Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Neurofuzzy Modelling and Pattern Matching for Online Fault Detection and Isolation of Nonlinear DC Motors

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: An online fault detection and isolation scheme for nonlinear systems based on neurofuzzy modelling and pattern matching is developed in this paper. The system is first modelled offline by a neurofuzzy network using data obtained under normal operating conditions. Another neurofuzzy network is then used to model the residual, which is the difference between the output of the system and that from the neurofuzzy network. For online fault monitoring, it is necessary to construct first a fault database that contains fuzzy rules for all possible faults in the system. Recursive least squares algorithm is used to train the network online, from which the IF-THEN rules are extracted. Faults are isolated online by comparing these fuzzy rules with those in the fault database using a nearest neighbour classifier. A simulation example involving a nonlinear DC motor control system is used to demonstrate the implementation and performance of the proposed FDI scheme.

1. INTRODUCTION

Fault diagnosis of electrical and mechanical systems is very important in industrial applications, as it can improve the system reliability. Model-based method is commonly used for fault detection and isolation (FDI) of systems since no additional hardware is needed to implement the algorithm (Isermann, 2005). Faults of a system are determined by comparing the available system measurements with the information represented by the system model (Chen and Patton, 1999). The model-based FDI method works well if complete and accurate mathematical models of the systems being monitored are available (Korbicz *et al*., 2004). However, it is difficult, and some times impossible to obtain accurate models in practice, as they are complex with perhaps too highly nonlinear dynamics (Frank *et al*., 2000).

Fuzzy systems allow symbolic generalisation of numerical data by fuzzy IF-THEN linguistic rules, which can be more readily understood by operators (de Miguel and Blázquez, 2005). The time consuming tuning process can be tackled by the neural networks as the neural networks can learn the model parameters from input-output data obtained from system operations (Nelles, 2001). Recently, FDI techniques based on neurofuzzy networks (NFNs) are attracting immense interest (Patton *et al*., 2000). However, the focus of existing works is mainly on the learning ability of NFNs in fault classification and very few of them utilise the linguistic ability of NFNs to develop transparent fault diagnosis techniques. On the other hand, many research works are devoted to fault diagnosis of DC motors as motors are fundamental components in engineering systems, (Chan *et al*., 2006).

In this paper, neurofuzzy modelling and pattern matching are used for online FDI of nonlinear DC motor control systems. NFNs are being used as they have the generalisation and learning abilities of neural networks, in addition to the ability to incorporate fuzzy rules. The latter ability allows expert knowledge in linguistic form to be included and extracted from the NFN (Brown and Harris, 1994). From this ability, a linguistic description of the faults in terms of fuzzy IF-THEN rules can be extracted from the NFN for fault isolation (Mok and Chan, 2005). The FDI technique proposed here consists of four steps. The first step is to model the system using an NFN. The second step is to model the residual behaviours both under normal and faulty operating conditions using another NFN. The third step is to extract the behaviours in terms of IF-THEN rules from the residual NFN and store them in a fault database. The final step is to isolate the fault based on the current system behaviours and the patterns previously stored in the fault database by classifiers (Friedman and Kandel, 1999).

2. PRELIMINARIES

2.1 Structure of B-spline neural network

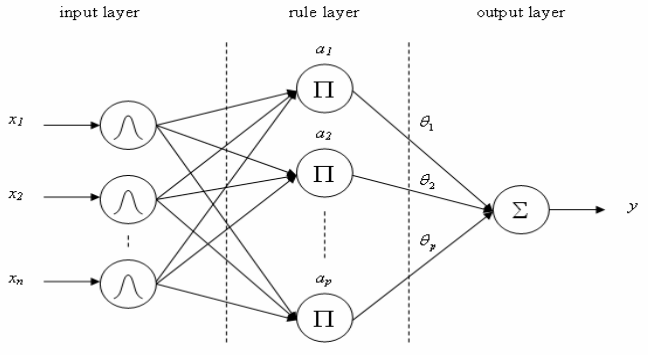

Fig. 1. Neurofuzzy network

A B-spline neural network, also known as an NFN, can be considered as a three-layer feedforward neural network as shown in Fig. 1. In the input layer, inputs are fuzzified using B-spline basis functions (BSBFs) given by the following recurrent formulae (de Boor, 1972):

$$\mu_k^{\rho}(x) = \left(\frac{x-\lambda_k}{\lambda_{k+\rho-1}-\lambda_k}\right)\mu_k^{\rho-1}(x) + \left(\frac{\lambda_{k+\rho}-x}{\lambda_{k+\rho}-\lambda_{k+1}}\right)\mu_{k+1}^{\rho-1}(x) \tag{3.1}$$

$$\mu_k^{1}(x) = \begin{cases} 1, & \text{if } x \in [\lambda_k, \lambda_{k+1}) \\ 0, & \text{otherwise} \end{cases}$$

where $\mu_k^{\rho}(x)$ is the kth membership function with order ρ (degree $\rho - 1$) and λ is the knot vector of the B-spline basis defining on the universe of discourse.

Fuzzy inference is performed in the rule layer using the T-norm (AND) operations for the antecedent rules. The algebraic products are used with the level of fulfilments of the antecedent rules represented by the multivariate BSBFs:

$$a_i(x) = \prod_{j=1}^{n} \mu_{x_j\,k}^{\ \rho}(x), \quad i = 1,..., p \tag{3.2}$$

where Π is used as the T-norm operator in the network and p is the number of antecedent fuzzy rules or the number of weights in the network given by:

$$p = \prod_{j=1}^{n} m_i^j \tag{3.3}$$

where m_i^j is the number of fuzzy membership functions used in the jth input.

The transformed inputs (3.2) can be written as:

$$a(x) = [a_1(x), a_2(x), ..., a_p(x)]^T \tag{3.4}$$

and the consequent rules expressed in terms of fuzzy singletons (constants) are stored in the network as the weights as follows:

$$\theta(x) = [\theta_1(x), \theta_2(x), ..., \theta_p(x)]^T \tag{3.5}$$

In the output layer, fuzzy outputs are inferred from each fuzzy rule and are aggregated to give a final numerical output by a defuzzifier in the output layer of the network. The defuzzification process is performed using the centre of gravity (COG) method, giving the network output as follows (Brown and Harris, 1994):

$$y(x) = \frac{\sum_{i=1}^{p} a_i(x)\theta_i}{\sum_{i=1}^{p} a_i(x)} = \sum_{i=1}^{p} a_i(x)\theta_i = a^T(x)\theta \tag{3.6}$$

2.2 Online Training of NFN

For simplicity, the argument x in (3.6) will be omitted in the following analysis. Let $\hat{\theta}$ be the estimate of the weights of the NFN. From (3.6), the estimate of the output of the NFN, $\hat{y}$, using the estimated weight, $\hat{\theta}$, at the kth training data is:

$$\hat{y}(k) = a^T(k)\hat{\theta}(k-1) \tag{3.7}$$

The training of NFN involves finding a set of weights $\hat{\theta}$ that minimises the cost function:

$$J = \sum_{k=1}^{N} \left(y(k) - \hat{y}(k)\right)^2 \tag{3.8}$$

where N is the number of training data, $y(k)$ is the kth output in the training set. Since the network output is a linear function of the transformed network input and the network weights. The well known recursive least squares (RLS) method can be used to online estimate the weights, as given below (Wellstead and Zarrop, 1991):

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \frac{P(k-1)a(k)[y(k) - a^T(k)\hat{\theta}(k-1)]}{1 + a^T(k)P(k-1)a(k)} \tag{3.9}$$

and the covariance matrix P(k) is updated by:

$$P(k) = P(k-1) - \frac{P(k-1)a(k)a^T(k)P(k-1)}{1 + a^T(k)P(k-1)a(k)} \tag{3.10}$$

To start the RLS method, $\hat{\theta}$ is often initialised to 0, and P to αI, where α is a positive constant, and I is the identity matrix. The initial setting of the covariance matrix reflects the uncertainty in the network weights being estimated. When the initial weights are known to be close to the true values, a small α is enough, typically 1 to 10. In contrast, a large α, 100 to 1000, is required when there is no prior knowledge of the weights.

2.3 Knowledge Extraction from Trained NFN

As the weights of the NFN are obtained from the consequences of the fuzzy rules, fuzzy rules can be extracted from the weights of the NFN under certain conditions to provide a linguistic description of the system that it generalised. Consider the following fuzzy rule for a SISO NFN:

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

for $i = 1,\ldots, m_x$ and $j = 1,\ldots, m_y$

where A_i, B_j denote respectively the fuzzy membership functions in the input space with m_x partitions and the output space with m_y partitions, $c_{i,j}$ is the rule confidence of $R_{i,j}$ being true. The fuzzy sets used in the output space are evenly distributed with centres at $\{\beta_1, \beta_2, \ldots, \beta_{m_y}\}$, and that the membership functions over the output space forms a partition of unity. If the COG method is used in the defuzzification process, the network weight is given by:

$$\theta_i = \sum_{j=1}^{m_y} c_{i,j}\beta_j \tag{3.11}$$

Reversing this process, the rule confidences of the fuzzy rules for a given set of optimal weights θ of the network can be computed as follows (Brown and Harris, 1994):

$$c_{i,j} = \mu_{B_j}(\theta_i) \tag{3.12}$$

The total number of rules extracted is given by:

$$n_r = pm_o \tag{3.13}$$

where m_o is the number of fuzzy membership functions used in the output space.

3. FDI USING NFN AND CLASSIFIER

3.1 Design of Neurofuzzy Residual-model-based Method

Model-based method is commonly used for FDI of engineering systems (Isermann, 2005). When the actual system output is different from the estimated model output, a fault is detected. Under different faulty conditions, the system may behave differently. Therefore, faults can be isolated based on their distinctive characteristics. However, complete and accurate mathematical models are not always available. In many occasions, empirical models are needed to construct from the operational data. However, these models may not be readily understood by operators due to their black-box nature. On the other hand, a transparent model with learning ability can be constructed for FDI purpose using the NFN.

The proposed FDI technique consists of four main steps, as illustrated in Fig. 2. The first step is to build a model for the system, which is denoted by NFN_{sys}. The second step is to build a residual model, NFN_{res}, that models the residual generated between the system output and the model output. The third step is to construct a fault database containing information of symptoms under normal and faulty operations from the NFN_{res}. The final step is to detect and isolate the happened fault online by comparing the current symptoms with those in the fault database.

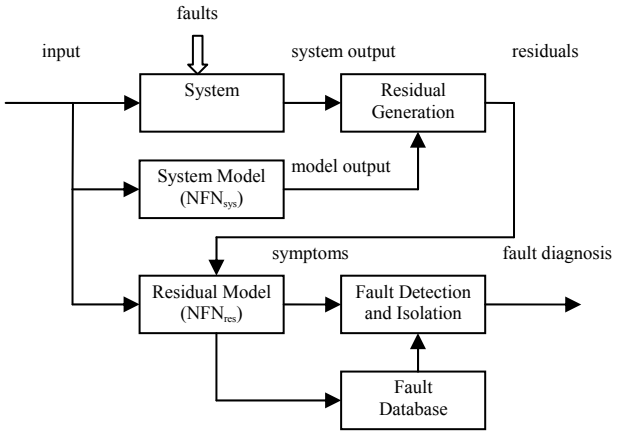

Fig. 2. Residual-model-based method

3.2 Construction of Fault Database with Fuzzy Rules

In order to isolate a fault from the system, some knowledge of the system behaviours under different fault conditions need to be available. When a fault is introduced into the system, the input-output relationships of the system will change and these changes are captured by the NFN_{res} during online training. The knowledge extracted from the NFN_{res} in terms of IF-THEN rules is stored in a fault database.

To construct a fault database for a system, some known fault scenarios are first simulated in the system. Let n_f be the number of possible faults in the system. As the fault-free case is also included in the fault database, the number of entries in the fault database is: $n_f + 1$. For ease of comparison, a symptom vector, $s_i \in \Re^{n_r}$, is formed from the fuzzy rule base extracted from the NFN_{res} for the ith fault. The component of the symptom vector is in fact the rule confidence of each fuzzy rule being true in the rule base, as given by:

$$s = [c_{1,1}, ..., c_{1,m_y}, c_{2,1}, ..., c_{2,m_y}, ..., c_{p,1}, ..., c_{p,m_y}]^T \tag{3.14}$$

The full fault database in matrix form, $D \in \Re^{(n_f+1)\times n_r}$, is formed by incorporating all the symptom vectors, as given by:

$$D = [s_0, s_1, ..., s_{n_f}]^T \tag{3.15}$$

where s_0 is the fault-free symptom vector, and s_i (i = 1, …, n_f), for all possible fault cases. The symptom space in terms of fuzzy IF-THEN rules is encoded into the symptom vectors which are stored in the database to be used later for fault classification.

3.3 Classification of Fault by Nearest Neighbour Classifier

Pattern classification by distance functions is one of the earliest concepts in pattern recognition (Friedman and Kandel, 1999). The goal of pattern classification is to associate a class with the given input pattern. For classification, patterns are usually first reduced to features, where the features of different classes should be different and separated from each other.

After a fault database has been constructed, faults can then be readily isolated online from input-output data of the operating system. Fault or fault-free cases are diagnosed by comparing the current symptom vector obtained online through the NFN_{res} with those stored in the fault database, as shown in Fig. 2. The fault classification can be achieved by a simple, yet robust nearest neighbour classifier (NNC), which is a geometric classifier based on the minimal Euclidean distance (Friedman and Kandel, 1999).

When the FDI scheme starts, the weights of NFN_{res} are updated recursively by the RLS method in each sampling interval, and fuzzy rules are extracted from the updated weights for describing the current residual dynamics in a linguistic manner. These IF-THEN rules are then encoded into a symptom vector based on the rule confidences, which is then used for classification. For a given symptom vector, it is matched with the symptom patterns stored in the fault database according to the minimal distance.

Let $f = \{f_0, f_1, \ldots, f_{n_f}\} \in \Re^{n_f+1}$ be the fault classes. The NNC isolates the fault, if any, by matching the faults in the fault database, D. The Euclidean distances between the symptom vector of the current state, $s(k)$, and that of the ith fault stored in the database are given by:

$$d_i(k) = \|s(k) - s_i\| = \left((s(k) - s_i)^T (s(k) - s_i)\right)^{\frac{1}{2}},\ i = 0, ..., n_f \tag{3.16}$$

The NNC classifies $s(k)$ as f_j such that $d_j(k)$ is a minimum, that is:

$$d_j(k) = \min \| s(k) - s_i \|, \quad i = 0,..., n_f \tag{3.17}$$

The shorter is the distance, the higher is the match, and hence the winner. If the minimum is achieved by several j's, all the f_j are selected as the possible faults and further investigation is needed to handle the case. If f_0 is not selected as the winner, then a fault has occurred and is isolated accordingly.

4. DC MOTOR EXAMPLE

4.1 System Equations

Consider a DC motor with a shunt field circuit described by the following set of nonlinear ordinary differential equations (Watanabe *et al.*, 1985):

$$\begin{cases} \dot{i}_f = -\dfrac{R_f}{L_f} i_f + \dfrac{1}{L_f} v \\ \dot{i}_a = -\dfrac{R_a}{L_a} i_a - \dfrac{M}{L_a} i_f \omega + \dfrac{1}{L_a} v \\ \dot{\omega} = -\dfrac{M}{J} i_f i_a - \dfrac{D}{J} \omega \end{cases} \tag{3.18}$$

where i_f is the field current, i_a is the armature current, ω is the rotational speed, and v is the input voltage. The system parameter for simulating the DC motor is given in Table 1.

Table 1. Parameters of DC motor

Symbol	Parameter	Value
R_f	Field resistance	50 Ω
L_f	Field inductance	20 H
R_a	Armature resistance	3.8 Ω
L_a	Armature inductance	0.5 H
D	Viscous resistance	0.042 Nm•s/rad
J	Moment of inertia	0.4 kgm^2
M	Mutual inductance	0.221 H

The system is simulated using Runge-Kutta method with a time step of 0.1s. The angular speed of the motor is controlled by a digital PI controller with a sampling interval of 1s:

$$v(k) = \left[K_P + \frac{K_I}{(1 - z^{-1})} \right] \left[\omega_{sp}(k) - \omega(k) \right] \tag{3.19}$$

where z^{-1} is the backward shift operator, K_P and K_I are respectively the proportional and integral gains of the controller, and ω_{sp} is the set-point of ω. The parameters of the PI controller are selected by trials and errors as: K_P = 1 and K_I = 0.2. Gaussian noises with zero mean and standard deviations of 0.1A and 1rad/s are added to i and ω respectively. An example of the controllable input v and the measurable outputs $i = i_f + i_a$ and ω are shown in Fig. 3. The angular speed of the motor is controlled by the PI controller, which takes about 60s to settle after a step change in the set-point. A cycle of step changes in the set-point is applied to the closed-loop system and the closed-loop output is shown in Fig. 3. As the purpose of the example is to illustrate the performance of the proposed FDI technique, there is no attempt to further improve the performance of the PI controller.

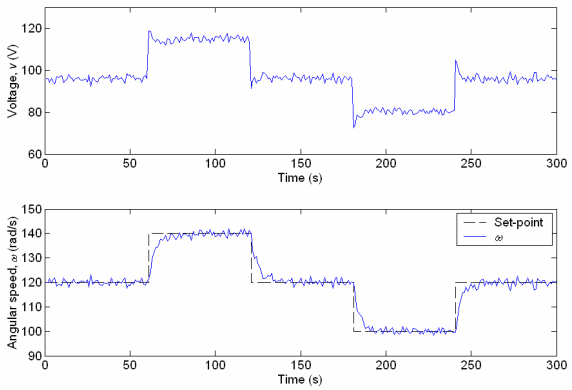

Fig. 3. Input and output of the DC motor

4.2 System Faults

There are four types of fault scenarios considered in the system (Watanabe *et al.*, 1985): (1) a fault in the brush in the rectifier that leads to an increase in the resistance R_a; (2) disconnection of the magnetic circuit of the motor that leads to a decrease in the values of L_a and M; (3) degradation of the insulation of the armature circuit that leads to a decrease in the values of M, R_a, and L_a; and (4) impairment of the shaft supports that leads to an increase in the value of D. Nine scenarios including eight fault cases and the fault-free case are considered here, as summarised in Table 2. It is assumed that the change in these parameters occurred abruptly at 30s in the simulations.

Table 2. Faulty conditions of DC motor

Fault	Parameter	Change
0	——	——
1	R_a	+ 5%
2	R_a	+ 10%
3	L_a, M	+ 5%
4	L_a, M	+ 10%
5	M, L_a, R_a	+ 5%
6	M, L_a, R_a	+ 10%
7	D	+ 5%
8	D	+ 10%

4.3 System and Residual Modellings

In this example, the input of the system model, NFN_{sys}, is: $x(k) = [v(k\text{-}1), i(k\text{-}1), \omega(k\text{-}1)]^T$, while the output of the model is the predicted angular speed of the motor: $y(k) = \hat{\omega}(k)$. The ranges of v, i, and ω are respectively 60 to 130V, 7 to 22A, and 90 to 150rad/s. The linguistic variables for the input of NFN_{sys} are: Small (S), Medium (M), and Large (L), as shown in Fig. 4. The number of membership functions, m_i, for each input is 3, and the order of basis functions, ρ, is 3.

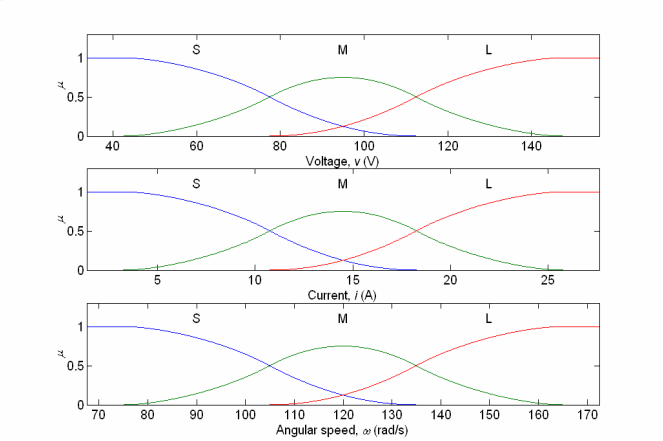

Fig. 4. Input spaces of neurofuzzy model

From (3.3), the total number of weights in the NFN_{sys}, p, is 27. The network weights, θ, are initialised to 0 and the initial covariance matrix is set to $1000I$. The NFN_{sys} is trained using data obtained from 1 to 540s. It is shown in Fig. 5 that $\hat{\omega}(k)$ follows closely the actual motor speed after training for 250s. The recursively updated network weights are also shown in Fig. 5. The learning process of NFN_{sys} is terminated after the network weights are seen to have approached some constant values after several step changes in the set-point, as the modelling error becomes very small.

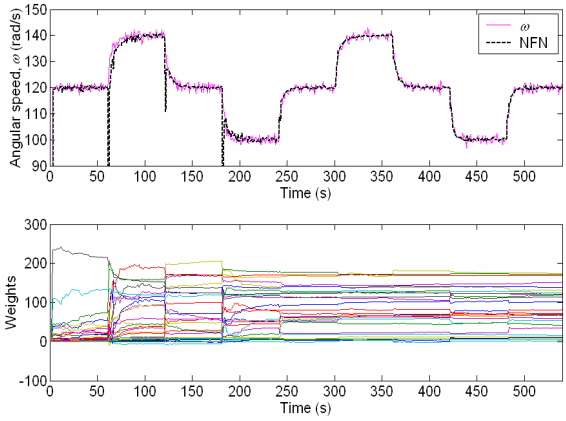

Fig. 5. Training of system model

Similarly, the neurofuzzy residual model, NFN_{res}, is set up with the same inputs of the NFN_{sys}, but the output is now $\hat{r}(k)$ instead of $\hat{\omega}(k)$ in the NFN_{sys}. The range of r is -5 to 5rad/s. The network weights are initialised to 0 and the initial covariance matrix is set to I, as the residual is close to 0. The training of the residual model NFN_{res} for the period from 1 to 540s is shown in Fig. 6.

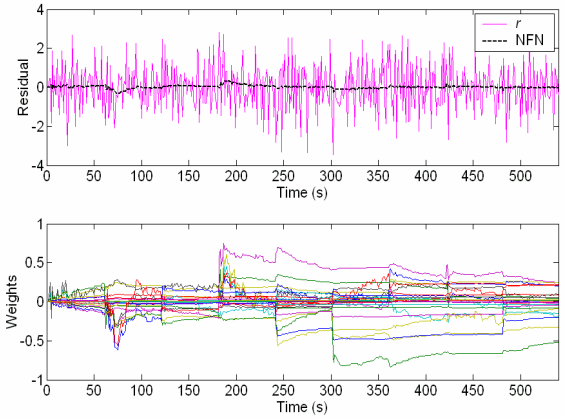

Fig. 6. Training of residual model

4.4 Construction of Fault Database

From (3.13), there are 81 fuzzy rules that can be extracted from NFN_{res}. These IF-THEN rules describe the behaviours of the nonlinear residual and can be helpful to the operator to better understand the operation of the system. Each of the 9 fault conditions given in Table 2 is simulated separately with the change in the parameters occurring abruptly at 30s. In each scenario, a symptom vector is obtained from NFN_{res} after its weights have converged, as described in Section 3.2. The behaviours of the residual can now be described in numerical terms by the symptom vector or linguistically by the corresponding fuzzy rules extracted in the form of input-output relationships of the DC motor. These symptom vectors are stored in a fault database that has a dimension of 9×81, as illustrated in Fig. 7. The rule confidences are represented in greyscale ranging from full confidence (black) to no confidence (white).

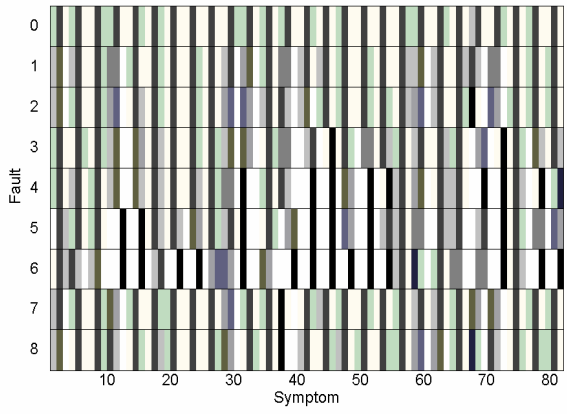

Fig. 7. Fault database

4.5 FDI Results

After the fault database is constructed, the proposed FDI technique is applied to detect the 9 faults given in Table 2. The fault isolated by the proposed FDI scheme is shown in Fig. 8. In all cases, the proposed FDI technique has selected the fault-free case before faults are being introduced into the system. After a fault has occurred, faults other than the fault-free case are selected by the proposed FDI technique, indicating faults have occurred in the system. The proposed FDI technique has successfully isolated the correct fault in all cases after the weights of NFN_{res} have settled on some constant values.

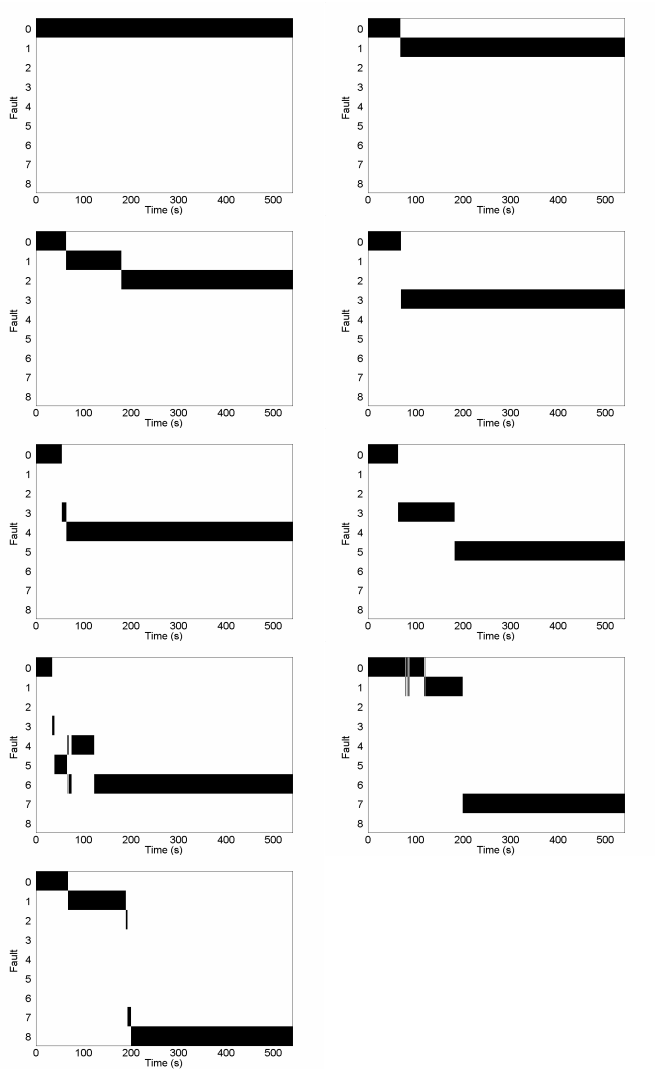

Fig. 8. FDI results (row 1: fault 0-1; row 2: fault 2-3; row 3: fault 4-5; row 4: fault 6-7; row 5: fault 8)

5. CONCLUSIONS

In this paper, an online FDI scheme based on neurofuzzy networks is proposed for monitoring nonlinear systems. Fuzzy rules describing the residual behaviours under different faulty operations are extracted from the neurofuzzy model and are stored into a fault database. The fault can then be isolated online based on the current behaviours and the patterns stored in the fault database by a nearest neighbour classifier. A simulation example involving a nonlinear DC motor control system is used to illustrate the implementation and performance of the proposed FDI scheme. It is shown that the proposed FDI scheme has successfully isolated the faults introduced in the DC motor.

REFERENCES

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

Chan, C.W., H. Song, and H.Y. Zhang (2006). Application of fully decoupled parity equation in fault detection and identification of DC motors. *IEEE Transactions on Industrial Electronics*, **53(4)**, 1277–1284.

Chen, J. and R.J. Patton (1999). *Robust model-based fault diagnosis for dynamic systems*, Kluwer Academic Publishers, Boston; Mass.

de Boor, C. (1972). On calculating with B-splines. *Journal of Approximation Theory*, **6(1)**, 50-62.

de Miguel, L.J. and L. F. Blázquez (2005). Fuzzy logic-based decision-making for fault diagnosis in a DC motor. *Engineering Applications of Artificial Intelligence*, **18(4)** ,423–450.

Frank, P.M., S.X. Ding, and B. Köppen-Seliger (2000). Current developments in the theory of FDI. *Proceeding of IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Budapest, Hungary, **1**, 16–27.

Friedman, M. and A. Kandel (1999). *Introduction to pattern recognition: statistical, structural, neural, and fuzzy logic approaches*, World Scientific, Singapore.

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

Korbicz, J., J.M. Kościelny, Z. Kowalczuk, and W. Cholewa (Eds.) (2004). *Fault Diagnosis. Models, Artificial Intelligence, Applications*, Springer-Verlag: Berlin.

Mok, H.T. and C.W. Chan (2005). Online fault diagnosis of nonlinear systems based on neurofuzzy networks, *Proceeding of 16th IFAC World Congress*, Prague, Czech Republic.

Nelles, O. (2001). *Nonlinear System Identification*. Springer-Verlag, Berlin.

Patton, R.J. , F.J. Uppal, and C.J. Lopez-Toribio (2000). Soft computing approaches to fault diagnosis for dynamic systems: A survey. *Proceeding of IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Budapest, Hungary, **1**, 303–315.

Watanabe, K., M. Sasaki, and D.M. Himmelblau (1985). Determination of optimal measuring sites for fault-detection of nonlinear-systems. *International Journal of Systems Science*, **16(11)**, 1345–1363.

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