A neural network based approach to sensor fault detection

Abhay Bulsari[†], Alexander Medvedev[*] and Henrik Saxén

Kemisk–tekniska fakulteten, Åbo Akademi
SF 20500 Turku/Abo, Finland.
Tel: 358 (21) 654 311
Fax: 358 (21) 654 479
E-mail: vt_ai@abo.fi

ABSTRACT

This work investigated the a new approach, combining two recently developed techniques to detect sensor faults in dynamic systems, exemplified by an application to an industrially vital biochemical process.

State vector estimation, the first part of the sensor fault detection procedure, is based on delayed measurements of the control and output vectors. It assumes linear dynamics and incorporates a priori information regarding system dynamics. Observer residuals are calculated as the difference between the two estimations of the plant state vector, one excluding one or more last measurements, and the other based on the all the output measurements available for the estimator at each moment. These residuals are fed to a feed–forward neural network, which is trained to recognise which particular sensor(s) is/are faulty.

Levenberg–Marquardt method was used to train the networks by error square sum minimization. The method is known to be fast, accurate and reliable. The output of each node was calculated using the logistic (sigmoid) activation function on the weighted sum of inputs to that node. The networks are trained for single as well as multiple faults in sensors. The biases were very small, indicating a good choice of input variables and training instances.

The linearised third order dynamic model was used for the state vector estimator. The neural networks had three inputs since there were three state variable estimates, and had three outputs used as fault indicators for each of the three sensors. The most appropriate choice for the network configuration was found to be (3,17,3) based on the training experience with 707 training instances.

The trained neural networks successfully identified single and multiple faults during the testing phase, when they were fed with data which they were not trained on. This work demonstrated that the technique works reliably for the problem considered in this paper. This technique can be implemented in real time systems since the computing power required is quite low, once the network is trained.

[†] on leave from Lappeenranta University of Technology
[*] on leave from Leningrad Electrical Engineering Institute

1. Introduction

Failures in a system are detectable if the outputs following the failure are different from the outputs prior to the failure. Therefore, the process history should be compared with the current plant state to detect possible system malfunctions. Such comparison can be performed by using delayed-data driven observer which computes a number of estimates – one based on process history and the other based on process history and more recent measurement data.

If the recent measurements do not fit the values predicted from process history, then they are most likely provided by faulty sensor readings.

Another objective to be achieved is fault isolation, *i.e.* a procedure of differentiating between failures which may occur in a system. Fault isolation procedure requires a substantial amount of logical operations due to decision making process involved. This task can be successfully completed by artificial neural networks.

Artificial neural networks (ANNs) offer interesting possibilities in chemical and biochemical engineering. Various configurations and topologies of ANNs have been suggested [1] . ANNs, also called connectionist models, parallel distributed models, neuromorphic systems or simply neural nets consist of many simple computational elements called nodes, or neurons each of which collects by weighted addition the signals from various other nodes which are connected to it directionally. This sum, the net input to the node, is processed by a function (usually a sigmoid or a step) resulting in the output or the activation of that node. In multilayer feed–forward networks of the type considered in this paper (Fig. 1), there are a few layers (the input layer, the output layer, and possibly some hidden layers) across which all the nodes of each layer are connected to all the nodes in the layer above it, but there are no connections within the layer. A constant term, a bias, is usually added to each of the nodes.

Although back propagation has become popular on grounds of simplicity and its capability to learn sequentially from training instances, we have used the Levenberg–Marquardt method [2, 3] for minimising the sum of squares of errors. Levenberg–Marquardt method is fast, reliable and easy to use when the number of weights is not very large, say less than 100. It has been successfully used for a system identification task without dynamics [4] .

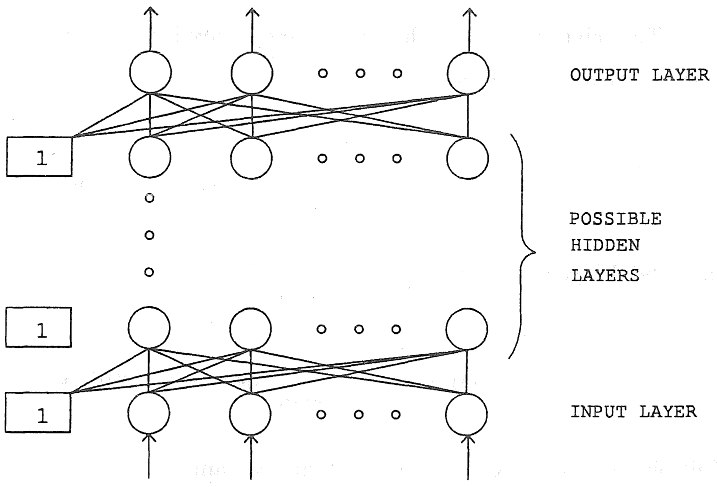

Figure 1. Feed–forward neural network

2. Deterministic Least Squares Estimator

In this section, we address the problem of linear dynamic system state vector reconstruction in presence of sensor failures. Let us consider the nonhomogeneous equation

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y_i &\triangleq y(t-\tau_i) = C_i x(t-\tau_i), \qquad i = 0, 1, \ldots, k \end{aligned} \tag{2.1}$$

where $x(t) \in R^n$ is the state vector, $u(t) \in R^m$ is the control vector, $y(t) \in R^l$ is the observation vector, and y_i are delayed observation vector values. A, B, C_i are real matrices of appropriate dimensions.

The relation between the current and delayed state vector values is given by the state-space equation

$$x(t) = \exp(A\tau_i)x(t-\tau_i) + \int_{t-\tau_i}^{t} \exp(A(t-\vartheta))Bu(\vartheta)d\vartheta \tag{2.2}$$

Then the delayed state vector $x(t-\tau_i)$ is as follows

$$x(t-\tau_i) = \exp(-A\tau_i)x(t) - \int_{t-\tau_i}^{t} \exp(A(t-\tau_i-\vartheta))Bu(\vartheta)d\vartheta \tag{2.3}$$

Multiplying (2.3) by C_i from the left and rearranging gives

$$C_i \exp(-A\tau_i)x(t) = y_i + \int_{t-\tau_i}^{t} C_i \exp(A(t-\tau_i-\vartheta))Bu(\vartheta)d\vartheta$$

If the linear system (1.1) output is measured $k+1$ times, then the state-vector could be found through solution of linear equation

$$W_k x(t) = Y_k \tag{2.4}$$

where $W_k \in R^{(k+1)l \times n}, Y_k \in R^{(k+1)l \times 1}$ are real matrices

$$W_k = \begin{pmatrix} C_0 \exp(-A\tau_0) \\ \cdot \\ \cdot \\ \cdot \\ C_k \exp(-A\tau_k) \end{pmatrix},$$

$$Y_k = \begin{pmatrix} y(t-\tau_0) + \int_{t-\tau_0}^{t} C_0 \exp\left(A(t-\vartheta-\tau_0)\right) B u(\vartheta) d\vartheta \\ \cdot \\ \cdot \\ \cdot \\ y(t-\tau_k) + \int_{t-\tau_k}^{t} C_k \exp\left(A(t-\vartheta-\tau_k)\right) B u(\vartheta) d\vartheta \end{pmatrix}$$

It is possible to solve equation (2.4) in the least-squares sense so that

$$(Y_k - W_k x)^T R^{-1} (Y_k - W_k x)$$

is minimized. The R is arbitrary with $R^{-1} = Q^T Q$ a positive-definite weighting matrix. The least-squares solution $\hat{x}_k$ is given by

$$\hat{x}_k(t) = (QW_k)^+ QY \tag{2.5}$$

where $(\cdot)^+$ denotes Moore-Penrose pseudoinverse of a matrix.

If $\mathrm{rank}(W_k) = n$ and R is unit matrix then from (2.4),

$$\hat{x}_k(t) = (W_k^T W_k)^{-1} W_k^T Y \tag{2.6}$$

The more natural form of the existence condition for analytical solution of (2.4) is the positive definiteness of the square real matrix $W_k^T W_k$;

$$\mathrm{rank}(W_k) = \mathrm{rank}(W_k^T W_k) = \mathrm{rank}(\sum_{i=0}^{k} (\exp(-A^T \tau_i) C_i^T C_i \exp(-A\tau_i))$$

or, in terms of nonsingularity

$$\det(\sum_{i=0}^{k} (\exp(-A^T \tau_i) C_i^T C_i \exp(-A\tau_i))) \neq 0 \tag{2.7}$$

After multiplication of block matrices in (2.6) the estimate computed from $k+1$ sample output values $i = 0, \ldots, k$ can be rewritten in the form

$$\hat{x}_k(t) = \mathcal{W}_k^{-1} \sum_{i=0}^{k} \exp(-A^T \tau_i) C_i^T (y(t-\tau_i) + \int_{t-\tau_i}^{t} C_i \exp(A(t-\delta-\tau_i)) Bu(\delta) d\delta) \qquad (2.8)$$

where

$$\mathcal{W}_k = W_k^T W_k = \sum_{i=0}^{k} \exp(-A^T \tau_i) C_i^T C_i \exp(-A\tau_i)$$

3. Fault Detection Window

Let us consider the application of the observation algorithm (2.8) to sensor fault detection, using time-redundancy principle. The main idea is the comparison of two estimates: one based on process output history, excluding r last measurements, and the other based on the all k output measurements available for the estimator at each moment. The r value will be refered as fault detection window (which lasts actually from $t - \tau_{k-r}$ to t), because it contains data suspected to be provided by a faulty sensor.

We assume that there are no sensor failures before the moment $t - \tau_{k-r}$. Our purpose is to detect a fault and a faulty sensor inside the fault detection window by means of algorithm (2.8).

Define $\hat{x}_k(t)$ as the estimate computed from $k+1$ sample output values $i = 0, \ldots, k$ and $\hat{x}_{k-r}(t)$ the estimate computed from $k-r$ sample output values $i = 1, \ldots, k-r$. Taking into account matrices partition, one may rewrite (2.4) as

$$\begin{pmatrix} W_{k-r} \\ W_r \end{pmatrix} x(t) = \begin{pmatrix} Y_{k-r} \\ Y_r \end{pmatrix} \qquad (3.1)$$

Using partitioned matrices notation for the corresponding estimates, we get

$$\hat{x}_{k-r} = W_{k-r}^{+} Y_{k-r}$$

$$\hat{x}_k = W_k^{+} Y_k$$

The well-known formula for the partitioned matrix pseudoinverse (Boullion and Odell, 1971) [10] gives the estimate in terms of submatrices

$$\hat{x}_k = (W_{k-r}^+ - T_r W_r W_{k-r}^+ ,\ T_r) Y_k$$

where

$$T_r = F_r^+ + (E - F_r^+ W_r) W_{k-r}^+ {W_{k-r}^+}^T W_r^T K_r (E - F_r F_r^+)$$
$$K_r = \left(E + (E - F_r F_r^+) W_r W_{k-r}^+ {W_{k-r}^+}^T W_r^+ (E - F_r F_r^+)\right)^{-1}$$
$$F_r = W_r (E - W_{k-r}^+ W_{k-r})$$

Eventually, the least-square solution for (3.1) is

$$\hat{x}_k = (W_{k-r}^+ - T_r W_r W_{k-r}^+) Y_{k-r} + T_r Y_r \tag{3.2}$$

The whole information about possible sensor fault contains in the difference between the reliable estimate $\hat{x}_{k-r}$ and the estimate $\hat{x}_k$ which includes also unreliable data

$$\varepsilon_r = \hat{x}_{k-r} - \hat{x}_k = T_r (W_r W_{k-r}^+ Y_{k-r} - Y_r) = T_r (W_r \hat{x}_{k-r} - Y_r) \tag{3.3}$$

Now it is easy to see that the ε_r vector demonstrates how fault detection window data fits process history. If $\mathrm{rank}(W_{k-r}) = n$ then the equation for T_r can be considerably simplified

$$F_r = 0$$
$$K_r = \left(E + W_r W_{k-r}^+ {W_{k-r}^+}^T W_r^+\right)^{-1}$$
$$T_r = W_{k-r}^+ {W_{k-r}^+}^T W_r^T K_r$$

Moreover, pseudoinverse operation for the W_{k-r} matrix can be substituted by $W_{k-r}^+ = (W_{k-r}^T W_{k-r})^{-1} W_{k-r}^T$. Now it follows that

$$W_{k-r}^+ {W_{k-r}^+}^T = (W_{k-r}^T W_{k-r})^{-1} = W_{k-r}^{-1}$$

and finally the difference between estimates for the fault detection window width r is

$$\begin{aligned}\varepsilon_r &= T_r(W_r\hat{x}_{k-r} - Y_r)\\ T_r &= W_{k-r}^{-1}W_r^T(E + W_rW_{k-r}^{-1}W_r^T)^{-1}\end{aligned} \tag{3.4}$$

Then, if all sensors are operating properly the estimate $\hat{x}_{k-r}$ and the estimate $\hat{x}_k$ will be nearly identical. If, however, a sensor fault occurs at the moment inside the fault detection window, the ε_r value would be of sufficient scale when the output change caused by this sensor fault is noticeable. To provide more information for decision making under fault detection/isolation procedure a number of ε_i can be generated. Since all matrix coefficients $T_i, i = 1, \dots, r$ are independent from the current measurements and do not use any additional information in comparison with the estimate $\hat{x}_k$, the set of $\varepsilon_i, i = 1, \dots, r$ seems to be reasonable choice.

4. The feed–forward neural network

The feed–forward neural network has an input layer which simply transmits the input variables without any processing to the next layer. The bias is a weighted unit input to each node, thus adding a constant term to the net input of the node. Two, one or no hidden layers are considered. Each node in the upper layers (hidden or output layers) receives weighted inputs from each of the nodes in the layer below it. These weighted inputs are added to get the net input of the node, and the output x_i (or the activation) is calculated by an activation function of the net input a_i as

$$a_i = \sum_{j=0}^{N} w_{ij}x_j \tag{4.1}$$

$$x_i = \frac{1}{1 + e^{-\beta a_i}} \tag{4.2}$$

where N is the number of nodes in a hidden layer or the number of input nodes and w_{i0} is the bias of node i. β is referred to as the gain term and is usually set to 1.

5. The biochemical process

Biochemical processes have highly non–linear characteristics, and have operability in limited domains. In the process considered here, *Saccharomyces cerevisiae*, a yeast, is grown on glucose substrate in a chemostat (a biochemical continuous stirred tank reactor) producing ethanol as a product of primary energy metabolism. There are three state variables : microbial concentration, X; substrate concentration, S; and product concentration, P. The kinetic and stoichiometric parameters were taken from a recent study on kinetics of this system [5] . The feed to the chemostat is sterile, *i.e.* there are no microorganisms in the feed. The feed concentration of substrate, glucose, is S_0, and D is the dilution rate (volumetric flow rate per volume of the chemostat.) See Fig. 2. The dynamics of this system can be described by the following equations.

$$\frac{dX}{dt} = (\mu - D)X \tag{5.1}$$

$$\frac{dS}{dt} = D(S_0 - S) - Y_{S/X}\mu X \tag{5.2}$$

$$\frac{dP}{dt} = -DP + Y_{P/X}\mu X \tag{5.3}$$

where the growth rate, μ and the yield coefficients, $Y_{S/X}$ and $Y_{P/X}$ are given by

$$\mu = \frac{0.427S}{0.245 + S}(1 - (P/101.6)^{1.95}) \tag{5.4}$$

$$Y_{P/X} = 3.436, \qquad Y_{X/P} = 0.291 \tag{5.5}$$

$$Y_{X/S} = 0.152(1 - P/302.3), \qquad Y_{S/X} = 1/Y_{X/P} \tag{5.6}$$

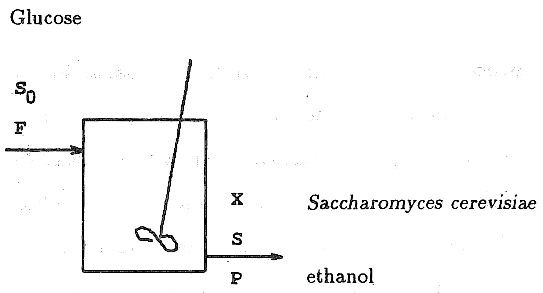

Figure 2. A bioreactor

The growth rate, μ cannot exceed 0.427, and that is also the upper limit for the dilution rate, D. If the dilution rate is higher than the growth rate, microbial concentration in the bioreactor decreases to zero. Table 1 shows typical steady state values for the process parameters.

5.1. Linearised equations of the process

This can be seen from the linearised set of equations at the operating parameters given in Table 1. Variables in small letters are the deviation from their corresponding operating parameters.

Table 1. Typical steady state values of the process parameters.

S_0	101.416 gm/lit
D	0.345 hr^{-1}
X	12.0 gm/lit
S	10.0 gm/lit
P	41.232 gm/lit
μ	0.345 hr^{-1}
$Y_{X/S}$	0.1313
$Y_{X/P}$	0.291

$$\begin{pmatrix} dx/dt \\ ds/dt \\ dp/dt \end{pmatrix} = \begin{pmatrix} 0 & -0.04075 & 0.00990 \\ 1.18542 & -0.48503 & 0.03402 \\ -2.62821 & 0.189656 & -0.42042 \end{pmatrix} \begin{pmatrix} x \\ s \\ p \end{pmatrix} + \begin{pmatrix} 0 & -12. \\ 0 & -41.232 \\ 0.345 & 91.416 \end{pmatrix} \begin{pmatrix} s_0 \\ d \end{pmatrix}$$

6. The Levenberg–Marquardt method

Levenberg–Marquardt method [2, 3] was used to determine the weights in the neural networks. Network training aims at minimising the sum of squares of errors, the errors measured as the difference between the calculated output and the desired output. Minimising the SSQ is not always the best way of training a neural network as pointed out in [4] , but for this application, it suffices. Back propagation by the generalised delta rule, a kind of a gradient descent method is one popular method [8] for training feed–forward neural networks. Many algorithms for least–squares optimization use Taylor–series models. The Levenberg–Marquardt method uses an interpolation between the approaches based on the maximum neighbourhood (a "trust region") in which the truncated Taylor series gives an adequate representation of the non–linear model. The method has been found to be quite efficient.

7. Training the neural networks

Neural networks were first trained for detecting just one fault. The training set consisted of 404 training instances. Each training instance includes three observer residuals followed by 1 for a faulty sensor, and 0 for a sensor in working condition. Initially, the whole state vector is supposed to be measureable. At the moment of a simulated sensor fault, a drastic change takes place in all residuals. which are the subject of the fault detection / isolation procedure performed by the neural network. An example of the observer residuals is shown in Fig. 3. The window width was $r = 1$ throughout this work.

The error square sum (SSQ) was 13 for (3,3), and for one hidden layered networks, it reduced to 12 when the number of hidden nodes was increased to (3,4,3). (3,8,3) and (3,9,3) resulted in a SSQ of 4, and (3,10,3) resulted in 0.01495. Two hidden layered networks were also considered, and the SSQ for (3,3,3,3) was 4. All these were trained with a β of 100. While most of the one hidden layered networks were trained using the program ANNEX [9] (with the Levenberg–Marquardt method), two hidden layered networks were often trained by simulated annealing, until the weights were close enough to the minimum, after which they were further improved by ANNEX.

The (3,10,3) was then tested for single faults on a set of data it was not trained on. It worked correctly for all the cases therein. It, however, failed to detect mulitple sensor faults, as was expected.

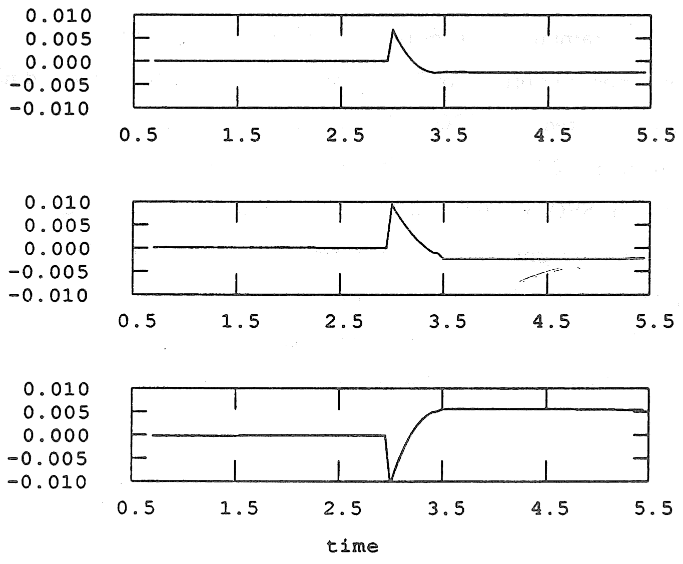

Figure 3. Observer residuals for the three state variables. Fault in sensors 2 and 3 occurring at t = 3.00

Therefore, neural networks were then trained for detecting just two faults. The 303 training instances consisted of faults in sensors 1 and 2, faults in sensors 2 and 3, faults in sensors 1 and 3, and no faults. The network without hidden layers (3,3) had a SSQ of 111.37. (3,5,3) resulted in a SSQ of 11.692, (3,8,3) in 4.035, and (3,11,3) in 1.000. The SSQ for (3,12,3) was 0.188×10^{-10}. This trained network correctly identified the faulty sensors after a small time lag, when the test cases had two faults, but failed to identify the faulty sensor when only one sensor was faulty. It was also observed that (3,11,3) performed better than (3,12,3). This kind of over-fitting of data is known to happen with feed-forward neural networks.

Finally, the training instances from the above two sets were put together in a training set, now comprising of 707 instances. This was fairly more difficult to train. (3,9,3) resulted in a SSQ of 51.41, and (3,10,3) in 24.56. There was little improvement in (3,14,3) with a SSQ of 24.40. However, the (3,15,3) and (3,16,3) resulted in SSQ of 6.000. (3,17,3) was found to have a SSQ of 2.000. This network could recognise 0, 1, or 2 correct faults in sensors. There is time lag of 10 sampling intervals before it can correctly point the faulty sensors, which is understandable because the algorithm for calculating ε requires 10 samples of measurements. More details of the testing are given in the next section.

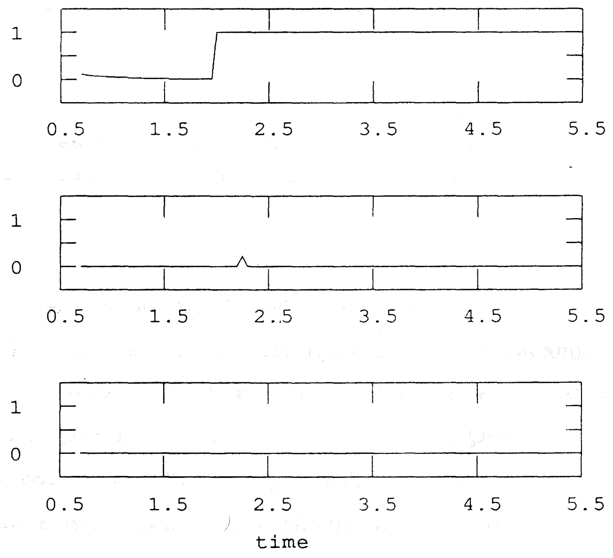

Figure 4. Detected faults in sensors 1, 2, 3 using the trained network (3,10,3)

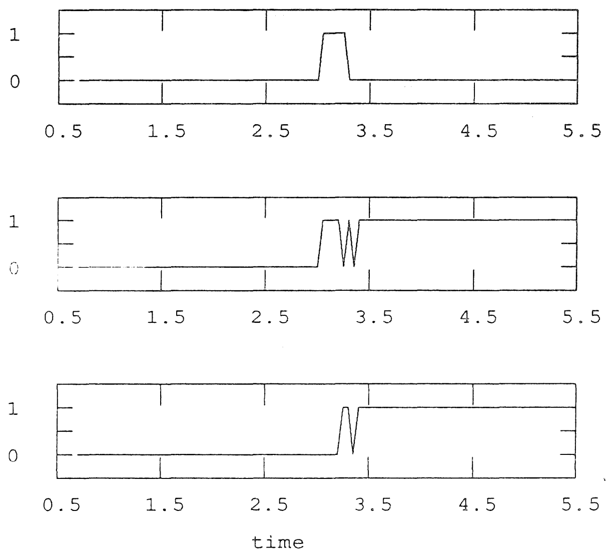

Figure 5. Detected faults in sensors 1, 2, 3 using the trained network (3,12,3)

8. Testing the neural networks

Networks trained for the detection of one faulty sensor could correctly detect the faulty sensor, when there was only one faulty sensor. Fig. 4 shows the results of the (3,10,3) neural network's output when one sensor (the first one) is faulty after $t = 2.00$ However, it could not correctly detect faults in two sensors simultaneously. Networks trained for detecting two faulty sensors could do it (see Fig. 5), but they could not detect the faulty sensor, when only one was faulty.

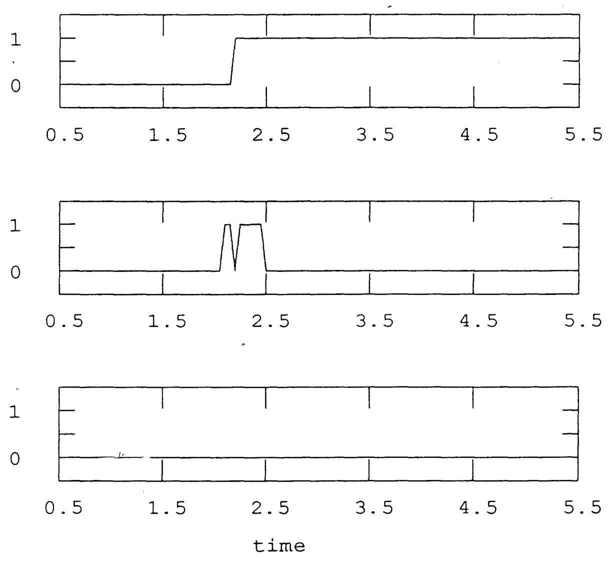

Figure 6. Detected faults in sensors 1, 2, 3 using the trained network (3,17,3)

The (3,17,3) network trained on 707 instances with 0, 1 or 2 faults could detect 0, 1 or 2 faults correctly after a time lag of about 10 sampling intervals. This is shown in Figs. 6 and 7. Fig. 6 shows the results of the neural network's output when one sensor (the first one) is faulty after $t = 2.00$, and each sampling interval is 0.05. Thus, the fault is recognised correctly after 10 sampling intervals. Similarly, Fig. 7 shows the results of the neural network's output when two sensors (2 and 3) are faulty after $t = 3.00$. Note that in both cases, the neural network output is consistently 0 when no fault is present.

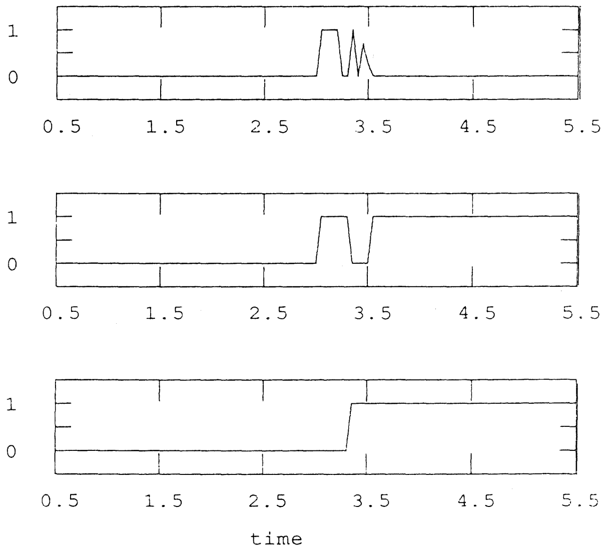

Figure 7. Detected faults in sensors 1, 2, 3
using the trained network (3,17,3)

9. Results and conclusions

The combination of using state vector estimation with neural networks proved to be effective. The estimator provides the neural network with substantial information on the sensor fault through the system dynamic model and a process history. It was shown that observer residuals are unique for each fault type. The neural network uses this information for fault detection and isolation. As shown by simulations, the proposed method successfully detected single and mulitple sensor faults in a linearised dynamic system.

For the third order biochemical system considered in this work, the network sizes required for detecting one faulty sensor, and one or two faulty sensors at a time were (3,10,3) and (3,12,3) respectively. The (3,10,3) network trained for one fault could not isolate two simultaneous faults, and the (3,12,3) trained for two faults coudl not reliably isolate a single fault. (3,17,3) was trained for, and could recognise 0, 1, or 2 faults. Neural networks were trained using the Levenberg–Marquardt method, which was once again found to be a good choice compared to the conventional back-propagation or its variants. Training is usually not so straightforward since there is always a possibility of getting into a local minimum. In most cases, very sharp sigmoids were used (with a gain of 100, looking almost like hard–limiter step functions), which would have made it very difficult for back–propagation–like methods to converge to reasonable solutions. Faults were detected almost at the point of occurrence, but there was usually some time lag before the fault(s) were isolated. This time lag was 10 delayed measurements or less, the amount required by the estimator.

References

1. Lippmann, R. P.,
   "An introduction to computing with neural nets",
   IEEE ASSP Magazine, (April 1987) 4-22.

2. Marquardt, D. W.,
   "An algorithm for least-squares estimation of nonlinear parameters",
   J. Soc. Indust. Appl. Math., 11 (June 1963) 431-441.

3. Levenberg, K.,
   "A method for the solution of certain nonlinear problems in least squares",
   Quart. Appl. Math., 2 (1944) 164-168.

4. Bulsari, A. and H. Saxén,
"Applicability of an artificial neural network as a simulator for a chemical process",
Proceedings of the fifth International Symposium on Computer and
Information Sciences, Nevsehir, Turkey (October 1990) 143-151.

5. Warren, R. K., G. A. Hill and D. G. Macdonald,
"Improved bioreaction kinetics for the simulation of continuous ethanol
fermentation by *Saccharomyces cerevisiae*",
Biotechnology Progress, 6 (1990) 319-325.

6. Medvedev, A.,
"Three approaches toward a fault tolerant controller structure",
Technical report 91–3, Reglerteknik, Åbo Akademi (1991)

7. Silverman, L. M. and M. Bettayeb,
"Optimal approximation of linear systems",
Proceedings of the Joint American Control Conference, (1980)

8. Jones, W. P. and J. Hoskins,
"Back-Propagation: A generalized delta learning rule"
Byte, (October 1987) 155-162.

9. Bulsari, A., B. Saxén and H. Saxén,
"Programs for feedforward neural networks using the Levenberg-Marquardt
method : Documentation and user's manual",
Technical report 90-2, Värmeteknik, Åbo Akademi, Finland (1990).

10. Bouillon, T. L. and P. L. Odell,
"Generalised inverse matrices",
Wiley Interscience, New York, (1971)