15th Triennial World Congress, Barcelona, Spain

FDI USING NEURAL NETWORKS - APPLICATION TO SHIP BENCHMARK ENGINE GAIN

Jan Dimon Bendtsen and Roozbeh Izadi-Zamanabadi *,[1]

* *Aalborg University, Department of Control Engineering, Fredrik Bajers Vej 7, DK-9220 Aalborg Ø, Denmark; fax: (45) 98 15 17 39; e-mail: {dimon,riz}@control.auc.dk*

Abstract: This paper concerns fault detection and isolation based on neural network modeling. A neural network is trained to recognize the input-output behavior of a nonlinear plant, and faults are detected if the output estimated by the network differs from the measured plant output by more than a specified threshold value. In the paper, a method for determining this threshold based on the neural network model is proposed, which can be used for a design strategy to handle residual sensitivity to input variations. The proposed method is used for successful fault detection and isolation of a diesel engine gain fault in a ship propulsion benchmark simulation.

Keywords: Ship control, Fault detection and isolation, neural networks

1. INTRODUCTION

Fault detection and isolation (FDI) has been subject to extensive research in the last three decades. The main part of the research has been performed on model based linear systems (Patton, 1997). FDI for nonlinear systems has come into focus in recent years (Frank *et al.*, 1999) and dedicated methods have been proposed to generate fault indicating residuals for different classes of nonlinear systems (DePersis and Isidori, 2000; Hammouri *et al.*, 1999).

In order for any fault detection scheme to succeed, it is necessary for the behavior of a model of the system in question to match the behaviour of the actual system. In case of nonlinear, noisy systems it may be difficult to establish a satisfying model of the system based on first principles, however, which has caused several researchers to study data driven approaches to the modeling task in the framework of fault detection. Normally, a model identified based on a finite training set that has been collected a priori, will, unless the training set is very rich, invariably exhibit some degree of modeling error. There may also be some effects, such as wear, that may cause the true system to change gradually with time without actually giving rise to faults. Hence, it may be advantageous to adopt some kind of online learning scheme. The main idea of using such an online adaptive scheme is to allow the model to adapt to slow changes or model mismatch in the system, but still observe faster and larger changes caused by faults. Several results have been published on fault detection using neural networks and other nonlinear system identification methods, for instance (Demetriou and Polycarpou, 1998). However, so far the tradeoff between detectability and learning in online identification of nonlinear dynamics has had to be selected in an ad hoc manner, see e.g. (Polycarpou and Trunov, 1998).

A more conservative approach is to allow a neural network to learn the behavior of the system, and then let it predict the system output over a period of time based on past measurements of system in- and outputs. If the system output and the corresponding network estimate deviate considerably from one another, it is highly likely that a fault has occurred in the system. If the network output only deviates slightly from the system output, on the other hand, it may be assumed that the network is not trained sufficiently well, and

[1] Partially supported by the ATOMOS IV project.

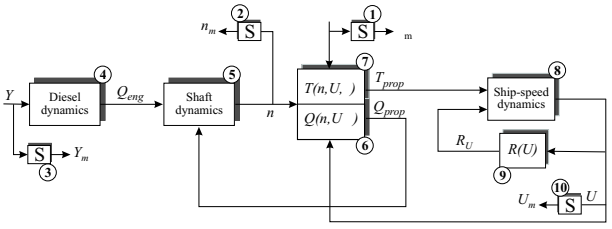

Fig. 1. Part of the ship propulsion system involving diesel, shaft, and ship dynamics and developed thrust and torque and hull characteristics.

the network can be improved through further offline training.

This paper will consider the problem of fault detection based on residuals generated by a model that has been identified from noisy data. However, in this case it can be quite hard to define a threshold for what constitutes a fault in a meaningful manner. In this paper, it will be shown that at least for certain types of neural network models, it is possible to estimate this threshold given bounds on disturbances on the input side, which can be attributed to e.g. structural faults, in a numerically tractable manner; in other words, it is possible to estimate how the input variations (due to faults) will propagate to the outputs. This information can then be used to choose appropriate threshold values.

2. SHIP PROPULSION BENCHMARK

The ship propulsion benchmark (Izadi-Zamanabadi and Blanke, 1999) is a simulation model that reproduces the non-linear behaviour of a ship's dynamics. The benchmark was developed to be used as a platform for development of new ideas for fault-tolerant control and also for comparison of different FDI methods. The main components/subsystems modeled in the ship benchmark are: diesel engine dynamics, shaft dynamics, propeller characteristics, ship speed dynamics, pitch angle and shaft speed controllers, and finally the coordinated control level, which gives setpoints for shaft speed and propeller pitch. To illustrate the method, only the torque-thrust related part of the benchmark is considered. An outline of this part is shown in Fig. 1, where the involved components are numbered. The components, denoted by $\mathsf{C}_i, i = 1,\ldots,10$, are: diesel engine dynamics C_4, shaft speed dynamics C_5, propeller torque characteristics C_6, propeller thrust characteristics C_7, hull characteristics C_9, and ship speed dynamics C_8. The related sensors are $\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3$, and C_{10}. The related relations (and governing equations) are listed below:

$$
\begin{array}{lll}
\mathsf{C}_1: & f_1(\theta,\theta_m)=0: & \theta=\theta_m \\
\mathsf{C}_2: & f_2(n,n_m)=0: & n=n_m \\
\mathsf{C}_3: & f_3(Y,Y_m)=0: & Y=Y_m \\
\mathsf{C}_4: & f_4(Y,K_y,Q_{\text{eng}})=0: & Q_{\text{eng}}+\tau_c\dot{Q}_{eng}=K_yY \\
\mathsf{C}_5: & f_5(Q_{\text{eng}},Q_{\text{prop}},n)=0: & I_m\dot{n}=Q_{\text{eng}}-Q_{\text{prop}} \\
\mathsf{C}_6: & f_6(n,\theta,U,Q_{\text{prop}})=0: & Table_Q \\
\mathsf{C}_7: & f_7(n,\theta,U,T_{\text{prop}})=0: & Table_T \\
\mathsf{C}_8: & f_8(U,T_{\text{prop}},R_U)=0: & m\dot{U}=R_U+t_TT_{\text{prop}} \\
\mathsf{C}_9: & f_9(U,R_U)=0: & Table_R \\
\mathsf{C}_{10}: & f_{10}(U,U_m)=0: & U=U_m
\end{array}
$$

θ denotes the propeller pitch, n and U denote shaft and ship speed, Y is the fuel index, and K_y is the engine gain. Q_{eng} is the engine torque and Q_{prop} and T_{prop} are torque and thrust developed by the propeller. R_U represents the hull characteristics. The thrust reduction number t_T represents losses in the thrust produced by the propellers. m, I_m, τ_c, and t_T are constant parameters. Q_{prop} and T_{prop} are functions of n, θ, and U, and R_U is a function of U, i.e. $Q_{\text{prop}} = Q_{\text{prop}}(n,\theta,U)$, $T_{\text{prop}} = T_{\text{prop}}(n,\theta,U)$, and $R_U = R(U)$. These are calculated by interpolating between data in $Table_Q$, $Table_T$ and $Table_R$, which represent the nonlinear behaviour of the propeller and hull characteristics obtained in model propeller test and sea trials.

Two faults (among the original four in (Izadi-Zamanabadi and Blanke, 1999)) are considered; fault in the shaft speed (Δn) measurement and the engine gain fault (ΔK_y). The shaft speed is measured by a dual pulse pick-up. EMI disturbances on one pick-up can generate a maximum signal Δn_{high}, while a minimum signal Δn_{low} is produced due to loss of both pick-up signals. A drop in generated shaft torque, manifested by ΔK_y, may happen due to the following causes: less (or hot) air inlet, less fuel oil inlet (due to leakage), degenerated cooling oil inlet, or drop-out of one or more cylinders. The consequences and the severity level of these faults are shown in table 1.

Table 1. Consequences and severity levels for the benchmark faults

Fault	Consequence	Severity
Δn_{high}	*deceleration* ⇒ *manoeuvering risk*	high
Δn_{low}	*acceleration* ⇒ *collision risk*	very high
Δk_y	*diesel overload* ⇒ *wear,slowdown*	medium

3. SYSTEM ANALYSIS

Performing an analysis of the system's structural model ((Cassar *et al.*, 1994) (Declerck and Staroswiecki, 1991), (Izadi-Zamanabadi, 1999), (Izadi-Zamanabadi and Blanke, 2002)) results in identifying the subsystems with inherent redundant information, which can then be used for FDI purposes. The following expressions are obtained:

$$f_1(\theta_m,U_m,n_m)=0 \quad (1)$$

$$f_2(\theta_m,U_m,n_m,K_y,Y_m)=0 \quad (2)$$

Each expression indicates that there is a relation, for instance f_1, which imposes a constraint on the involved variables (in this case θ_m, n_m, and U_m) under

normal conditions (i.e., no-fault situations), so that the value of each variable cannot be changed independently of the other involved variables values. This means that there exists a certain invariant structure between the involved variables.

Both expressions (1) and (2) can be used directly for fault detection and isolation. However, both residuals obtained from these expressions will be affected by faults in the shaft speed measurement. In (Izadi-Zamanabadi and Blanke, 2002) a structural method is used to obtain residual expressions that can be directly used to detect gain fault independent of the shaft speed measurement fault. The obtained expression has the following form:

$$f(\theta_m, U_m, Y_m, K_y) = 0 \tag{3}$$

Taking system causality into considerations, the invariant structure between the involved variables can be captured by a neural network defined by:

$$\hat{U} = \hat{f}_{NN}(\theta_m, K_y Y_m) \tag{4}$$

A residual can be defined as

$$r = U_m - \hat{U} \tag{5}$$

Obviously, this residual will be affected by a fault in the engine gain, while the shaft speed fault has no direct effect on the residual.

The next issue that will be considered, is the determination of an appropriate threshold value, which is dependent on the allowable variation in system parameters and/or measurements.

4. NEURAL NETWORK MODELING

The type of neural networks that will be employed in this paper for the modeling task, is the so-called multilayer perceptron (MLP). It is assumed that the actual system can be written on the general form

$$x_{k+1} = f(x_k, u_k), \quad y_k = Cx_k \tag{6}$$

in which k is the discrete-time sample number, $x \in \mathbb{R}^{n_s}$ is an appropriately chosen state vector, $u \in \mathbb{R}^m$ is a vector containing external inputs and $y \in \mathbb{R}^p$ is the output. As was proven in (Hornik *et al.*, 1989), an MLP with one hidden layer can approximate any continuous function to any desired degree of accuracy, assuming that a sufficiently rich training set is available and the number of neurons is sufficiently large. The following recurrent neural network will thus be used to identify the nonlinear mapping

$$\hat{x}_{k+1} = W_2 \sigma(W_1 \zeta_k + W_b), \quad \hat{y}_k = C\hat{x}_k \tag{7}$$

$$\varepsilon_k = y_k - \hat{y}_k \tag{8}$$

where $\hat{y}_k \in \mathbb{R}^p$ denotes the prediction of the actual output vector $y_k \in \mathbb{R}^p$ at sample k, and ε_k is the prediction error to be minimized. $\zeta_k \in \mathbb{R}^{n_\zeta}$ is a vector of inputs to the network. $W_1 \in \mathbb{R}^{q \times n_\zeta}$, $W_b \in \mathbb{R}^q$ and $W_2 \in \mathbb{R}^{n_s \times q}$ are matrices containing the weights and biases of the network. The neuron function $\sigma(\cdot) : \mathbb{R}^q \to \mathbb{R}^q$ is a continuous, diagonal, monotonously increasing nonlinear mapping with $\sigma(0) = 0$; these requirements are for instance satisfied by the hyperbolic tangent function. If the states in (6) are available for measurement, the most natural approach is to select $\zeta_k = [x_k^T \; u_k^T]^T$. If only output samples are available for measurement, on the other hand, it is possible to train the network using the choice of inputs $\zeta_k = [\hat{x}_k^T \; u_k^T \; \varepsilon_k^T]^T$. Since networks of the latter type involves delayed feedbacks from the network output to the input, they are known as *recurrent networks*, while networks of the first type is known as *non-recurrent networks*.

The Back Propagation Error Algorithm (see e.g., (Chen and Billings, 1992)), abbreviated BPEA, is used due to its simplicity and robustness. The BPEA is the most commonly used algorithm for training MLPs and can be briefly summarized as follows. The aim is to find the parameter vector Θ which minimizes the standard performance function over N samples,

$$J(N, \Theta) = \frac{1}{2N} \sum_{k=1}^{N} \varepsilon_k^T \varepsilon_k$$

Θ is a column vector containing all the MLP weights, arranged such that the weights of the first neuron in the hidden layer come first, then the weights and biases of the second neuron etc., ending with the output weights of the last neuron. The minimization is achieved using so-called *sample updating*. Assume a measurement y_k is available at sample number k. After calculating $\hat{x}_k$,$\hat{y}_k$, and ε_k according to (7)–(8), the parameters are updated in the direction of the negative gradient of the instantaneous performance functional

$$J_k = \frac{1}{2} \varepsilon_k^T \varepsilon_k$$

according to the parameter update rule

$$\Theta_k = \Theta_{k-1} - \eta \frac{dJ_k}{d\Theta_{k-1}} = \Theta_{k-1} + \eta \psi_k C^T \varepsilon_k \tag{9}$$

η is a step length which dictates how large the parameter updates are. The model gradient $\psi_k = d\hat{x}_k^T / d\Theta_{k-1}$ is the gradient of the network state estimate with respect to the parameter vector. It thus depends on the chosen neural model structure, neuron function, etc. In case of non-recursive networks the calculation of ψ_k is fairly straightforward, while for recurrent networks it is necessary to calculate recursive estimates of the derivatives of the network state estimate with respect to the fed back signals as well. Refer to for instance (Sarkar, 1995) for further information on training MLPs using the BPEA.

Now consider a model described by (7)–(8) and assume that the neural network model has been sufficiently well trained to generate a prediction error

which yields a satisfactory performance.[2] Then it can be estimated how large an effect an input perturbation will have on the network output, due to the following lemma.

Lemma 1. Consider the bounded input perturbation $\delta \in [-\bar{\delta}; \bar{\delta}] \subset \mathbb{R}$ with $\bar{\delta} > 0$. If this perturbation only affects a single input channel of a neural network described by (7), all possible network outputs resulting from δ belong to a convex set.

Proof: The perturbed input can be written as $\tilde{\zeta}_k = \zeta_k + \delta^i$, $\delta^i = \delta e_i$, where $e_i \in \mathbb{R}^{n_\zeta}$ is the i'th unit vector. Denote the output from the neuron function (the *hidden layer* in ANN terminology) by $\xi \in \mathbb{R}^q$ and the perturbed output from the hidden layer by $\tilde{\xi} \in \mathbb{R}^q$. The input perturbation yields the following:

$$\tilde{\xi} = \sigma(W_1\tilde{\zeta}_k + W_b) = \sigma(W_1\zeta_k + W_b + w_1^i\delta)$$

where w_1^i is the i'th column vector of W_1. Looking at the j'th neuron, $j = 1, \ldots, q$, it is then noted that

$$\begin{aligned}\sigma_j(W_1\zeta_k + W_b - |w_{1,j}^i|\bar{\delta}) &\le \sigma_j(W_1\zeta_k + W_b + w_{1,j}^i\delta) \\ &\le \sigma_j(W_1\zeta_k + W_b + |w_{1,j}^i|\bar{\delta})\end{aligned} \tag{10}$$

for all $\delta \in [-\bar{\delta}; \bar{\delta}]$, since $\sigma_j(\cdot)$ is a monotonously increasing scalar function. Let us define the vectors $\bar{\xi}^\kappa = [\xi_1, \ldots, \xi_q]^T, \kappa = 1, \ldots, 2^q$, where the scalar entries ξ_l is given by either $\xi_l = \sigma_l(W_1\zeta_k - |w_{1,l}^i|\bar{\delta})$ or $\xi_l = \sigma_l(W_1\zeta_k + |w_{1,l}^i|\bar{\delta})$ for $l = 1, \ldots, q$. Now, from (10) and the continuity of $\sigma_j(\cdot)$, it is deduced that all perturbed output vectors $\tilde{\xi}$ are contained in the set

$$\Xi = \left\{\xi | \xi = \sum_{\kappa=1}^{2^q} \alpha_\kappa \bar{\xi}^\kappa, \sum_{\kappa=1}^{2^q} \alpha_\kappa = 1\right\} \subset \mathbb{R}^q \tag{11}$$

i.e., the polygon formed by all convex combinations of $\bar{\xi}^\kappa$. Then it is immediately seen that, since W_2 is a linear mapping from $\mathbb{R}^q$ to $\mathbb{R}^{n_s}$, the set $\mathscr{X}(\zeta_k, \bar{\delta}) = \{x | x = W_2\xi, \forall \xi \in \Xi\} \subset \mathbb{R}^{n_s}$ is a convex set as well. $\diamondsuit$

This tells us that, for a given pair $(\zeta_k, \bar{\delta})$, we can find bounds on the output estimate in a numerically tractable way, since all output estimates are contained in the polygon in the output space given by the 'vertices' $\bar{y} = CW_2\bar{\xi}^\kappa$. If, for instance, y is scalar, the interval in which the perturbed output estimate can end up, is simply found by taking the maximum and minimum value of these 2^q values.

5. STRATEGY FOR APPLYING NEURAL NETWORKS IN FAULT DETECTION

The dynamics of the considered system is assumed to be mainly time invariant. However, the method can also be used on systems with very slow varying dynamics. The basic strategy is as follows:

The neural network will be trained and validated with sets of non-faulty data covering the complete range of operation (to the extent possible). It is assumed that some small deviation from the range of operation is allowed in order to take the possibilities for parameter variations in the system into consideration. These allowable deviations, if not learned, have an impact on the residual. The strategy is to register these deviations over a given time interval. When the mean value of the residual stays within a predetermined boundary, but still differs from zero, then this indicates that some adjustment to (or updating of) the neural network weights is required. The approach is then to update the network online with the new set of data (gathered during the specified time interval). If the plant had been completely static and perfectly known, a fault would yield a residual for which an indicative threshold could be estimated off-line. However, since the weights are allowed to adapt to slow changes in the nonlinear plant or slight model mismatch, it is difficult to determine a priori what effect a fault will have on the output. This necessitates the recalculation of the threshold value at the given operation point, as outlined in the previous section.

Obtaining useful results requires the residual to be insensitive to the variations in the input signals. This requirement is similar to the requirement for the analytical case (Isidori *et al.*, 2000). However, the difference is that the requirement in this case concerns insensitivity w.r.t. *variations* in the input signals and not w.r.t. the input signals. This requirement arises from the fact that the trained neural network behaves as a static model. Thus, it cannot reproduce exactly the same output as the system during the transient states. The generated residual will hence differ from zero (in mean value) when the neural network is subjected to (fast) variations in the inputs. To cope with this problem, the rate change r_i in the individual inputs, i.e.

$$r_i = \frac{\Delta u_i}{T_s}, \qquad i = 1, ..., n_s$$

where T_s is the sampling time, is computed. The obtained residual will not be evaluated when the value of the rate change differs from zero. When noisy inputs are used, as in the case considered here, a dedicated adaptive CUSUM algorithm is designed to detect the rate change in the inputs. The adaptive part is designed to ensure that the residual is not affected by unmodeled dynamics generated in the transient state. The algorithm is described below:
The cumulative sum value (Basseville and Nikiforov, 1994) at time k is

$$s_k = \frac{1}{2\sigma^2}\left((|r_{i,k}| - \mu_0)^2 - (|r_{i,k}| - \mu_1)^2\right) \tag{12}$$

where σ^2 represents the variance of the noise on r_i. μ_0 and μ_1 represent the expected value of r_i in the

[2] The network is considered sufficiently trained when the performance function reaches a preset value, say, $J(N, \Theta) \le \bar{J}$, and the auto-correlation of ε_k is sufficiently close to that of white noise

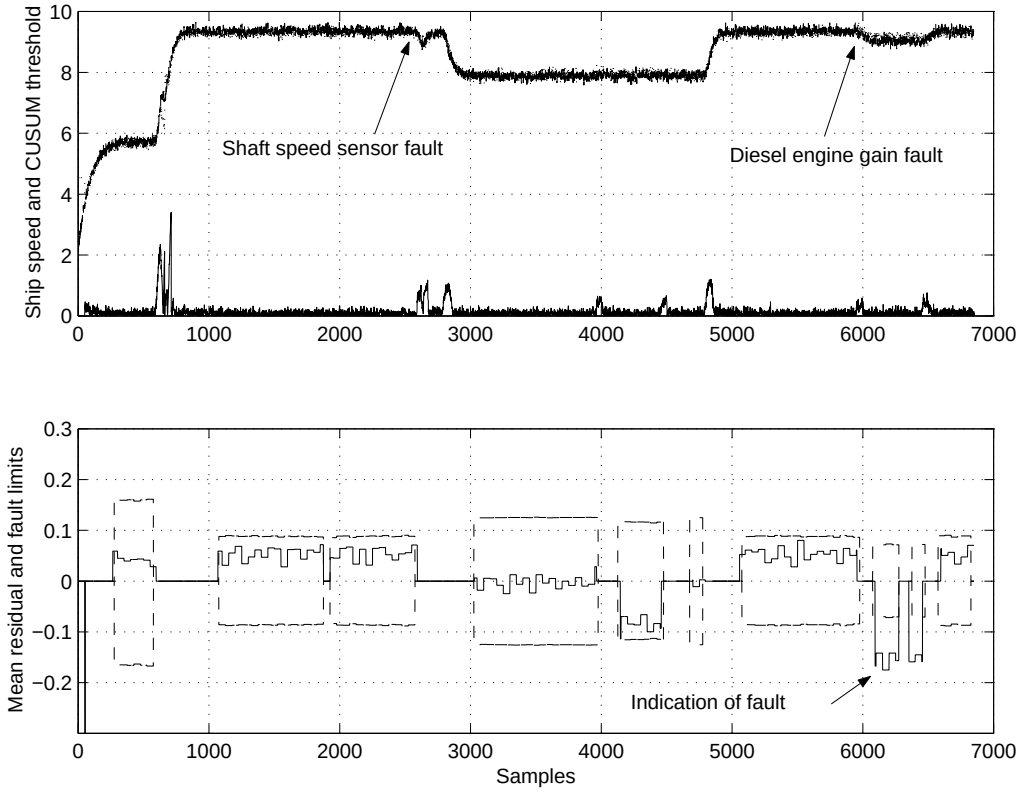

Fig. 2. Top: U_m (–), $\hat{U}(\cdots)$ and CUSUM threshold value (–); Bottom: 50-sample mean values of r (–) and y_{max}, y_{min} (- -).

stationary and non-stationary case. The absolute value of the rate change is used to cover both negative and positive input variations. A decision function g is defined as

$$g_k = g_{k-1} + s_k \tag{13}$$

Two separate identifiers are used: id_b classifies the cases where the input variations occur and id_e is used to indicate that these variations cease to occur. The identification part of these cases as well as corresponding actions are performed in a sequential manner in the following

$$\textbf{If} \quad \begin{cases} g_k \geq Th \\ id_e = 0 \end{cases} \quad \textbf{then} \quad \begin{cases} g_k = Th \\ id_b = 1 \\ id_e = 1 \end{cases} \tag{14}$$

$$\textbf{If} \quad \begin{cases} g_k < Th \\ id_e = 0 \end{cases} \quad \textbf{then} \quad \begin{cases} g_k = 0 \\ id_e = 0 \end{cases} \tag{15}$$

$$\textbf{If} \quad \{ id_e = 1 \quad \textbf{then} \quad \{ g_k = g_{k-1} + \gamma s_k \tag{16}$$

The value of the threshold Th is determined by the designer. The filter in eqn. (16) ensures a smooth and slow (as determined by the value of γ) change of the decision function from Th back to zero. It hence ensures that the residual is not affected by the unmodelled dynamics in transient states.

6. SIMULATION EXAMPLE OF FAULT DETECTION

An MLP model with ten neurons in the hidden layer was trained based on 12000 samples collected from the ship propulsion process with a sample period of 1 second. The in- and outputs to the network was chosen as indicated in (4) supplemented by an extra input, the ship speed measurement delayed by 25 samples. This extra input was included in order to prevent the network estimate from drifting slowly away from the measured ship speed.

After the model was trained, another simulation (6900 samples) was carried out, where the trained MLP predicted the ship speed at each sample. The adaptive CUSUM algorithm explained in Section 5 was also applied, and whenever the adaptive threshold indicated that the inputs were not changing significantly the vertices corresponding to a 10% disturbance in $K_y Y_m$ were calculated. To reduce the effects of the measurement noise, these values were calculated based on mean values of the network inputs over periods of 50 samples at a time.

The top plot in Figure 2 shows the measured and estimated ships speed along with the adaptive CUSUM threshold value. Two faults were made to occur during the simulation, a shaft speed sensor fault at sample number 2800 and a diesel engine gain falt at sample number 6000. The effects of each fault was removed again after approximately 500 samples. The bottom

plot shows the residuals and the upper and lower fault limits indicated by the network. These values are set to zero whenever the CUSUM threshold indicated that the inputs were varying significantly, thereby preventing the fault indication at the speed sensor fault. The diesel engine gain fault, on the other hand, is clearly indicated, as the residual is below the lower limit value in the period from 6000 to 6500 samples.

7. DISCUSSION

In this paper an approach to employing artificial neural networks for nonlinear fault detection and isolation was examined. The approach relies on training a multi-layer perceptron network to predict system outputs based on measurements taken from the plant. This output prediction allows the generation of a residual, which, if a fault occurs, will exceed an appropriately chosen threshold value. As the system is nonlinear, this threshold can be expected to depend on the local operating point, and may therefore be hard to specify in advance. The main contribution of this paper was therefore to devise a numerically tractable way of estimating this threshold, given a fault of a given size on the input side. The idea is that it is often easier to quantify the effects of a fault in the same subsystem or component where it occurs, for instance an engine, than to quantify its effects on a measured output a priori, such as the overall vehicle speed. The threshold estimation approach was used for fault isolation purposes in conjunction with the so-called adaptive CUSUM method, which was also reviewed in the paper. By applying the CUSUM method, it is possible to remove the influence from dynamics caused by external influences. It is still possible to detect changes in the internal structure, however, since their influence on the output can be considered equivalent to input disturbances.

The principle was illustrated on a ship propulsion simulator. After a structural analysis of the system, a neural network was trained to estimate the speed of the ship based on measurements of the fuel consumption, propeller pitch angle and old measurements of the ship speed. The simulation showed that it was possible to detect and isolate the gain fault in the diesel engine from a propeller shaft speed sensor fault.

Finally, the approach opens up for possibilities for letting the neural network train online, in case there are areas in the operating range where the network has not been trained sufficiently well. If the mean value of the residual is nonzero, but not large enough to be considered to caused by a fault, the network can be trained online to learn the new behavior of the plant.

8. REFERENCES

Basseville, M. and I. Nikiforov (1994). *Statistical Change Detection*. Prentice Hall.

Cassar, J. Ph., R. G. Litwak, V. Cocquempot and M. Staroswiecki (1994). Approche structurelle de la conception de systèmes de surveillance pour des procédé industriels complexes. *Revue Européenne de Diagnostic et Sûreté de fonctionnement*. Vol.4, no. 2, pp. 179-202.

Chen, S. and S. A. Billings (1992). Neural networks for nonlinear dynamic system modelling and identification. *International Journal of Control* **56**(2), 319–346.

Declerck, P. and M. Staroswiecki (1991). Characterization of the canonical components of a structural graph for fault detection in large scale industrial plants. In: *Proceedings of ECC'91*. Grenoble, France. pp. 298–303.

Demetriou, M. A. and M. M. Polycarpou (1998). Incipient fault diagnosis of dynamical systems using online approximators. *IEEE Transactions on Automatic Control* **43**(11), 1612–1617.

DePersis, C. and A. Isidori (2000). A geometric approach to nonlinear fault detection and isolation. In: *IFAC Safeprocess2000*. pp. 209–214.

Frank, P. M., G. Schreier and E. A. García (1999). *New Directions in Nonlinear Observer Design*. Chap. Nonlinear Observers for Fault Detection and Isolation, pp. 399–422. Springer-Verlag, London.

Hammouri, H., M. Kinnaert and E. H. Yaagoubi (1999). Observer-based approach to fault detection and isolation for nonlinear systems. *IEEE Trans. Automat. Contr.* **44**(10), 1879–1884.

Hornik, K., M. Stinchcombe and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* **2**(5), 359–366.

Isidori, A., M. Kinneart, V. Cocquempot, C. De Persis, P. M. Frank and D. N. Shields (2000). *Residual Generation for FDI in Non-linear Systems*. Springer.

Izadi-Zamanabadi, R. (1999). Fault-tolerant Supervisory Control - System Analysis and Logic Design. PhD thesis. Dept. of Control Eng., Aalborg University, Denmark.

Izadi-Zamanabadi, R. and M. Blanke (1999). A ship propulsion system as a benchmark for fault-tolerant control. *Control Engineering Practice* **7**(2), 227–239.

Izadi-Zamanabadi, R. and M. Blanke (2002). Structural analysis for diagnosis with application to ship propulsion problem. In: *Submitted to IFAC World Congress 2002*.

Patton, R. J. (1997). Fault tolerant control: The 1997 situation. In: *IFAC Safeprocess'97*. Hull, United Kingdom. pp. 1033–1055.

Polycarpou, M. M. and A. B. Trunov (1998). Learning approach to nonlinear fault diagnosis: Detectability analysis. *IEEE Transactions on Automatic Control* **45**(4), 806–812.

Sarkar, D. (1995). Methods to speed up error back-propagation learning algorithm. *ACM Computing Surveys* **27**(4), 519–542.