Fault Detection and Isolation of Three-tank System using Neurofuzzy Networks with Local Approaches

H. T. Mok and C. W. Chan

Department of Mechanical Engineering, The University of Hong Kong, Hong Kong
(e-mail: htmok@graduate.hku.hk; mechan@hku.hk)

Abstract: In this paper, a fault detection and isolation (FDI) scheme is derived based on fuzzy rules extracted from the neurofuzzy network that models the residual of the system. First, a fault database (FDB) is constructed from fuzzy rules extracted from the neurofuzzy networks that model all possible faults in the system. By comparing the currently extracted fuzzy rules with those in the FDB using the nearest neighbour classifier, faults are diagnosed online. As the number of rules in the FDB can be quite large, the FDI scheme proposed here utilises the local approaches to reduce the computation load and to improve the sensitivity of the method. The proposed FDI scheme is successfully applied to diagnose faults in a nonlinear three-tank control system.

1. INTRODUCTION

The main goal for fault detection and isolation (FDI) is to minimise fatal failures by closely monitoring possible faults in the systems (Isermann, 2005). Recently, more and more works are devoted to FDI based on neurofuzzy networks (NFNs), that incorporate the learning ability of neural networks and transparency of fuzzy logic (Patton *et al*., 2000; Zio and Gola, 2006). Online FDI schemes based on NFNs for nonlinear systems are also developed (Mok and Chan, 2008). However, a crucial factor in applying these techniques in practice is the curse of dimensionality problem in NFN, as computation time in training the network increases exponentially as the increase in the dimension of the network (Brown and Harris, 1994).

In this paper, the FDI scheme in Mok and Chan (2008) is extended by taking into account the local properties of NFNs in order to alleviate the computing problem and to increase the sensitivity of the FDI scheme. The motivation for the local approach is as follows. Since the input space of the NFN is partitioned by the B-spline basis functions (BSBFs) into subspaces, only a small number of univariate or multivariate BSBFs are activated each time. Consequently, information in the NFN is stored and learned locally (Hong and Harris, 2001). It follows that the operating region of the NFN can be considered to be a combination of a number of local operating regions. Another interesting property of NFNs is the local change property, arising from the compact support of the BSBFs (Chan *et al*., 2000). For a given input, only weights in the vicinity of the input are activated. This local property can be fully utilised to reduce the computing time for online training of NFNs. As the operating region of the NFN is now divided into a number of local operating regions, fault isolation can be performed by detecting changes in each of these local regions. An additional advantage is that the sensitivity of the FDI scheme can be increased, as fewer information is now required in the fault classification process.

A popular benchmark example in FDI is the three-tank control system, as it contains actuator, sensor, and process faults (Join *et al*., 2005; Zhang *et al*., 2002). It is therefore used here to demonstrate the performance of the proposed FDI scheme. The paper is organised as follows. The description and possible faults of a three-tank system is given in Section 2, followed by the derivation of the proposed FDI scheme in Section 3. The local approaches are described in Section 4, and the FDI of the three-tank control system is presented in Section 5.

2. THREE-TANK SYSTEM

2.1 System Description

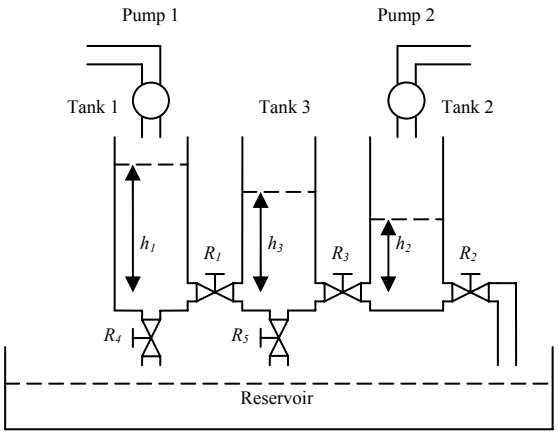

Fig. 1. Schematic diagram of three-tank system

The three-tank control system consists of three tanks connected in series as shown in Fig. 1. Water is pumped respectively into tanks 1 and 2 by pumps 1 and 2 from the

reservoir. Tank 1 is connected to tank 3 through an interconnecting pipe with valve R_1, and tank 3 is connected to tank 2 through an interconnecting pipe with valve R_3. After flowing through an outlet pipe with valve R_2, water is returned to the reservoir. Leakage pipes with valves R_4 and R_5 are installed respectively at the bottom of tanks 1 and 3. The valves that regulate the flow of water through the pipes are fully opened by 5 turns.

The FDI scheme is implemented on a personal computer with an A/D and D/A cards. The water flow from the positive displacement pump is controlled by the voltage applied to the motors, v_1 and v_2, through power supply units with unity gain. The voltage-flow relationship of the pump is approximately linear over the operating range between 3 and 10V. The water level of tanks 1, 2, and 3 are denoted by h_1, h_2, and h_3, and are obtained by pressure transducers fitted at the base of the tanks. The output of the transducer is between 0 and 5V, corresponding to a water level between 0 and 1m. The openings of the valves for the experiment are: 1 and 1/2 turns for R_1, 2 and 1/2 turns for R_2, 1 and 1/2 turns for R_3, while both R_4 and R_5 are closed.

The pumps are controlled by discrete PI controllers at a sampling interval of 1s:

$$v_i(k) = \left[K_P + \frac{K_I}{(1 - z^{-1})} \right] \left[h_{sp,i}(k) - h_i(k) \right], \; i = 1,2 \qquad (1)$$

where z^{-1} is the backward shift operator, K_P and K_I are respectively the proportional and integral gains of the controller, $h_{sp,1}$ is the set-point of h_1, and $h_{sp,2}$ is the set-point of h_2. The parameters of the PI controllers are chosen to be: $K_P = 30$ and $K_I = 1$. An example of the controllable inputs v_1 and v_2, and the measurable outputs h_1, h_2, and h_3 are shown in Fig. 2. The water levels in tanks 1 and 2 with PI control take about 50s to settle after a step change in the set-points. A cycle of the operation consists of an up step, followed by a down step. As the main concern is on FDI rather than closed-loop control, there is no attempt to further improve the performance of the PI controllers.

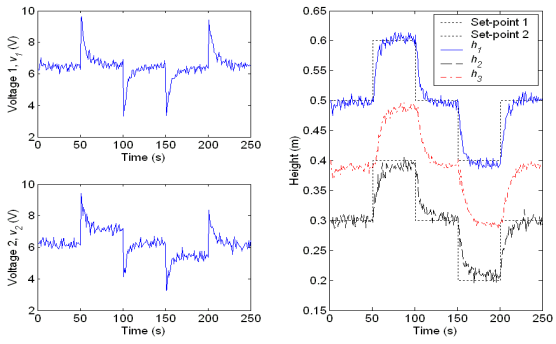

Fig. 2 Inputs and outputs of three-tank control system

2.2 *Faults of System*

There are a total of 22 fault scenarios in the system, as summarised in Table 1.

Table 1. Faulty conditions of three-tank system

Fault	Type	Change
0	Fault-free	No
1	Clogging in pipe 1	R_1 – 1/4 turn
2	Clogging in pipe 1	R_1 – 1/2 turn
3	Clogging in pipe 2	R_2 – 1/4 turn
4	Clogging in pipe 2	R_2 – 1/2 turn
5	Clogging in pipe 3	R_3 – 1/4 turn
6	Clogging in pipe 3	R_3 – 1/2 turn
7	Leakage in tank 1	R_4 + 1/4 turn
8	Leakage in tank 1	R_4 + 1/2 turn
9	Leakage in tank 2	R_2 + 1/4 turn
10	Leakage in tank 2	R_2 + 1/2 turn
11	Leakage in tank 3	R_5 + 1/4 turn
12	Leakage in tank 3	R_5 + 1/2 turn
13	Actuator 1 fault	v_1 – 5%
14	Actuator 1 fault	v_1 – 10%
15	Actuator 2 fault	v_2 – 5%
16	Actuator 2 fault	v_2 – 10%
17	Sensor 1 fault	h_1 – 5%
18	Sensor 1 fault	h_1 – 10%
19	Sensor 2 fault	h_2 – 5%
20	Sensor 2 fault	h_2 – 10%
21	Sensor 3 fault	h_3 – 5%
22	Sensor 3 fault	h_3 – 10%

As the fault-free case is also included in the fault database (FDB) for ease of implementing the proposed FDI scheme, the total number of entries in the FDB is therefore 23. All faults occur abruptly at 25s in the experiments. When it occurs, the PI controllers performs a passive fault tolerant operation by adjusting the control to compensate for the fault, and the regulated levels of tanks 1 and 2 in general is able to return to the desired set-points after a short period of time. Consider fault 1 as an example. When it occurs, the interconnecting pipe between tanks 1 and 3 is clogged, leading to a sudden rise in the water height in tank 1, which in turn leads to a fall in the water heights in tanks 3 and 2. Consequently, the voltage of pump 1 is decreased, while that of pump 2 is increased to maintain the water levels in both tanks close to the set-points, as shown in Fig. 3.

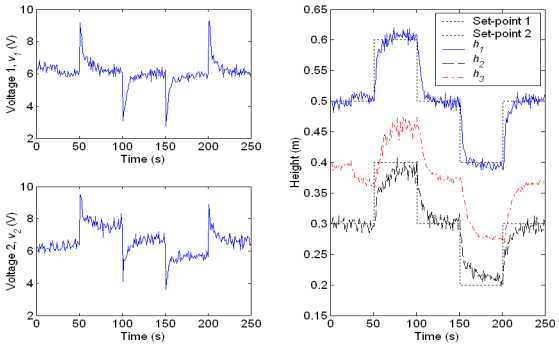

Fig. 3 Inputs and outputs of three-tank system with fault 1

3. FDI BASED ON NEUROFUZZY NETWORKS

There are four steps in the proposed FDI technique as shown in Fig. 4. In the first step, a model, NFN_{sys}, for the system is constructed, and the model, NFN_{res}, that approximates the residual generated by NFN_{sys} is obtained in the second step. In the third step, a FDB is built based on the fuzzy rules extracted from the NFN_{res} for all possible faults and the fault-free case. In the fourth step, faults are diagnosed online by comparing the current fuzzy rules with those in the FDB using a classifier. More details of these steps are presented below.

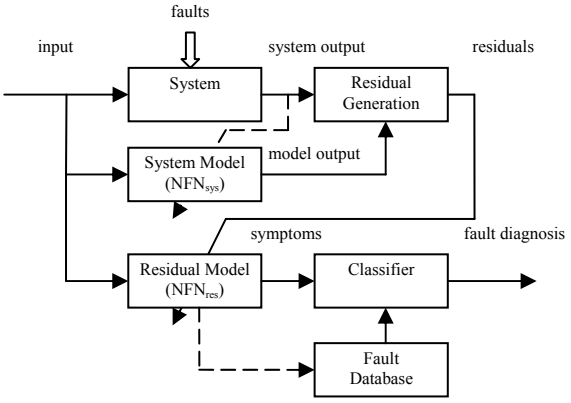

Fig. 4. Schematic diagram of FDI scheme

3.1 Modelling of System

Similar to other model-based methods for fault diagnosis of dynamic systems, a system model is first built. Since nonlinear autoregressive with exogenous input (NARX) model is able to represent a large class of discrete nonlinear dynamic systems and can be readily implemented by the NFN, it is used here to model the nonlinear system. The NARX model is given by (Chen and Billings, 1989):

$$y(k) = f[y(k-1),..., y(k-n_y), u(k-1), ..., u(k-n_u)] + \xi(k) \quad (2)$$

where $f(.)$ is a smooth nonlinear function, $y(k)$ is the output of the system with order n_y, $u(k)$ is the input of the system with order n_u, and $\xi(k)$ is a white noise.

The NFN can be considered as a three-layer feedforward network, as shown in Fig. 5.

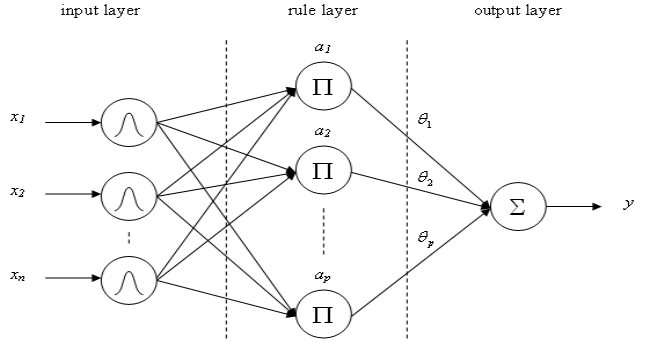

Fig. 5 Schematic diagram of neurofuzzy network

The inputs of the NFN are fuzzified using BSBFs in the input layer. Fuzzy inference is performed in the rule layer by *T*-norm (AND) operations on the antecedent rules using the algebraic products and the level of fulfilments of the antecedent rules:

$$a_i(x) = \prod_{j=1}^{n} \mu_{\{i\}}(x_j),\ i = 1,..., p \quad (3)$$

where Π is the *T*-norm operator, μ is the BSBFs, and p is the number of antecedent fuzzy rules or the number of weights in the network, which is given by:

$$p = \prod_{j=1}^{n} m_i^j \quad (4)$$

with m_i^j as the number of fuzzy membership functions in the jth input. In the output layer, fuzzy outputs are inferred by aggregating the outputs from the fuzzy rules using a defuzzifier. The common one is the centre of gravity (COG) method. The network output is then given by (Hong and Harris, 2001):

$$y(x) = \sum_{i=1}^{p} a_i(x)\theta_i = a^T(x)\theta \quad (5)$$

where a is the transformed input vector and θ is the weight vector.

For the NFN that models the nonlinear system (2), denoted by NFN_{sys}, the network input can be readily chosen as: $x(k) = [y(k–1), \ldots, y(k–n_y), u(k–1), \ldots, u(k–n_u)]^T$. To set up the network, it is necessary to select first n_y and n_u in $x(k)$, and then the range, the number, and the order of BSBFs for each element in $x(k)$. The centre and knot vectors for the variables are then chosen next. The number of BSBFs is selected based on the nonlinearity of the system. More BSBFs are used, if the nonlinearity of the system is high, and vice versa. Since the overlapping regions of the BSBFs increase as the order of the basis functions increases, the approximation of the nonlinear system is likely to be smoother and better. However, the computation burden will increase. Since the network output is a linear function of the transformed network inputs and the network weights, the well known linear least squares method can be used to obtain $\hat{\theta}$, and for online estimation, the recursive least squares (RLS) method can be used (Wellstead and Zarrop, 1991).

3.2 Modelling of Residual

The residual, r, is computed as follows:

$$r = y - \hat{y} \quad (6)$$

where y is the system output and $\hat{y}$, the output from NFN_{sys}, both of which are trained online for FDI using the RLS method. Since the residual contains information of faults in the system, the knowledge extracted from the residual model can therefore be used to diagnose these faults.

Extraction of Knowledge from Neurofuzzy Network. By reversing the process that computes the weights of the NFN_{res} from the consequents of the fuzzy rules, fuzzy rules can be

extracted from the weights of the NFN_{res} under certain conditions. From the extracted fuzzy rules, a linguistic description of the residual can be obtained. For example, consider the fuzzy rule base (FRB) of a SISO NFN:

$$R_{i,j}\text{: IF } x \text{ is } A_i\text{, THEN } y \text{ is } B_j \qquad (c_{i,j});$$

$$\text{for } i = 1,\ldots, m_i \text{ and } j = 1,\ldots, m_o$$

where A_i, B_j denote respectively the fuzzy sets in the input space with m_i partitions and the output space with m_o partitions, $c_{i,j} \in [0,1]$ is the confidence level (CL) of rule $R_{i,j}$ being true. Let the fuzzy sets in the output space be at the centres: $\{ \beta^1, \beta^2, \ldots, \beta^{m_o} \}$ and the membership functions over the output form a partition of unity. If the COG method is used in the defuzzification process, the network weights can be obtained as follows:

$$\theta_i = \sum_{j=1}^{m_o} c_{i,j}\beta_j, \ i = 1,..., p \tag{7}$$

From (7), the CLs of the fuzzy rules for a given set of weights $\theta = \{\theta_1, \ldots, \theta_{mi}\}$ can be computed by reversing this process, as given below (Brown and Harris, 1994):

$$c_{i,j} = \mu_j(\theta_i), \ i = 1,..., p; \ j = 1,..., m_o \tag{8}$$

where μ is the BSBFs and the total number of rules in the FRB is given by:

$$n_r = pm_o \tag{9}$$

3.3 Construction of Fault Database

To facilitate the detection and isolation of faults in the system, the fault patterns are transformed first into features. As the FRB extracted from the NFN_{res} is a concise description of the fault, they can be utilised as a feature to classify faults in the system. To facility the FDI process, the linguistic FRB is encoded into symptom vectors (SVs) containing the CLs of the fuzzy rules. For the FRB extracted from the NFN_{res}, the sum of the CLs of all the rules in the same antecedent is unity (Mok and Chan, 2008). A FDB containing SVs for all possible faults in the system is constructed and is used for fault classification. As the dimension of the SVs extracted from the NFN_{res} is identical to the number of rules in the FRB given by (9), the FDB containing n_f faults is formed by incorporating these SVs, as follows:

$$D = [s_0 \quad s_1 \quad \cdots \quad s_{n_f}]^T \tag{10}$$

where $\{s_i, i = 1, \ldots, n_f\}$ are SVs for all possible faults, and s_0, the fault-free case. As the fault-free case is also included in the FDB for ease of implementation of the proposed scheme, then from (9) and (10), D is a matrix with a dimension of $(n_f + 1) \times n_r$.

3.4 Classification of Fault

After the FDB is constructed, faults are classified by comparing SVs updated online with those in the FDB using classifiers. As the weights of NFN_{res} are updated recursively by the RLS method, fuzzy rules are extracted from the weights of the NFN_{res} and then encoded as a SV. By comparing this SV with those in the FDB using some classifiers, faults are detected and isolated.

Nearest neighbour classifier (NNC) is a geometric classifier derived based on the minimal Euclidean distance, and is used to classify faults in this paper. Let the Euclidean distances between the SV updated online, $s(k)$, and each of the SV in the FDB are given by:

$$d_i(k) = \|s(k) - s_i\|, \ i = 0,..., n_f \tag{11}$$

The NNC classifies $s(k)$ to be fault j if $d_j(k)$ is a minimum amongst all possible faults in the FDB, i.e.,

$$d_j(k) = \min\|s(k) - s_i\|, \ i = 0,..., n_f \tag{12}$$

In this case, faults are isolated on the basis that the smaller is $d_j(k)$, the closer is the match. If several faults have the same minimum distance, all these faults are identified as possible faults and further investigation is required to resolve which one of these faults is the actual fault. If the winner is other than the fault-free case, a fault is detected and isolated later when the weights have approached some constants.

4. FDI BASED ON NEUROFUZZY NETWORKS WITH LOCAL APPROACHES

4.1 Operating Region of Neurofuzzy Networks

Since the input space of an NFN is partitioned by the BSBFs into local spaces, only a small number of univariate or multivariate BSBFs are contributing to the output of the network. Information is stored and learned locally across ρ basis functions in each input. Because of this local property, the global operating region can be considered as a combination of a number of local operating regions, and the number of local regions is given by:

$$n_l = \prod_{i=1}^{n} (m_i - \rho + 1) \tag{13}$$

where m_i is the number of BSBFs in each input, ρ is the order of BSBFs, and n is the number of inputs.

Consider an NFN with two inputs. Three second order BSBFs: {S, M, L} are used for each input. The centres of the BSBFs of input x_1 are: $\{\alpha_1, \alpha_2, \alpha_3\}$, and those of x_2 are: $\{\beta_1, \beta_2, \beta_3\}$. The input space of the NFN is shown in Fig. 6.

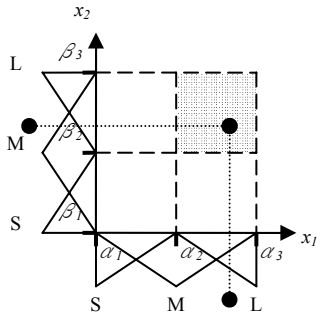

Fig. 6. Local operating regions

The global operating region is divided into 4 local operating regions, as shown in Fig. 6. For each combination of x_1 and x_2, only one local region is activated each time. Therefore, it follows that only that particular local space of the NFN is sufficient to represent the state of the system at that time.

4.2 Online Training using Modified Recursive Least Squares

Chan *et al*. (2000) proposed to simplify the RLS method by updating only the elements of the covariance matrix $P(k)$ associated with the non-zero elements of $a(k)$, resulting in a modified RLS (MRLS) method, as shown in Fig. 7.

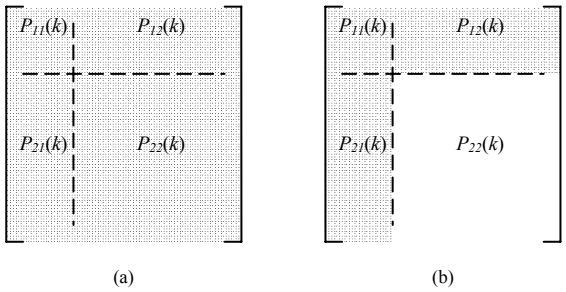

Fig. 7. Update of covariance matrix: (a) RLS method; (b) MRLS method

Since $P_{22}(k)$ is not updated in this MRLS algorithm, and it is a symmetrical matrix, the number of elements computed in the MRLS method is given by:

$$n_{pl} = p'\left(p - \frac{(p'-1)}{2}\right) \tag{14}$$

where p' is the number of non-zero elements in $a(k)$. In contrast, the number of elements of $P(k)$ computed in the RLS method is given by:

$$n_p = \frac{p(p+1)}{2} \tag{15}$$

From (14) and (15), the computing time in the MRLS method is a linear instead of a quadratic function of the number of weights in the network, giving a considerable saving in computation, especially when p is much greater than p'.

4.3 Fault Classification using Local Fault Databases

Since the global operating region of the NFN can be considered as a combination of a number of local operating regions, local SVs can be used to form local FDBs for FDI purpose. In this approach, the FDI results are more sensitive, as only local sections of the SV are used. The number of local FDBs is equal to the number of local operating regions of the NFN, and the number of fuzzy rules in a local SV is:

$$n_{rl} = \rho^n m_o \tag{16}$$

where ρ is the order of BSBFs and is also the number of univariate basis functions activated in each input, n is the number of inputs, and m_o is the number of BSBFs in the output space. At each time interval, only the local elements in the global SV are updated and as only these elements are compared with those in the global FDB, the sensitivity of the global scheme can be computed as follows:

$$\gamma = \frac{n_{rl}}{n_r} \tag{17}$$

If all the elements in the local SV are updated instead and compared with those in the corresponding local FDB, the sensitivity of the local scheme is given by:

$$\gamma_l = \frac{n_{rl}}{n_{rl}} = 1 \tag{18}$$

Therefore, the sensitivity of the local approach is higher than that of the global one as $n_{rl} \leq n_r$.

5. FDI OF THREE-TANK SYSTEM

5.1 Modelling of System

The three-tank system is a third order MIMO system with two inputs and two outputs. It is modelled by two MISO systems. For system model 1, denoted by $\text{NFN}_{\text{sys},1}$, five inputs: $v_1(k-1)$, $v_2(k-1)$, $h_1(k-1)$, $h_2(k-1)$, and $h_3(k-1)$ are used to model h_1(k). Similarly, an input vector $x(k) = [v_1(k-1)\ \ v_2(k-1)\ \ h_1(k-1)\ \ h_2(k-1)\ \ h_3(k-1)]^T$ is used to model $h_2(k)$ for the system model 2, denoted by $\text{NFN}_{\text{sys},2}$. The ranges of v_1, v_2, h_1, h_2, and h_3 are from 2 to 10V, 2 to 10V, 0.35 to 0.65m, 0.15 to 0.45m, and 0.20 to 0.60m, respectively. The number of membership functions, m_i, for each input is 3. The order of the basis functions, ρ, is chosen to be 2. The linguistic labels in the input spaces of the system models are: {Small, Medium, Large}, and the total number of weights in each NFN_{sys} is 243.

5.2 Modelling of Residual

Two residual models, $\text{NFN}_{\text{res},1}$ and $\text{NFN}_{\text{res},2}$ are constructed in a similar way as the system models. The inputs of the residual models are the same as those of system models and the output of the residual models are respectively the predicted residual of the height in tank 1: $y(k) = \hat{r}_1(k)$ and the predicted residual of the height in tank 2: $y(k) = \hat{r}_2(k)$. The range of r_1 and r_2 is from -0.03 to 0.03m. Three second order BSBFs are also used in the output of the residual models and the linguistic labels used in the output space of NFN_{res} are: {Negative, Zero, Positive}. The number of network weights and fuzzy rules in the NFN_{res} are 243 and 729 respectively. In the training of the NFNs, the number of elements requires computing in the covariance matrix in the MRLS method is 7280 given in (14) while that in the RLS method is 29646 given in (15), and the local approach saves 75% in computing time.

5.3 Construction of Fault Database

Each of the 22 faults, as given in Table 1, occurs abruptly at 25s in the experiments. For each scenario, a corresponding SV is obtained from the $\text{NFN}_{\text{res},1}$ and $\text{NFN}_{\text{res},2}$ after their weights have converged. The residual behaviours of the

three-tank system can be described numerically by the SVs or linguistically by the corresponding fuzzy rules. The length of each SV obtained from a residual model is 729, which is the same as the number of fuzzy rules in each FRB extracted from the NFN_{res}, as given by (9). A final SV is a combination of SVs from the two residual models that has a length of 1458. The SV for the fault-free case and the 22 SVs for the faulty cases are finally stored in a FDB with a dimension of 23×1458. The operating region of the NFN_{res} can be divided into 32 local regions, as given by (16). Similarly, the FDB can also be divided into 32 local FDBs same as the local operating regions. In practice, there is no need to create and store all local FDBs. It is only necessary to store the symptom index associated with the local operating regions in local fault classification.

5.4 FDI Results

After the FDB is constructed, the FDI method is applied to detect and isolate the 22 faults given in Table 1. Each fault is introduced in the system and is isolated by the NNC with FDB. Both global and local approaches select the fault-free case before faults are introduced and identify the correct fault after faults are added into the system in all cases. In addition to the higher computational efficiency and sensitivity, the local approach outperforms the global one in terms of the time to detect and to isolate the faults, as given Figs.8 and 9.

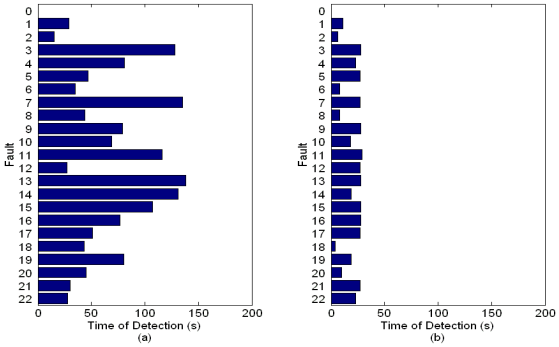

Fig. 8. Time of detection of faults: (a) global; (b) local

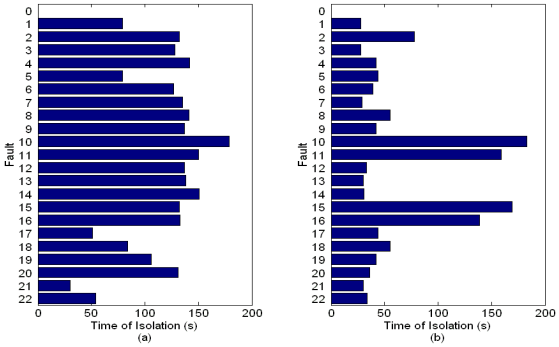

Fig. 9. Time of isolation of faults: (a) global; (b) local

6. CONCLUSION

In this paper, a FDI scheme is developed for a three-tank control system. The scheme is modified using the local approaches to reduce the computing time in training the NFNs and to increase the sensitivity in the classification of the faults. The MRLS method that utilises the local change property of NFNs is used for training the NFNs, and by dividing the NFN into a number of local operating regions, local FDBs are constructed from the local SVs for FDI. These measures can alleviate the problem of curse of dimensionality in NFNs, especially for monitoring complex systems that has a large number of inputs. In addition to the higher computational efficiency and sensitivity, the local approach outperforms the global one with respect to the time to detect and to isolate faults in the system.

REFERENCES

Brown, M. and C.J. Harris (1994). *Neurofuzzy Adaptive Modelling and Control*. New York: Prentice Hall.

Chan, C.W., K.C. Cheung and W.K. Yeung (2000). A computation-efficient on-line training algorithm for neurofuzzy networks. *International Journal of Systems Science*, **31(3)**, 297–306.

Chen, S. and S.A. Billings (1989). Representation of non-linear systems: the NARMAX model. *International Journal of Control*, **49(3)**, 1013–1032.

Hong, X. and C.J. Harris (2001). Neurofuzzy design and model construction of nonlinear dynamical processes from data. *IEE Proceedings - Control Theory and Applications*, **148(6)**, 530–538.

Isermann, R. (2005). Model-based fault-detection and diagnosis - status and applications. *Annual Reviews in Control*, **29(1)**, 71–85.

Join, C., J.-C. Ponsart, D. Sauter and D. Theilliol (2005). Nonlinear filter design for fault diagnosis: application to the three-tank system. *IEE Proceedings - Control Theory and Applications*, **152(1)**, 55–64.

Mok, H.T. and C.W. Chan (2008). Online fault detection and isolation of nonlinear systems based on neurofuzzy networks. *Engineering Applications of Artificial Intelligence*, **21(2)**, 171–181.

Patton, R.J., Chen, J. and Benkhedda, H. (2000). A study on neuro-fuzzy systems for fault diagnosis. *International Journal of Systems Science*, **31(11)**, 1441–1448.

Wellstead, P.E. and M.B. Zarrop (1991). *Self-tuning Systems: Control and Signal Processing*. Chichester: Wiley.

Zhang, X., M.M. Polycarpou and T. Parisini (2002). A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems. *IEEE Transactions on Automatic Control*, **47(4)**, 576–593.

Zio, E. and Gola, G. (2006). Neuro-fuzzy pattern classification for fault diagnosis in nuclear components. *Annals of Nuclear Energy*, **33(5)**, 415–426.