FAULT DETECTION AND ISOLATION IN ROBOTIC MANIPULATORS USING A MULTILAYER PERCEPTRON AND A RBF NETWORK TRAINED BY THE KOHONEN'S SELF-ORGANIZING MAP

Renato Tinós and Marco Henrique Terra

tinos@sel.eesc.sc.usp.br, terra@sel.eesc.sc.usp.br
Departamento de Engenharia Elétrica - EESC / USP
Caixa Postal 359
São Carlos, São Paulo, Brazil, 13560-970

Abstract: In this work, Artificial Neural Networks are employed in a Fault Detection and Isolation scheme for robotic manipulators. Two networks are utilized: a Multilayer Perceptron is employed to reproduce the manipulator dynamical behavior, generating a residual vector that is classified by a Radial Basis Function Network, giving the fault isolation. Two methods are utilized to choose the radial unit centers in this network. The first method, Forward Selection, employs Subset Selection to choose the radial units from the training patterns. The second employs the Kohonen's Self-Organizing Map to fix the radial unit centers in more interesting positions. Simulations employing a two link manipulator and the Puma 560 manipulator indicate that the second method gives a smaller generalization error.

1 INTRODUCTION

The search for Fault Detection and Isolation (FDI) systems for robotic systems should increase in the next years due to the moving of robots from accessible areas (like laboratories and factories) to unstructured and hazardous environments. It is already usual to talk of robots in space and undersea exploration, in medicine, in nuclear plants and manipulating explosives. Furthermore, robots should be a common household item in the next future. In these environments, and even in factories, a faulty robot can cause irreversible damages and inadmissible economic losses..

Unfortunately, faults in robots have been usual. In a research made by the Japanese Ministry of Labor, 28.7% of the industrial robots studied had a mean-time-between-failure of 100h or less; 60 % had mean-time-between failure less than 500 h (Dhillon, 1991 ad in Groom *et al.*, 1999). Thus, there are good reasons to research FDI systems in robotic manipulators.

Usually, the FDI techniques employ the mathematical model to reproduce the dynamical behavior of the fault-free system. The outputs of the mathematical model are compared with the real measurements generating a residual vector that when properly analyzed, gives the fault information. However, modeling errors can obscure the faults and can be a false alarm source (Gertler, 1997). In several cases, it is necessary the use of robust techniques (Patton *et al.*, 1989; Mangoubi, 1998; Chen & Patton, 1999). Alternatively, a recurrent Artificial Neural Network (ANN) may be employed to reproduce the fault-free system dynamical behavior, generating the residual vector (Köppen-Seliger & P. M. Frank, 1996; Korbicz, 1997).

Generally in robotic manipulators, the researchers have proposed FDI schemes utilizing the system mathematical model (Visinsky *et al.*, 1995; Schneider & Frank, 1996; Naughton *et al.*, 1996; Vemuri & Policarpou, 1998), employing different methods for residual analysis. In (Terra & Tinós, 1998a) a Multilayer Perceptron (MLP) trained with the Backpropagation algorithm has been utilized to reproduce the dynamical behavior of a fault-free two link manipulator to generate the residual to be analysed.

The residual classification procedures have been received an important attention by several researchers, we have two central categories of analysis on the fault diagnosis: considering static and dynamic thresholds. For the static threshold we can see interesting applications in (Korbicz *et al.*, 1999; Patan & Korbicz, 2000), and references therein (we can see, too, in these references, other techniques based on fuzzy-logic sets and genetic algorithms applied on fault-diagnosis). Obviously it is hopped that a dynamic threshold should improve the diagnosis quality, decreasing false alarms.

For dynamics thresholds on the analysis of the residual classification, we have utilized Radial Basis Function Network (RBFN). This ANN has been trained using three different methods: the first, called Forward Selection (FS), employs Subset Selection to choose the radial unit centers from the training set; the second method (Global Ridge Regression - GRR) employs regularization, applying a penalty term in the large weights; the third method (Local Ridge Regression - LRR) employs regularization too, but instead of only one term, a penalty term is applied in each radial unit (Orr, 1996).

Artigo Submetido em 22/12/99
1a. Revisão em 31/05/00; 2a. Revisão em 27/09/00.
Aceito sob recomendação do Ed. Consultor Prof. Dr. Fernando Gomide

In this paper , a new RBFN training procedure is employed. In this training procedure, the RBFN centers are chosen by the Kohonen's Self-Organizing Map (SOM) that differs fundamentally of that three training procedures on the update of the radial unit centers. We choose the training FS method to be compared with the SOM procedure because it presents similarities on the choose of the radial unit centers. A comparing study among FS, GRR, and LRR in Fault Detection and Isolation applied in robotic manipulators can be seen in (Terra & Tinós, 1998b).

This paper is organized as follows: the FDI architecture is presented in Section 2. In Section 3, the training procedures employing the Kohonen's SOM and the FS are given. The simulation results employing a two link planar manipulator and the Puma 560 manipulator are presented in Section 4 and, finally, the conclusions comparing the two RBFN training procedures are given in Section 5.

2 FAULT DETECTION AND ISOLATION SCHEME

The dynamic of a fault-free robotic manipulator with actuators in each joint is given by

$$\ddot{\boldsymbol{\theta}}(t) = \mathbf{M}(\boldsymbol{\theta},t)^{-1}\left[\boldsymbol{\tau} - \mathbf{v}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) - \mathbf{g}(\boldsymbol{\theta},t) - \mathbf{z}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\boldsymbol{\tau},t) - \mathbf{d}(t)\right] \quad (1)$$

where $\boldsymbol{\theta}$ is the vector of joint angular positions, $\boldsymbol{\tau}$ is the vector of joint torques, $\mathbf{M}$ is the inertia matrix, t is the time index, $\mathbf{v}$ is the vector of Coriolis and centripetal terms, $\mathbf{g}$ is the vector of gravitational terms, $\mathbf{z}$ is the vector of friction terms and other nonlinearities and $\mathbf{d}$ is the vector of external uncorrelated disturbances.

The MLP mapping should reproduce the Equation (1) in order to generate the residual vector. However, the joint accelerations usually are not measured in real robotic manipulators. Considering a small sample rate Δt,

$$\ddot{\boldsymbol{\theta}}(t) = \frac{\dot{\boldsymbol{\theta}}(t+\Delta t) - \dot{\boldsymbol{\theta}}(t)}{\Delta t} \quad (2)$$

can be substituted in Equation (1),

$$\dot{\boldsymbol{\theta}}(t+\Delta t) = \mathbf{M}(\boldsymbol{\theta},t)^{-1}\left[\boldsymbol{\tau} - \mathbf{v}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) - \mathbf{g}(\boldsymbol{\theta},t) - \mathbf{z}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\boldsymbol{\tau},t) - \mathbf{d}(t)\right]\Delta t + \dot{\boldsymbol{\Theta}}(t). \quad (3)$$

This is the function that should be reproduced by the MLP mapping. The residual generation scheme is displayed in Figure 1.

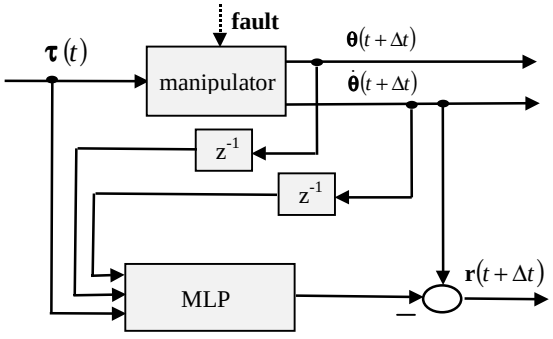

Figure 1. Residual generation (r is the residual vector).

The next step is to classify the residual vector and the joint velocities by the RBFN (Figure 2). The RBFN is employed here because, usually in FDI problems, it produces a more interesting classification than the MLP (Leonard & Kramer, 1991). Other advantages over the MLP: there are not local minimums in the calculation of the weights and the training time is smaller (Looney, 1997). However, the high number of radial units calls for more memory and the choice of the radial units and their parameters are suboptimal.

The RBFN outputs are trained to present signal 1 in the occurrence of the fault and 0 otherwise. The fault criteria is as follows: a specified number of sequential RBFN output samples have to be greater than 0.5 to a fault be detected. This criteria is adopted to avoid the occurrence of false alarms due misclassified individual patterns (Tinós, 1999). The sensitivity

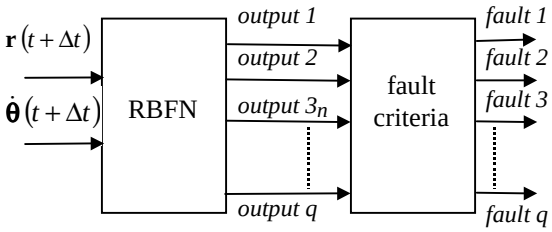

Figure 2. Residual analysis.

of the FDI system can be prejudiced adopting this criteria. However, if the sample rate is small, it does not represent a problem.

3 THE RADIAL BASIS FUNCTION NETWORK

The RBFN employed here has three layers. It does not exist weights between the first and the second layers. The second layer has neurons with radial activation functions. Each neuron j in the hidden layer (called radial unit j) is responsible for the creation of a receptive field in the p-dimensional input space. The receptive field of each radial unit is centered in a p-dimensional vector $\boldsymbol{\mu}_j$, called radial unit center. Thus, the radial unit j has activation according to the vector distance between the input vector and the radial unit center. There are weights between the hidden and the output layers and the activation in the last layer is linear.

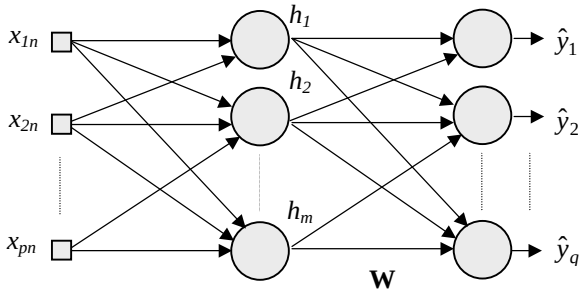

Figure 3. RBF network (W is the matrix of the weights w_{ki}).

Presenting in the sample n the input vector $\mathbf{x}_n=[x_{1n}\ x_{2n}\ \ldots\ x_{pn}]^{\mathrm{T}}$, the RBFN activation of the output neuron k (k=1,2,...,q) is given by

$$\hat{y}_k(\mathbf{x}_n) = \sum_{j=0}^{m} w_{kj} h_j(\mathbf{x}_n) \quad (4)$$

where m is the number of radial units, w_{kj} is the weight between the radial unit j and the output neuron k, and h_j is the activation of the radial unit j. In this work, the Cauchy radial function is

employed as activation function in the radial units. Thus, the activation of the radial unit j is given by

$$h_j(\mathbf{x}_n) = \frac{1}{1+\left\|\mathbf{R}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_j)\right\|^2} \quad (5)$$

where $\mathbf{R}$ is a diagonal matrix formed by the individual parameters defining the receptive field size in each dimension of the input space and $\|\ .\ \|$ defines a vector norm (here, the Euclidean norm).

If the RBFN has only three layers, the output layer activation is linear and the radial unit activations are determined, then this ANN can be viewed as a linear model. Thus, the training procedure can be made in two steps: the radial units and their parameters are determined and, then, the weights are calculated minimizing a cost function. In the following the problem of the ANN generalization will be described.

In order to achieve the best generalization, the complexity of the model (ANN) needs to be optimized. Considerable insight into this phenomenon can be obtained decomposing the generalization error into the sum of the bias squared plus the variance (Bishop, 1995). A model which is too simple, or too inflexible, will have a large bias, while a model which is too flexible in relation to the particular data set will have a large variance. Consider the mean-squared error applied to the patterns in the input space (Orr, 1996)

$$E_{MSE} = \left\langle \left(y(\mathbf{x}_n) - \hat{y}(\mathbf{x}_n)\right)^2 \right\rangle \quad (6)$$

where $\hat{y}(.)$ is the model prediction, $y(.)$ is the desired output and <.> is the expectation taken over the patterns in the input space. Breaking the Equation (6) in two (Geman *et al.*, 1992) we have

$$E_{MSE} = \left(y(\mathbf{x}_n) - \left\langle \hat{y}(\mathbf{x}_n)\right\rangle\right)^2 + \left\langle \left(\hat{y}(\mathbf{x}_n) - \left\langle \hat{y}(\mathbf{x}_n)\right\rangle\right)^2 \right\rangle \quad (7)$$

where the first part is the squared bias and the last part is the variance. The bias indicates the difference between the average of the model function output and the desired output. The variance indicates the sensitivity to the peculiarities (such as noise and choice of the patterns) of each particular training set. The best generalization is reached when there is the best compromise between the conflicting requirements of small bias and small variance. This may be obtained by controlling the flexibility of the model. To control the flexibility of the RBFN, regularization can be utilized to decrease the sensitivity to the training set and, thus, the variance of the model. Other approach is to control the flexibility changing the number of adaptive parameters in the RBFN. This is the approach utilized by the FS method.

3.1 Forward Selection

In this procedure, the RBFN flexibility is controlled by changing the number of adaptive parameters in the network. This is obtained comparing different architectures formed with different subsets of radial units chosen from the same training set. This is called Subset Selection in Linear Regression, usually employed to identify subsets of fixed functions with independent variables which can model most of the variation in the dependent variables. As the search of the optimal subset is prohibitive, a heuristic procedure is utilized to find a plausible one. FS is one of these procedures (Rawlings, 1988).

Chen *et al*. (1991) employed FS to choose the radial units in an RBFN. FS starts with an empty subset and adds a radial unit at time: the one which most reduces the sum-of-squared errors function. The procedure may be more efficient by using a technique called Orthogonal Least Squares that ensures that the new column formed by the radial unit activations for the training set added to the matrix of the growing subset is orthogonal to all previous columns. To halt the selection (stop criteria), a fixed threshold on the variance may be utilized. However the Generalized Cross-Validation may be utilized to halt the Subset Selection, being more effective than a fixed threshold (Orr, 1995). When the selection of the radial units is finished, the optimal weights can be calculated. Minimizing the sum-of-squared errors function, the optimal weight matrix is given by

$$\hat{\mathbf{W}} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{Y} \quad (8)$$

where the n_p x m matrix $\mathbf{H}$ is formed by the radial unit activations h_j for the different training patterns (n_p is the number of training patterns) and the n_p x q output matrix $\mathbf{Y}$ is formed by the training desired outputs (q is the number of neurons in the output layer). Thus, the procedure for the FS is:

1. Start with an empty subset of radial units;
2. Add the radial unit (chosen from the training set) that most reduces the sum-of-squared error;
3. If the stop criteria is satisfied, go to the next step; otherwise return to step 2;
4. Compute the matrix $\mathbf{H}$ composed by the radial unit activations (eq. 5) for the training set;
5. Compute the matrix of weights $\mathbf{W}$ (eq. 8).

In the following will be presented the Kohonen's SOM with an alternative to the RBFN training in FDI.

3.2 The Kohonen's SOM

The use of the Kohonen's SOM in the training of RBFN's is not a new approach. In (Ojala & Vuorimaa, 1995) for example, Kohonen's SOM is employed to find the initial centers of the radial units and, then, a modified Learning Vector Quantization (LVQ2.1) algorithm (Kohonen, 1995) is utilized to tune all the parameters of an RBFN. Here, the Kohonen's SOM will be applied to train an RBFN employed to pattern recognition in a FDI scheme. Some changes will be made to adequate the Kohonen's SOM in this problem. The first step is to separate the training set according to different classes. This procedure is adopted to prevent that patterns belonging to different classes to be tuned on the same radial unit. Thus, the algorithm described below should be repeated for each class.

Initially, all patterns of each class are chosen as radial unit centers. The neuron activations of all radial units for each training pattern are calculated employing the Equation (5) and the unit with the highest activation is selected according to

$$h_c(t) = \max_j \left\{h_j(\mathbf{x}(t))\right\} \quad (9)$$

where j=1,...,m_k (m_k is the number of patterns in the class k), k=1,...,q (q is the number of classes) and t=1,...,$t_{\max}$ is the discrete-time coordinate. The next step is to update the radial unit centers according to

$$\boldsymbol{\mu}_j(t+1) = \boldsymbol{\mu}_j(t) + \alpha(t)\beta(t)\left[\mathbf{x}(t) - \boldsymbol{\mu}_j(t)\right] \qquad (10)$$

where $\alpha(t)$ is a decaying function of time that defines the learning rate and $\beta(t)$ is a function of the vector distance from the radial unit center $\boldsymbol{\mu}_j$ to the radial unit center $\boldsymbol{\mu}_c$. Here, this function is given by

$$\beta(t) = \begin{cases} \dfrac{1}{1 + \left\| \mathbf{R}^{-1}\left(\boldsymbol{\mu}_c - \boldsymbol{\mu}_j\right) \right\|^2}, & \text{if } \left\| \mathbf{R}^{-1}\left(\boldsymbol{\mu}_c - \boldsymbol{\mu}_j\right) \right\| < \sigma(t), \\ 0, & \text{otherwise} \end{cases}$$

where $\sigma(t)$ is a decaying function of time that defines the neighborhood size around the radial unit center $\boldsymbol{\mu}_c$. If the number of iterations (t_{max}) is sufficiently large and the training parameters are chosen appropriately, the radial unit centers in the same cluster will move to the cluster center. Thus, as some radial units have centers very close, they will be grouped. This is made calculating the distance between the radial unit centers. If the norm of the distance of two radial unit centers is very small, one radial unit is pruned. Thus, the complexity of the network is reduced because the number of adaptive parameters decreases. This procedure is important because if there are radial unit centers very close, then the matrix inverse used to determine the optimal weights (Equation 8) will have ill-posed problems.

Performing the training procedure above for all classes, the radial unit centers are grouped in a single RBFN and the optimal weight matrix can be calculated using the Equation (8). A simple example will be presented bellow to demonstrate the effectiveness of this training procedure.

Example 1: Consider that we have to classify 2-dimensional patterns in two classes. For the training, we have 40 patterns (20 in each class). The patterns in the first class are generated with mean=$[0.5\ \ 0.5]^T$ and variance=$[0.01\ \ 0.09]^T$ and in the second class with mean=$[0\ \ 0]^T$ and variance=$[0.01\ \ 0.09]^T$. The Figure 4 and 5 show respectively the training patterns and the receptive fields formed by the RBFN trained by FS and by Kohonen's SOM. The two training algorithms described previously are employed. The training set and the diagonal of the matrix $\mathbf{R}$=$[0.2\ \ 0.6]^T$ that defines the size of the receptive field are the same for the two procedures. The FS uses a fixed threshold to halt the radial unit selection and selects two radial units: the first centered in $[0.047\ \ -0.083]^T$ and the second in $[0.578\ \ 0.589]^T$. The Kohonen's SOM algorithm selects two radial units centered in $[0.015\ \ 0.039]^T$ and in $[0.513\ \ 0.560]^T$. For the generalization test, 200 patterns with the same characteristics of the training set are generated. The Figure 6 displays the generalization test patterns and the receptive field formed by the RBFN trained by FS and the Figure 7 displays for the RBFN trained by Kohonen's SOM. It can be noted that the RBFN trained by Kohonen's SOM presents a more interesting classification.

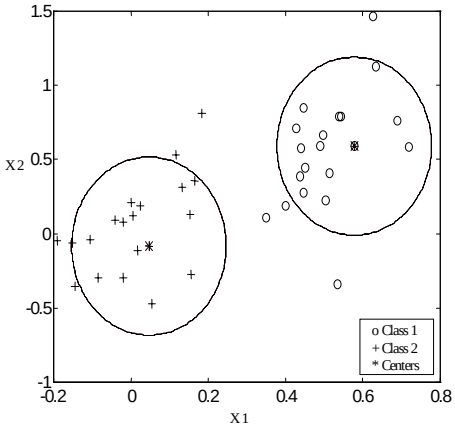

Figure 5. Training patterns and the receptive field created by the RBFN trained by FS.

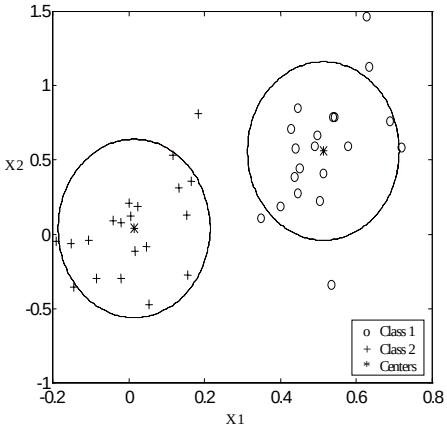

Figure 6. Training patterns and the receptive field created by the RBFN trained by Kohonen's SOM.

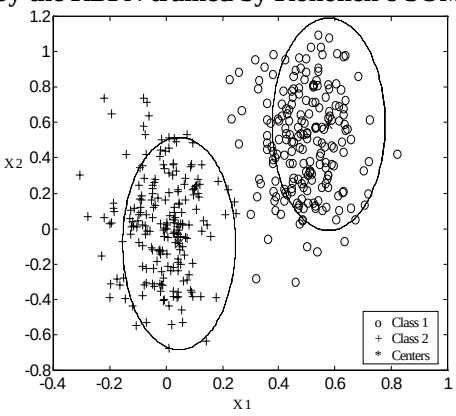

Figure 6. Test patterns and the receptive field created by the RBFN trained by FS.

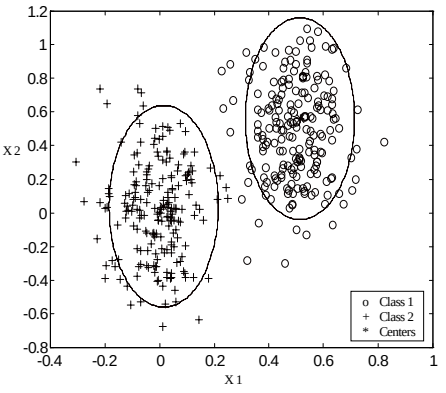

Figure 7. Test patterns and the receptive field created by the RBFN trained by Kohonen's SOM.

4 SIMULATION RESULTS

The FDI scheme is applied in two robotic manipulators simulated in MATLAB. The first is a two-link planar manipulator and the second is a Puma 560 manipulator (Corke, 1996). For both systems, the FS and the Kohonen's SOM are used to train the RBFN's.

4.1 Two link manipulator results

For this system, two faults are considered: in the first, the joint 1 is locked and in the second, the joint 2 is locked. This kind of faults can be caused when, for example, an actuator is locked in a fixed position or when a brake is improperly applied (Lewis & Maciejewski, 1997). The MLP utilized to reproduce the manipulator dynamical behavior has 6 inputs, 13 neurons in the hidden layer and 2 outputs. The MLP is trained by backpropagation. The training set has 500 patterns and noises are added to the measures of joint positions (mean=0 and variance=0.005) and velocities (mean=0 and variance=0.05). After 18000 steps of training, the sum-of-squared error for the training set is equal to 5.21×10^{-5}. After trained, the MLP reproduces the manipulator dynamical behavior presenting a small residual vector for nontrained fault-free trajectories. The Figure 8 shows the residues of a nontrained trajectory in that occurred fault 1.

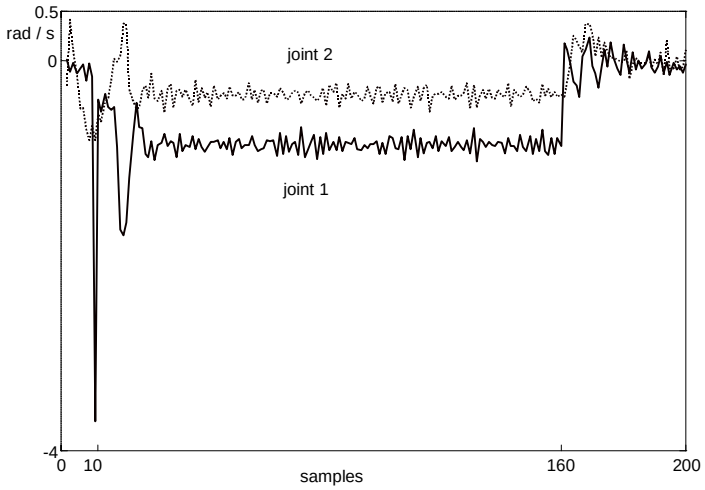

Figure 8. Residues for the nontrained trajectory from $\theta=[\pi/6\ \pi/4]^T$ to $\theta=[2\pi/3\ \pi/3]^T$. The fault 1 occurs between the samples 10 and 160.

The RBFN employed has 4 inputs (the 2-dimensional residual vector and the 2-dimensional joint velocities vector) and 2 outputs (one for each fault). The diagonal of the matrix **R** is $[0.05\ 0.035\ 0.1\ 0.1]^T$ where the two first variables define the receptive field size for the residual vector and the last two define the receptive field size for the joint velocities. For the RBFN training, 9 trajectories with 40 samples each are employed. During the training procedure, the trajectories are presented to the RBFN three times: one for each fault and one for the fault-free operation, performing a total of 1040 patterns. For the generalization test, 17 nontrained trajectories with 100 samples are presented three times (faults 1 and 2 and fault-free operation) to the RBFN.

During the RBFN training, FS selects 415 radial units and Kohonen's SOM selects 215 radial units. The Table 1 and 2 show the sum-of-squared errors for the training and the test set for the RBFN's. The Figures 9 and 10 display the results for a nontrained trajectory in that the fault 1 occurs. To detect and isolate the faults, the criteria adopted is: five consecutive outputs have to be greater than 0.5. The results of the FDI with RBFN's trained by the two procedures are displayed in Table 3.

Table 1. Sum-of-squared errors for the training set.

	Output 1	Output 2
Forward Selection	0.0008	0.0042
Kohonen's SOM	0.0100	0.0212

Table 2. Sum-of-squared errors for the test set.

	Output 1	Output 2
Forward Selection	0.0699	0.2727
Kohonen's SOM	0.0273	0.0788

Table 3. FDI Results for the test set.

	Number of trajectories in that occurred false alarms	Number of trajectories with faults do not detected
Forward Selection	1 (1.85 %)	0 (0 %)
Kohonen's SOM	0 (0 %)	0 (0 %)

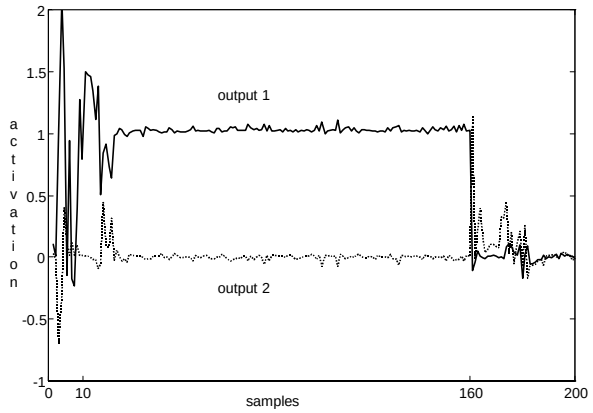

Figure 9. Outputs of the RBFN trained with FS for the nontrained trajectory from $\theta=[\pi/6\ \pi/4]^T$ to $\theta=[2\pi/3\ \pi/3]^T$. The output 1 indicates the fault 1 and the output 2 the fault 2, according to the fault criteria. The fault 1 occurs between the samples 10 and 160.

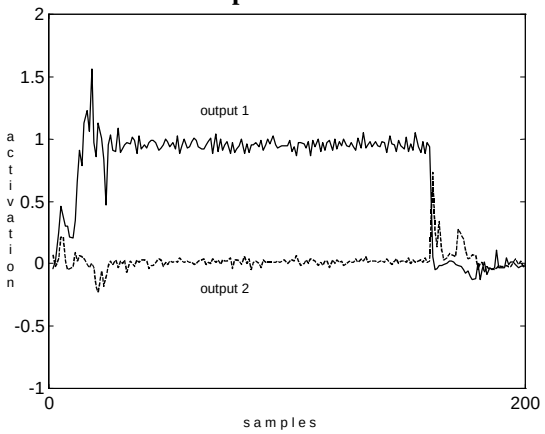

Figure 10. Outputs of the RBFN trained with Kohonen's SOM the nontrained trajectory from $\theta=[\pi/6\ \pi/4]^T$ to $\theta=[2\pi/3\ \pi/3]^T$. The fault 1 occurs between the samples 10 and 160.

4.2 Puma 560 manipulator results

Three faults are considered in the Puma 560 manipulator simulations. In the fault 1, the torque in the joint 1 is set to zero (free-swinging joint fault). In the fault 2, the torque in the joint 2 is set to zero and the same occurs for the joint 3 in the fault 3. The MLP has 9 inputs, 29 neurons in the hidden layer and 3 outputs. Free-swinging joint fault can be caused, for example, by a rupture seal on a hydraulic actuator, by a loss of electric power and brake in a electric actuator and by a mechanical fault in a drive system (English and Maciejewski, 1998). The MLP is trained by backpropagation. The training set has 8100 patterns. After 8000 steps of training, the sum-of-squared error for the training set is equal to 1.58×10^{-5}. After trained, the MLP reproduces the manipulator dynamical behavior presenting a small residual vector for nontrained fault-free trajectories (Figure 11).

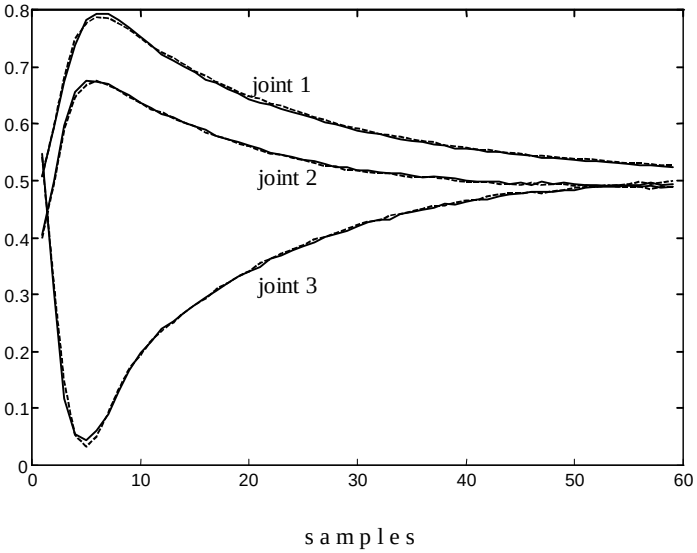

Figure 11. The first 3 Normalized joint velocities (solid line) and respective outputs of the MLP (dashed line) for a nontrained trajectory from $\theta=[0\ 0\ 0\ 0\ 0\ 0]^T$ to $\theta=[\pi\ \pi/3\ -5\pi/6\ \pi/6\ \pi/2\ 5\pi/12]^T$.

The RBFN employed has 6 inputs (the 3-dimensional residual vector and the 3-dimensional joint velocities vector) and 3 outputs. The diagonal of the matrix **R** is $[0.022\ 0.027\ 0.032\ 0.8\ 0.8\ 0.8]^T$ where the three first variables define the receptive field size for the residual vector and the last three for the joint velocities. For the RBFN training, 15 trajectories with 12 samples each are employed. During the training procedure, the trajectories are presented to the RBFN four times: one for each fault and one for the fault-free operation, performing a total of 720 patterns. For the generalization test, 30 nontrained trajectories with 15 samples are presented four times (faults 1, 2 and 3 and fault-free operation) to the RBFN. During the RBFN training, FS selects 251 radial units and Kohonen's SOM selects 255. The Figures 12 and 13 display the results fornontrained trajectories (in the first three, a different fault occurs in each trajectory and in the last one, none fault occurs).

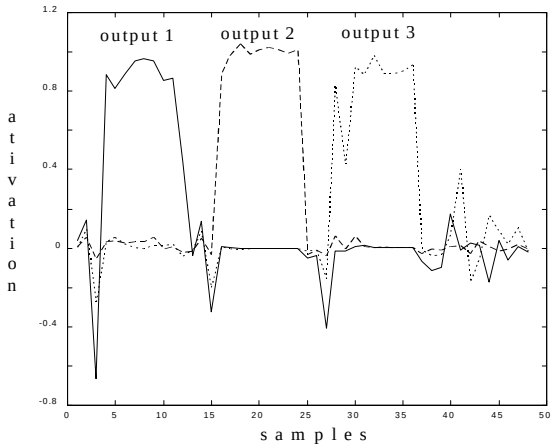

Figure 12. Outputs of the RBFN trained via FS with 4 trajectories from $\theta=[0\ 0\ 0\ 0\ 0\ 0]^T$ to $\theta=[\pi\ \pi/3\ -5\pi/6\ \pi/6\ \pi/2\ 5\pi/12]^T$. In each trajectory, a different fault occurs and in the last one, none fault occurs.

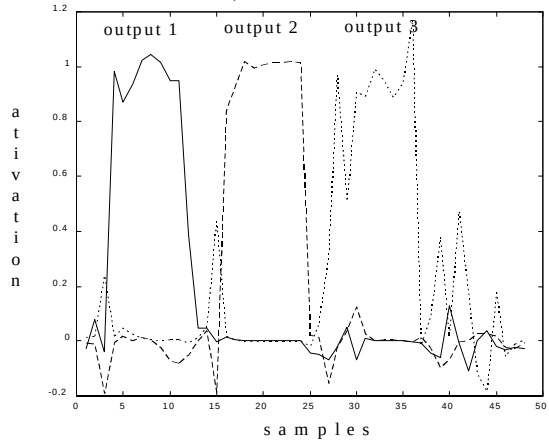

Figure 13. Outputs of the RBFN trained with Kohonen's SOM for 4 trajectories from $\theta=[0\ 0\ 0\ 0\ 0\ 0]^T$ to $\theta=[\pi\ \pi/3\ -5\pi/6\ \pi/6\ \pi/2\ 5\pi/12]^T$. In each trajectory, a different fault occurs and in the last one, none fault occurs.

The Table 4 and 5 show the sum-of-squared errors for the training and the test set for the RBFN's. To detect and isolate the faults, the criteria adopted is: 3 consecutive faults have to be greater than 0.5. The results of the FDI for the two procedures are displayed in Table 6.

Table 4. Sum-of-squared errors for the training set.

	Out. 1	Out. 2	Out. 3
F. S.	0.0040	0.0017	0.0017
Kohonen's SOM	0.0096	0.0042	0.0092

Table 5. Sum-of-squared errors for the test set.

	Out. 1	Out. 2	Out. 3
F. S.	0.0712	0.0431	0.0347
Kohonen's SOM	0.0462	0.0400	0.0293

Table 6. FDI Results for the test set.

	Number of trajectories in that occurred false alarms	Number of trajectories with faults do not detected
Forward Selection	8 (6.67 %)	2 (1.67 %)
Kohonen's SOM	4 (3.33 %)	0 (0 %)

5 CONCLUSIONS

The FDI scheme employed presents good results when it is applied in a two link manipulator and in a Puma 560 manipulator. According to the tests, the FDI scheme can detect and isolate faults that occur in nontrained trajectories.

In this work, the RBFN's trained by FS have presented smaller errors for the training set than the RBFN's trained by Kohonen's SOM. This has occurred because the RBFN's trained by FS have presented smaller bias in the sum-of-squared errors than for the RBFN's trained by Kohonen's SOM. However, the RBFN's trained by Kohonen's SOM have presented the smallest errors for the generalization test set indicating that the variance in the sum-of-squared errors have been smaller than for the RBFN's trained by FS. This shows that the RBFN's trained by FS have been more sensitivity to the peculiarities (such as noise and choice of the patterns) of the training sets because of the radial unit centers are chosen from the training patterns. The RBFN's trained by Kohonen's SOM have been more flexible, presenting the best generalization results because of the radial unit centers have been fixed near of the cluster center. Furthermore, in the FS procedure, local minima could occur in the radial unit search. Nevertheless, it is important to accentuate that these results have been dependent on the choice of the parameters in each training procedure and the choice of the parameters in the Kohonen's SOM is more complicate than for the FS. It is noted too in this work, that the Kohonen's SOM has been simpler than the FS because it does not employ complex calculus involving great matrices.

Acknowledgements: This work is supported by FAPESP under grants nr. 98/15732-5 and 99/10031-1.

REFERENCES

Bishop, C. M. (1995). "*Neural networks for pattern recognition*". Oxford University Press.

Chen, J. & Patton, R. J. (1999). "*Robust model-based fault diagnosis for dynamic systems*", Kluwer Academic Publishers.

Chen, S., Cowan, C. F. & Grant, P. M. (1991). "Orthogonal least squares learning algorithm for radial basis function networks". *IEEE Trans. on Neural Networks*, vol. 2, n. 2: p. 302-309.

Corke, P. I. (1996). "A robotics toolbox for MATLAB", *IEEE Transactions on Robotics & Automation Magazine*, vol. 3, n. 1, p.24-32.

Dhillon, B. S. (1991). "*Robotic reliability and safety*", New York: Springer-Verlag.

English, J. D. & Maciejewski, A. A. (1998). "Fault tolerance for kinematically redundant manipulators: anticipating free-swinging joint failures", *IEEE Trans. on Robotics and Automation*, vol. 14, n. 4: p. 566-575.

Geman, S., Bienenstock, E. & Doursat, R. (1992). "Neural networks and the bias/variance dilemma". *Neural Computation*, vol. 4, n. 1: p. 1-58.

Gertler, J. (1997). "A cautious look at robustness in residual generation". *Proc. of the IFAC Simp. On Fault Detection, Supervision and Safety for Tech. Processes*, Kingston Upon Hull, U.K., vol. 1: p. 133-139.

Groom, K. N., Maciejewski, A. A. & Balakrishnan, R. (1999). "Real-time failure-tolerant control of kinematically redundant manipulators". *IEEE Transactions on Robotics and Automation*, vol. 5, n. 6: p. 1109-115.

Kohonen, T. (1995). "*Self-organizing maps*", Springer-Verlag, Berlin.

Köppen-Seliger, B. & Frank, P. M. (1996). "Neural Networks in Model-Based Fault Diagnosis", *Proc. of the 13th IFAC World Congress*, San Francisco, USA, p. 67-72.

Korbicz, J. (1997). "Neural networks and their application in fault detection and diagnosis", *Proc. of the IFAC Simp. on Fault Detection, Sup. and Safety for Tech. Proc.*, Kingston Upon Hull, UK, vol. 1: p. 377-382.

Korbicz, J., Patan, K. & Obuchowicz, A (1999). "Dynamic neural networks for process modelling in fault detection and isolation systems", International Journal of Applied Mathematics and Computer Science, vol. 9(3), p. 519-546.

Leonard , J. A. & Kramer, M. A. (1991). "Radial basis function networks for classifying process faults", *IEEE Control Systems*, vol. 11, 3, p.31-38.

Lewis, C. L. & Maciejewski, A. A. (1997). "Fault tolerant operation of kinematically redundant manipulators for locked joint failures", *IEEE Trans. on Robotics and Automation*, vol.13, n.4, p.622-629.

Looney, C. G. (1997). "*Pattern recognition using neural networks*", Oxford University Press.

Mangoubi, R. S. (1998). "*Robust estimation and failure detection*", Springer-Verlag.

Naughton, J. M.; Chen, Y. C. & Jiang, J. (1996)."A neural network application to fault diagnosis for robotic manipulator", *Proc. of the IEEE Int. Conf. on Control Applications*, vol. 1: p. 988-1003.

Ojala, T. & Vuorimaa, P. (1995). "Modified Kohonen's learning laws for RBF networks", *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, Ales, France, p.356-359.

Orr, M. J. L. (1995). "Regularization in the selection of radial basis function centers". *Neural Computation*, 7: p. 606-623.

Orr, M. J. L. (1996). "*Introduction to radial basis function networks*". Center for Cognitive Science, Edinburgh University, Scotland, U. K.

Patton, R. J., Frank, P. M. & Clark, R. N. (1989). "*Fault diagnosis in dynamic systems - theory and application*". Prentice Hall Int..

Patan, K. & Korbicz, J. (2000). "Application of dynamic neural networks in an industrial plant". 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes, Budapest, Hungary, p. 186-191.

Rawlings, J. O. (1988). "*Applied regression analysis - a research tool*". Wadsworth & Brooks / Cole Advanced Books & Software.

Schneider, H. & Frank, P. M. (1996). "Observer-based supervision and fault-detection in robots using nonlinear and fuzzy logic residual evaluation", *IEEE Trans. on Control Systems Technology*, vol. 4, n. 3: p. 274-282.

Terra, M. H. & Tinós, R. (1998a). "Fault detection and isolation for robotic systems using a multilayer perceptron and a radial basis function network". I*n the Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, San Diego, USA.

Terra, M. H. & Tinós, R. (1998b). "Fault Detection and Isolation in Robotic Systems via Artificial Neural Networks" - The 37th IEEE Conference on Decision and Control - CDC98 - December - Tampa - USA.

Tinós, R. (1999). "*Detecção e diagnóstico de falhas em robôs manipuladores via redes neurais artificiais*", Dissertação de Mestrado, Escola de Eng. De São Carlos, USP, 118 p.

Vemuri, A. T. & Polycarpou, M. M. (1997). "Neural-network-based robust fault diagnosis in robotic systems", *IEEE Trans. on Neural Networks*, vol. 8, n. 6: p. 1410-1420.

Visinsky, M. L., Cavallaro, J. R. & Walker, I. D. (1995). "A dynamic fault tolerance framework for remote robots". *IEEE Trans. on Robotics and Automation*, vol. 11, n. 4: p. 477-490.