (3) corresponding to the finally resulted GDFLNNs are presented in Fig.4, the case of testing data with faults. All residuals are influenced by the faults on the output signal (samples 2100...2400). The second residual, r_1, is almost not influenced by the faults on the first process input (samples 300...600). The third residual, r_2, is almost not influenced by the faults on the second process input (samples 1500...1800). Finally, the fourth residual, r_3, is almost not influenced by the faults affecting the third process input (samples 2400...2700).

For each faulty signal, two classes were considered. Altogether nine classes of sub-process behaviour are diagnosed: normal behaviour, and positive and negative deviations of each of four process variables, respectively. To train an MLP/BP classifier with two layers of sigmoid neurons [2], the residuals obtained from testing data with faults (Fig.4) were used. A supplementary criterion of decision was considered to eliminate isolated false and wrong alarms [9]: a fault is validated if the classifier recognises a certain class at least three diagnosis cycles of T_S=10 s each. The best results of initial classification and final decision ("diagnosis") are included in Table 1, fourth row.

Table 1: Evaporator, performance of diagnosing systems

NGOS (generated residuals)	Recognition rate	
	classification	diagnosis
DFLNN_LAF (r_0, r_1, r_2, r_3)	96.00%	96.06%
DFLNN_LOF (r_0, r_1, r_2, r_3)	91.12%	91.89%
DFLNN_MIX (r_0, r_1, r_2, r_3)	95.93%	96.33%
GDFLNN (LAF: r_0; LOF: r_1, r_3; MIX: r_2)	*96.90%*	*98.03%*
GDMLP (r_0, r_1, r_2, r_3)	98.57%	98.87%

Table 1 presents comparatively the results of fault classification and diagnosis of the ES based on the NGOS designed with the previously developed DFLNN_LAF, DFLNN_LOF [8], and DFLNN_MIX [11], the suggested GDLFNN genetically evolved, and a Generalised Dynamic MLP (GDMLP) genetically evolved [9]. The results obtained with the GDFLNN genetically evolved are quite close to the best ones obtained with the GDMLP. However, the GDFLNN is characterised by a much simpler architecture, and a reduced training and evaluation time.

6 Conclusions

The paper suggests a multi-objective genetic approach to the design of a generalised functional-link neural network with internal dynamics. This is used to develop a generalised observer scheme. The latter provides structured sets of residuals, in order to perform a robust diagnosis of process faults. The obtained symptoms are classified by a static neural net. Diagnosing systems were designed to detect and isolate incipient sensor and actuator faults in an evaporation process.

The developed genetic procedure represents a semi-automatic method of selecting the appropriate network architecture for the task of non-linear system identification. The user must assign certain parameters of the genetic search. This seems to be easier than manually selecting the network architecture. The design time is reduced, for the presented application this meant a reduction from several days to several hours for each designed neural network. Such an approach seems to be appropriate for an efficient re-design of a neural diagnosing system, in the face of changing operating conditions of the monitored process. Current research refers to the application of the presented methodology to other types of processes [6].

Acknowledgements. The authors would like to acknowledge the support for this work by the EU-IST-2000-30009 project "Multi-Agents-Based Diagnostic Data Acquisition and Management in Complex Systems" (MAGIC).

References

[1] Chiepperfield, A.J., P.J. Fleming, H. Pohlheim and C.M. Fonseca, *Genetic Algorithm Toolbox for Use with MATLAB*, University of Sheffield, UK, (1996).

[2] Demuth, H. and M. Beale, *Neural Network Toolbox*, The MathWorks Inc., Natick, MA, (2002).

[3] Fonseca, C.M., *Multiobjective Genetic Algorithms with Application to Control Engineering Problems*, Ph.D. Thesis, University of Sheffield, UK, (1995).

[4] Frank, P.M., "Enhancement of Robustness in Observer-based Fault Detection", *Int. Journal of Control*, **59**, 4, pp. 955-981, (1994).

[5] Isermann, R., S. Ernst and O. Nelles, "Identification with Dynamic Neural Networks", Prep. *IFAC Symposium SYSID*, Fukuoka, Japan, Vol.3, pp. 997-1022, (1997).

[6] Köppen-Seliger, B., S.X. Ding and P.M. Frank, "MAGIC – IFATIS: EC-Research Projects", CD-ROM Proc. *IFAC World Congress*, Barcelona, Spain, (2002).

[7] Man, K.F., K.S. Tang and S. Kwong, *Genetic Algorithms: Concepts and Designs*, Springer-Verlag, London, (1999).

[8] Marcu, T., L. Mirea, P.M. Frank and H.D. Kochs, "System Identification and Fault Diagnosis Using Functional-Link Neural Networks", CD-ROM Proc. *EUCA European Control Conference*, Porto, Portugal, pp. 1618-1623, (2001).

[9] Marcu, T. and P.M. Frank, "Process Fault Detection and Isolation Using Dynamic Multilayer Perceptrons Genetically Evolved", *Int. Journal of Differential Equations and Dynamical Systems*, **10**, 1-2, pp. 169-200, (2002).

[10] MathWorks, *MATLAB – The Language of Technical Computing*, The MathWorks Inc., Natick, MA, (2002).

[11] Mirea, L. and T. Marcu, "System Identification Using Functional-Link Neural Networks with Dynamic Structure", CD-ROM Proc. *IFAC World Congress*, Barcelona, Spain, (2002).

[12] Nørgaard, M., O. Ravn, N.K. Poulsen and L.K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag, London, (2000).

[13] Pao, Y.H., G.H. Park and D.J. Sobajic, "Learning and Generalisation Characteristics of the Random Vector Functional-Link Net", *Neurocomputing*, **6**, pp. 163-180, (1994).

[14] Patra, J.C., R.N. Pal, B.N. Chatterji and G. Panda, "Identification of Non-linear Dynamic Systems Using Functional-Link Artificial Neural Networks", *IEEE Trans. on Systems, Man, and Cybernetics – part B: Cybernetics*, **29**, 2, pp. 254-262, (1999).

[15] Syfert, M. (Organiser), "Research on Quantitative and Qualitative FDI Methods based on Data from Lublin Sugar Factory" (Invited Session), Prep. *IFAC Symp. SAFEPROCESS*, Budapest, Hungary, Vol.1, pp. 331-363, (2000).