Neural Networks in Process Fault Diagnosis

Timo Sorsa, Heikki N. Koivo, *Senior Member, IEEE*, and Hannu Koivisto

Abstract—**Fault detection and diagnosis is currently a very important problem in process automation. Both model-based methods and expert systems have been suggested to solve the problem. Pattern recognition approach has also been investigated. Recently neural networks have been advocated as a possible technique. A number of possible neural network architectures for fault diagnosis are studied. The multilayer perceptron network with a hyperbolic tangent as the nonlinear element seems best suited for the task. As a test case, a realistic heat exchanger–continuous stirred tank reactor system is studied. The system has 14 noisy measurements and 10 faults. The proposed neural network was able to learn the faults in under 3000 training cycles and then to detect and classify the faults correctly. Principal component analysis is used to illustrate the fault diagnosis problem in question.**

I. Introduction

FAULT DETECTION AND DIAGNOSIS have become more and more important in a number of processes. A good example of its relevance is a nuclear power plant, where after a fault tens of alarms can occur in a few seconds. Locating the fault might be of utmost importance because of safety. In other production processes a serious fault can easily mean expenses of more than $10,000 in an hour. For these reasons both the research community and industry, especially digital automation manufacturers are concentrating their efforts to solve the problem.

A number of useful techniques have been suggested in the literature. So-called model-based techniques have been discussed e.g., in four excellent survey papers [1], [5], [13], [35]. A problem with model-based techniques appears to be the limited class of problems that they can be applied to and the heavy computational machinery required.

An early idea has been to approach the problem from the operator point of view. Then pattern recognition rises as a natural technique, and later also knowledge-based methods. The books of Himmelblau [10] and Pau [26] give a good idea of the former, and Kramer and Palowitch [19] and Rich and Venkatasubramanian [28] of the latter. A recent idea has been to use artificial neural networks to represent the knowledge and do the pattern recognition task with these [12], [22]. The idea is to mimic the human operator behavior even more. The neural net is first taught the different fault situations. After the network has learned them, it can do the proper fault detection and diagnosis.

Hoskins and Himmelblau [12] illustrated an artificial neural network approach to the fault diagnosis of a simple example process composed of three continuous-stirred-tank reactors in series. Watanabe *et al.* [34] presented a network architecture to estimate the degree of failures. They used an example system of three measurements and five faults. Venkatasubramanian and Chan [32] used a binary-input network to diagnose faults of a fluidized catalytic cracking process. They also compared the neural network approach with knowledge-based approach. Naidu *et al.* [22] studied a sensor failure detection system based on a multilayer perceptron network and back-propagation learning algorithm. They also discussed traditional fault detection algorithms.

The present paper differs from the previous ones in the way that here a wider variety of possible network architectures for fault detection and diagnosis are studied. In addition, a more realistic process is used as a test case. Every fault situation has effects on many measurements and the relationships between the faults and the measurements are not obvious at all. Principal component analysis [2] is used as a transparent way to visualize the difficulties in fault diagnosis.

The paper begins with a short summary of different fault detection methods. The neural network architectures suitable for fault detection and diagnosis are then reviewed. The technique is then tested on a simulated heat exchanger-continuous stirred tank reactor system with 14 noisy measured variables and 10 faults.

II. Fault Detection and Diagnosis

When a process fault appears, it has to be detected as early as possible. The fault detection system must indicate that something is going wrong in the process. After the detection, the fault diagnosis is carried out. The fault is isolated and the cause of the fault is located. Usually the techniques used in fault detection and diagnosis are divided into two general categories: estimation methods and pattern recognition methods [10], [26].

A. Estimation Methods

Estimation methods require mathematical process models that represent the real process satisfactorily. The model should not, however, be too complicated, because calculations easily become very time-consuming.

Fault Detection Based on State Variable Estimation: All state variables are seldom measurable and so nonmeasurable state variables have to be estimated. For estimation a dynamic process model is linearized around an operating point. The estimation can be done using different methods depending on how stochastic the model is [13]. Then the residuals, i.e., the differences between the estimated and the actual measured variables, are generated and the fault detection is carried out

Manuscript received May 25, 1990; revised October 28, 1990 and February 16, 1991. This work was supported in part by the Finnish Academy.

The authors are with Tampere University of Technology, Department of Electrical Engineering, P. O. Box 527, SF-33101, Tampere, Finland.

IEEE Log Number 9144664.

using statistical testing methods [13]. This approach requires a relatively exact knowledge of the parameters of the linearized model. In addition the process must operate near the point where linearization was done because the model is valid only in the neighborhood of the operating point.

Fault Detection Based on Parameter Estimation: In many cases the process model parameters have a complicated relationship to physical process coefficients. Malfunctions usually effect the physical coefficients and the effect can also be seen in the process parameters. Because not all physical process parameters are directly measurable their changes are calculated via estimated process model parameters. The relationship between the model parameters and the physical coefficients should be unique and preferably exactly known. This seldom is the case.

B. Pattern Recognition Methods

No mathematical model of the process is necessarily needed in pattern recognition methods for fault detection and diagnosis. The idea is that the operation of the process is classified according to measurement data. Formally this is a mapping from measurement space into decision space.

Traditional pattern recognition and classification can be divided into three stages: measurements, feature extraction, and classification. First the appropriate data is measured. Then a feature vector is computed. The extraction should remove redundant information from measurement data and create simple decision surfaces. Finally the feature vector is classified into one or more classes. When fault detection and diagnosis are combined, the classes are the following: normal operation, fault number 1, fault number 2, etc. Traditionally pattern recognition concentrates in finding the classification of features. The problem is, how to calculate the features. There is no *a priori* basis for the choice of the calculation. It's difficult to know which of the features are essential and which are irrelevant. Inappropriate choices lead to the need for complex decision rules, whereas good choices result in simple rules [10], [25].

A human being has an amazing skill to recognize patterns. A human uses often very complex logic in recognizing patterns and in classification. A human can pick up some examples of the classes but cannot determine any formal law for the classification. Mathematically this can be seen as an opaque mapping from pattern space into class-membership space. In computer-implemented pattern recognition the opaque mapping has to be replaced by an explicitly described procedure—a transparent mapping [25].

Like human beings neural networks are also trained with a group of examples. When a classification is realized with neural networks, the whole mapping from measurement space into decision space is done at the same time and the mapping is learned by training examples.

C. A Comparison

The performance of model-based methods depends strongly on the usefulness of the model. The model must include every situation under study. It must be able to handle changes in the operation point. If the model fails, the whole diagnostic system fails. The sensitivity to modelling errors has become the key problem in the application of model-based methods [5]. Pattern recognition methods do not require analytical model of the process but need representative training data. Pattern recognition methods get all their information from the data and so the data play a very important role in this method.

Model-based methods are usually computationally demanding especially when nonlinear models are used. On-line modelling of the process requires even more computational expenditure [5]. Pattern recognition methods are usually computationally easier but the calculation task depends very much on the data and the actual problem.

Model-based methods are mostly difficult to change afterward. The build-up of a model-based diagnostic system requires a lot of effort and changing one equation leads easily to changes in many other equations or parameters. Pattern recognition methods are more flexible in the sense that changing the data means the same as changing the properties of the diagnostic system. On the other hand changing the data may lead to repeating the pattern recognition task all over again.

III. Neural Networks

Recently neural networks have been studied very intensively. New architectures and learning algorithms are developed all the time. Even though the present neural network models don't achieve human-like performance, they offer interesting means for pattern recognition and classification. The traditional pattern recognition includes a large collection of very different types of mathematical tools (preprocessing, extraction of features, final recognition). In many cases it is difficult to say what kind of tool would best fit to a particular problem. Neural networks make it possible to combine these steps, because they are able to extract the features autonomously. They are practical to use, because they are nonparametric. It has also been reported that the accuracy of neural classifiers is better than of traditional ones [7], [22].

In this paper we study three well-known architectures: a single-layer and a multilayer perceptron network, and a counterpropagation network. The network must be able to handle *continuous input data* and *the learning must be supervised*, in order to solve the fault detection and diagnosis problem. The single-layer perceptron network is a traditional network used for classification in many cases. The multilayer perceptron network is the most common network today. It is used in many classification and mapping problems, which a single-layer perceptron is not able to solve [7], [12], [29]. The counterpropagation network [9] can also be used to implement continuous mappings. Its architecture is totally different from those of perceptrons and so its capabilities and limitations are considerably different too. Next we briefly summarize the networks used.

A. Single-Layer Perceptron

The perceptron network is composed of many nonlinear computational elements, which are arranged in different layers.

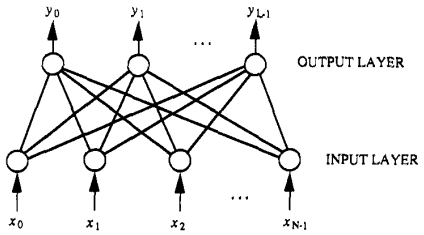


Fig. 1. Single-layer perceptron.

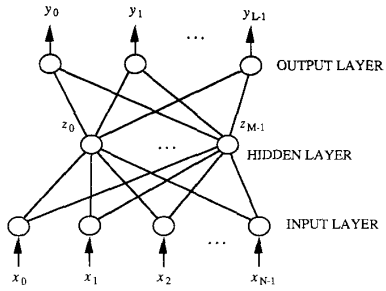


Fig. 2. Two-layer perceptron network.

The single-layer perceptron consists of an input and an output layer. The input layer works as a buffer that may include some preprocessing of the input data. The actual nonlinear elements or nodes are in the output layer. The outputs of the network are calculated from the formula

$$y_j = f\left(\sum_{i=0}^{N-1} w_{ij}x_i - \theta_j\right) \quad \text{for } j = 0, 1, \cdots, L-1 \qquad (1)$$

where x_i are inputs, y_j outputs, w_{ij} weights, θ_j internal offsets (bias or threshold) and the function f is the nonlinearity, usually hard limiter, threshold logic, sigmoid function or hyperbolic tangent. Fig. 1 presents a single-layer perceptron. Circles denote nodes and lines are connections.

To be able to classify data the perceptron network has to adapt its weights so that the network can perform the desired mapping. The adaptation is achieved through a learning process where the weights are altered according to a learning algorithm. For the single-layer perceptron with continuous-valued inputs and outputs the delta rule is the learning algorithm that minimizes the error between the actual output of the network and the desired output [25], [29].

B. Multilayer Perceptron

The problem with the single-layer perceptron is that it can produce only linearly separable decision regions [20]. This severely restricts the use of the single-layer perceptron. More general decision regions are achieved with the multilayer perceptron network that consists of three or more layers; an input layer, one or more hidden layers and an output layer. A network with one hidden layer is shown in Fig. 2.

In this case the outputs of the hidden nodes are calculated from the following formula

$$z_j = f\left(\sum_{i=0}^{N-1} w'_{ij}x_i - \theta'_j\right) \quad \text{for } j = 0, 1, \cdots, M-1 \qquad (2)$$

where x_i are the inputs of the network, z_j the outputs of the hidden layer, w'_{ij} the weights between the input and hidden layer, θ'_j internal offsets in the hidden nodes and f is the nonlinearity. The outputs of the network are calculated from the formula

$$y_k = f\left(\sum_{j=0}^{M-1} w_{jk}z_j - \theta_k\right) \quad \text{for } k = 0, 1, \cdots, L-1 \qquad (3)$$

where w_{jk} are the weights between the hidden and output layer and θ_k internal offsets in the output layer.

Cybenko [3], Funahashi [6], and Hornik *et al.* [11] have proved that a two-layer perceptron network (Fig. 2) can approximate any continuous mapping with an arbitrary accuracy, provided sufficiently many hidden units are available. Cybenko [3] has also shown that arbitrary decision regions can be arbitrarily well approximated by a two-layer perceptron network. Unfortunately the papers don't tell how these accurate approximations can be achieved.

The Back-Propagation Algorithm: For multilayer perceptrons Rumelhart *et al.* [29] have created a learning algorithm that minimizes the mean square error between the desired and the actual output of the network. The optimization is carried out with a gradient-descent technique. The algorithm is called the back-propagation algorithm or the generalized delta rule.

The back-propagation algorithm does not always find the global minimum but may stop at a local minimum. In practice the type of minimum has little importance as long as the desired mapping or classification is reached with a desired accuracy. The optimization criterion of the back-propagation algorithm is not very good from the pattern recognition point of view. The algorithm minimizes the square error between the actual and the desired output—not the number of faulty classifications, which is the main goal in pattern recognition. The algorithm is too slow for practical applications especially if many hidden layers are used [25], [33]. In addition a back-propagation net has poor memory. When the net learns something new it forgets the old one. Despite of its shortcomings the back-propagation is very widely used.

C. Counterpropagation Network

The counterpropagation network combines the self-organizing map of Kohonen and the outstar structure of Grossberg [8], [9], [33]. The forward-only version of the counterpropagation network is shown in Fig. 3. It consists of three layers. The first is the input layer that works as a buffer to the inputs $x_0, x_1, \cdots, x_{N-1}$. Then the Kohonen layer classifies the input data. The learning in the Kohonen layer is unsupervised [17] and so the Grossberg layer is needed to get the desired mapping. The Grossberg layer is trained via a supervised learning rule and the whole network is said

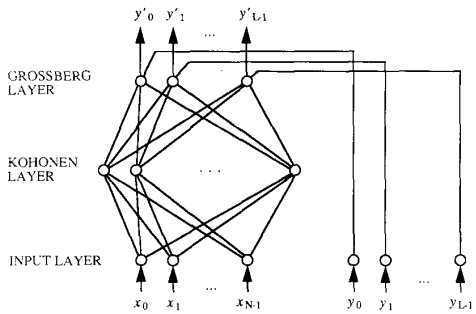


Fig. 3. Forward-only counterpropagation network, inputs are $x_0, x_1, \cdots, x_{N-1}$, outputs $y'_0, y'_1, \cdots, y'_{L-1}$, and desired outputs $y_0, y_1, \cdots, y_{L-1}$.

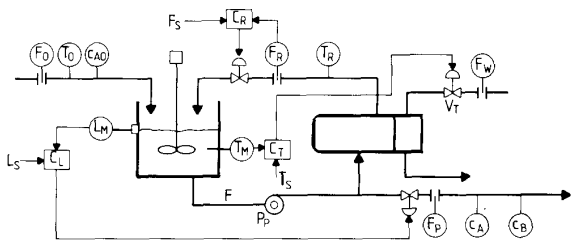


Fig. 4. The simulated process.

to be supervised [8], [9]. The outputs of the network are $y'_0, y'_1, \cdots, y'_{L-1}$ and the desired outputs $y_0, y_1, \cdots, y_{L-1}$.

Many times the output layer of the Kohonen self-organizing network is two-dimensional (2-D) and the neighborhood set N_C is composed of those elements that lie within a certain radius from a certain element C [17], [18]. The weights of the Kohonen network are updated so that the network learns a vector quantization of the input space, and thus a classification of all its vectors [18]. If we denote the input vector by $\boldsymbol{x} = [x_0 x_1 \cdots x_{N-1}]^T$ and the weight vector of the Kohonen layer element i by $\boldsymbol{m}_i = [m_{i0} m_{i1} \cdots m_{iN-1}]^T$, we can write the shortcut algorithm that describes the self-organizing operation [17], [18]:

In the beginning the weights are initialized to random values.

For $t = 0, 1, 2, \cdots$ compute

1) the best matching Kohonen element C

$$\|\boldsymbol{x}(t) - \boldsymbol{m}_C(t)\| = \min_i \{\|\boldsymbol{x}(t) - \boldsymbol{m}_i(t)\|\} \tag{4}$$

2) and the new weight vectors

$$\boldsymbol{m}_i(t+1) = \begin{cases} \boldsymbol{m}_i(t) + \alpha(t)(\boldsymbol{x}(t) - \boldsymbol{m}_i(t)) & \text{for } i \in N_C \\ \boldsymbol{m}_i(t) & \text{for all other indices } i \end{cases} \tag{5}$$

where $\alpha(t)$ is monotonically decreasing function of time.

First the best matching element C of the Kohonen layer is selected. A cluster in the neighborhood of the element C is then represented by a neighborhood set N_C of elements and the weight vectors are updated. Usually function $\alpha(t)$ decreases from close to unity towards zero. Empirically it has been found that the radius of the neighborhood should decrease in time too [17], [18].

The output of the Grossberg layer is the weighted sum of the Kohonen layer outputs. The weights of the Grossberg layer are adjusted so that the desired classification is achieved.

Counterpropagation networks can be used to implement approximations of mappings in the same way as perceptrons. Counterpropagation networks for pattern classification problems are usually quite large because an appropriate accuracy requires very many elements in the Kohonen layer. On the other hand the learning algorithms are simple and fast. One advantage is that the algorithms are easy to implement.

IV. Simulation Example

A. Example Process [24]

The simulated process in shown in Fig. 4. The process consists of a heat exchanger and a continuous stirred tank reactor where an irreversible first-order reaction $A \rightarrow B$ takes place. The reaction is catalytic and exothermic. The temperature of the reactor is controlled by pumping a part of the reactor outlet stream back to the reactor through the heat exchanger where the recycle flow is cooled by an external flow. The process has three feedback control loops where discrete PI-controllers are used. The controllers keep the recycle flow rate, the level of the reactor and the temperature of the reactor constant. The process is chosen because it displays the most common characteristics appearing in industrial processes. The model of the process is given in Appendix A.

All together 14 variables are measured from the process, encircled variables (Fig. 4) and the outputs of the controllers. In simulations noise is added to measurements. The simulated noise varied from 0% to 10% of measurement regions. Ten fault situations are studied and in every simulated fault situation all 14 measurements are saved for training and testing of neural networks.

Ten representative fault situations are selected for study: 1) Input pipe partially blocked, 2) Recycle pipe partially blocked, 3) Input concentration of A high, 4) Recycle flow set point high, 5) Fouled heat exchanger, 6) Deactivated catalyst, 7) Temperature control valve stuck high, 8) Leak flow in reactor, 9) Recycle flowmeter stuck high, 10) Malfunction in pump. The computer simulations are carried out with an IBM PS/2 Model 70. The simulator is coded in C and it uses a fourth-order Runge–Kutta algorithm with adaptive stepsize control to solve the differential equations [27]. Because the simulator has to solve both the static nonlinear equations and the nonlinear differential equations and because a flexible simulation environment is needed, the simulator is coded in C and no commercial simulator is used. The measurements are taken mainly as steady state, but in some situations changes are so slow that the measurements are taken before the process reaches the steady state. All measurements are scaled to a continuous range from -1 to $+1$. The scaling makes the classification easier to learn, because the original measurement data contains both small and large values. For example the

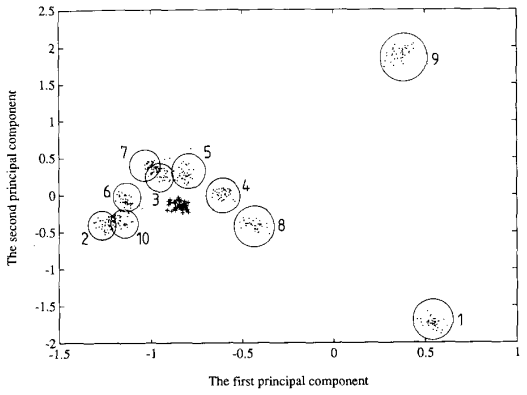


Fig. 5. The simulated data presented by the two first principal components. Normal operation data are denoted by crosses (+) and fault situation data by points (.) and different faults are labelled with numbers.

level of the reactor is about 3 m and the input concentration of A 1200 mol/m^3. Without the scaling the fluctuations in large measurement values would dominate the operation of neural networks.

The number of measurements is so large that the visualization of the situation requires the reduction of the measurement dimension. In the following, this is done by principal component analysis [2]. The sample estimate $\boldsymbol{S}$ of the covariance matrix of the measurements $\boldsymbol{x} = [x_0 x_1 \cdots x_{N-1}]^T$ is calculated using the formula

$$\boldsymbol{S} = \frac{1}{p-1} \sum_{i=1}^{p} (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T \tag{6}$$

where p is the number of observations (measurement patterns), $\boldsymbol{x}_i$ is the ith observation (measurement pattern), and $\overline{\boldsymbol{x}}$ is the mean vector of measurements. The first principal component is $\nu_1 = \boldsymbol{a}_1^T \boldsymbol{x}$ where $\boldsymbol{a}_1$ is the eigenvector of $\boldsymbol{S}$ corresponding the largest eigenvalue of $\boldsymbol{S}$ and the second principal component is $\nu_2 = \boldsymbol{a}_2^T x$ where $\boldsymbol{a}_2$ is the eigenvector corresponding the second largest eigenvalue of $\boldsymbol{S}$. The eigenvalue is the variance of the particular principal component.

The training data used in the following simulations contain 440 measurement patterns. The fault diagnosis problem can be visualized by the first and second principal component, which together can explain about 65% of the total variance of the measurement data. The data is presented by its first and second principal component in Fig. 5. It should be emphasized that a fourteen dimensional space has been projected into a 2-D space. The measurement data concerning the normal operation are denoted by crosses (+) and the data concerning the fault situations are denoted by points (.). There are some clusters that are badly overlapping and some that are far from other clusters in this 2-D case.

B. Fault Diagnosis with Single-Layer Perceptron

The simulated single-layer perceptron network is shown in Fig. 6, which is a copy of the screen of the computer. The program used is NeuralWorks Professional II by NeuralWare

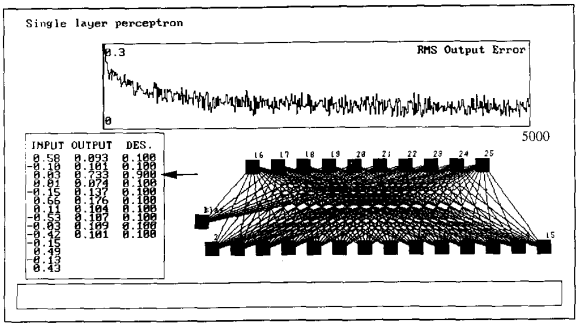


Fig. 6. Simulated single-layer perceptron: the architecture (below right), RMS output error during training from 0 to 5000 cycles (above), and an example of a fault diagnosis (below left).

Inc. [36]. Besides this, Matlab by MathWorks Inc. [37] is used to simulate some properties that are not included in NeuralWorks Professional II. The neural network simulations are also carried out in IBM PS/2.

The network has 14 input nodes—one for every measurement value—and 10 output nodes—one for every examined fault situation. The output nodes have standard sigmoids as activation functions:

$$\text{Sigmoid}(s) = \frac{1}{1+e^{-s}}. \tag{7}$$

The training data contains 440 measurement patterns, forty patterns for the normal operation and for every fault situation, and the desired outputs of the network. In the normal operation the network should produce all outputs near zero. Fault number one should produce one in the first output and the other outputs a value near zero, etc. Instead of using 0 and 1 as desired outputs, the values 0.1 and 0.9 are chosen because they do not demand infinitely large weight values when the sigmoid functions are used in the output nodes. In the testing phase the measurement pattern is interpreted as a normal operation data if all network outputs are smaller than 0.5 and a fault is reported if an output is larger than 0.5.

The network shown in Fig. 6 has been trained 5000 times. Every time an input pattern is picked randomly from the training data. In the beginning the weights have small random values from -0.1 to $+0.1$. The root mean square (RMS) error between the actual output and the desired output is shown in the box above the network in Fig. 6:

$$\text{RMS} = \sum_{k=0}^{L-1} E_k^2 = \sum_{k=0}^{L-1} \frac{1}{2}(d_k - y_k)^2. \tag{8}$$

Actually the box presents the average values of every ten RMS errors. The error saturates to a constant level but a lot of fluctuation remains. This indicates that the classification of the network is not very accurate.

The network can classify most situations without any problem but some measurements produce outputs that are far from the desired values. In any case the network can classify the whole training data. In every fault situation only one output is over 0.5 and it is the desired one. When an input pattern

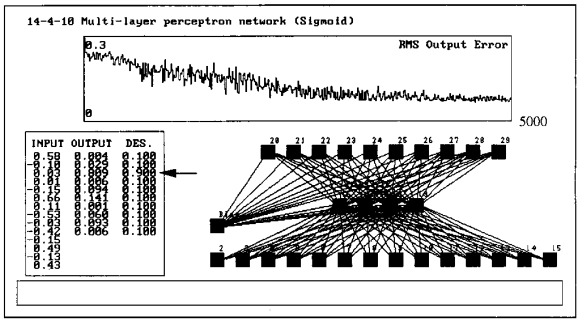


Fig. 7. Simulated two-layer perceptron with sigmoid nonlinearities: the architecture (below right), RMS output error during training from 0 to 5000 cycles (above), and an example of a fault diagnosis (below left). Fault number 3 has occurred as indicated in the third row.

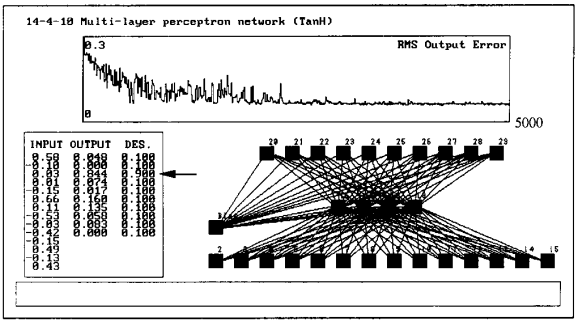


Fig. 8. Simulated two-layer perceptron with sigmoid nonlinearities in the output layer and hyperbolic tangents in the hidden layer: the architecture (below right), RMS output error (above), and an example of a fault diagnosis (below left).

that is not in the training data is used in testing the trained network (recall), the operation is worse. The classification fails sometimes but still the successful classifications are in majority.

An example of the input, output and desired output of the network is shown in the box left to the network in Fig. 6. This measurement pattern is not included in the training data. Fault number 3 has occurred shown by the third row. The network indicates this by showing 0.733 in the third output (0.9 is the desired value) and about 0.1 for the other outputs. The calculations are carried out with double precision numbers, but only the first decimals are shown in the box.

C. Fault Diagnosis with Multilayer Perceptron

The multilayer perceptron network used in fault detection and diagnosis is shown in Fig. 7. The input layer contains 14, the hidden layer 4 and the output layer 10 nodes. The classification requires at least four elements in the hidden layer [21]. Sigmoid functions are used in the hidden and output nodes. The training data used is the same as earlier and the network in Fig. 7 has also been trained 5000 times. The initial weight values are small random numbers $[-0.1, 0.1]$ and the learning coefficients of the back-propagation algorithm are selected so that the gain term is $\eta = 0.95$ and the momentum term $\alpha = 0.60$. Other values of the coefficients have also been simulated and these values are selected because they yield fast learning.

The trained network display smaller fluctuations in the error than in Fig. 6 where the single-layer perceptron was used. The multilayer perceptron can classify the training data well. The patterns that are not in the training data are also classified correctly indicating robustness. An example of diagnosis of the fault number 3 is shown in Fig. 7.

The training cannot be speeded up significantly by changing the learning parameters but using hyperbolic tangent as the nonlinearity in the hidden layer changes the situation dramatically. This can be seen from Fig. 8 that shows a network containing hyperbolic tangents in the hidden nodes. The network is otherwise the same as the network in Fig. 7, even the initial values of the weights are the same. There is no difference in the accuracy of the classification with these two networks, but the latter learns the mapping much faster than the former.

The addition of more hidden nodes improves the rate of training somewhat but the ability of the network to classify patterns that are not included in the training data suffers. The ability of generalization gets worse. Too many hidden nodes leave too much freedom for the weights to adapt during the training. The number of the hidden nodes must be large enough to form a decision region that is as complex as is required by the given problem, and on the other hand so small that the generalization ability remains good.

The second hidden layer makes the training slower. For example the 14-6-6-10 perceptron network with sigmoid functions in both hidden layers requires over 8000 training cycles. The simulations become much slower because more layers require more weights. Although the 14-6-6-10 perceptron network learns to classify the training data accurately, it fails to classify new data.

All measurements are scaled to the symmetrical range of -1 to $+1$. A nonsymmetrical scaling from 0 to 1 has also been used but it produces slower training.

D. Fault Diagnosis with Counterpropagation Network

The simulated counterpropagation network for fault detection and diagnosis is shown in Fig. 9. The network has 14 input nodes (bottom) and above these a normalizing layer with 15 nodes. This layer normalizes all inputs so that every input vector has the same length or

$$x_0^2 + x_1^2 + \cdots + x_{N-1}^2 = 1. \tag{9}$$

The normalizing is not mandatory but it makes the training phase faster [33]. All input vectors have then the same length in the measurement space and have the same possibility to be selected as the best matching element C in the Kohonen layer.

The Kohonen layer requires many elements for good performance. The same classes are often mapped into several Kohonen elements and many elements remain unused [18]. In Fig. 9 there are 18 elements in the Kohonen layer. If there are less than 18 elements, the classification does not produce the desired classes but some measurement data of different

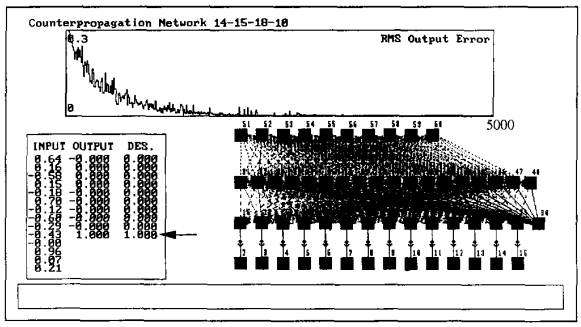


Fig. 9. The counterpropagation network (successful training): the architecture (below right), RMS output error during training from 0 to 5000 cycles (above), and an example of a fault diagnosis (below left).

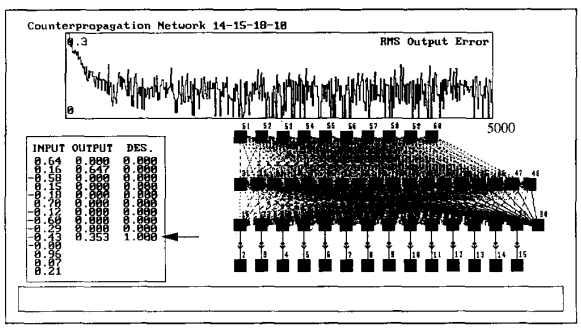


Fig. 10. The counterpropagation network (unsuccessful training): the architecture (below right), RMS output error during training from 0 to 5000 cycles (above), and an example of a fault diagnosis (below left).

fault situations are classified into the same classes. When the number of Kohonen elements increases, the probability of successful classifications increases too. An example of a successful classification is presented in Fig. 9 and an example of an unsuccessful classification in Fig. 10. Further training does not make the network in Fig. 10 to learn the classification. In both figures the network architecture, the RMS output error, and an example of fault diagnosis is shown. In Fig. 10 the Grossberg layer is not able to distinguish every fault situation, because the Kohonen layer classifies the measurements concerning the faults number two and ten into the same class. So the whole network fails to classify the faults number two and ten. Other situations are classified correctly. The desired outputs are now 0 and 1 (see Fig. 9 and 10) because the output of the Kohonen layer can have only values 0 and 1 in the neural network simulator.

The number of successful trainings is about identical to the number of unsuccessful trainings when the Kohonen layer has 18 elements. The portion of successful trainings increases when more elements are used in the Kohonen layer and when the Kohonen layer has 33 elements—three times the number of classes—the portion of unsuccessful trainings is very small but still sometimes the faults number two and ten are mapped into the same class.

The gain function $\alpha(t)$ is usually selected so that in the beginning, when the ordering is still going on, it has a fairly big value (1.0 . . . 0.2) and then during the fine adjustment, a relatively small value (0.10 . . . 0.01) [18]. In the simulations described above, the gain function is 0.5 during the training cycles from 1 to 300, 0.25 from 301 to 600, 0.15 from 601 to 1000, 0.1 from 1001 to 2000, and 0.07 from 2001 to 5000. The initial weights are usually selected randomly [17], [18] and can be normalized in the same way as the inputs [33]. In these simulations the initial weights are selected randomly $[-0.1, 0.1]$. If they have much larger initial values (for example $[-0.5, 0.5]$) the number of successful trainings decreases.

In the above simulations the neighborhood has not been used. In a fault diagnosis problem the goal is to classify the data correctly, the organization of the network is not so important. However we have also done simulations with the neighborhood. These simulations are carried out by Matlab. These networks have the same problem as the previous ones, the classification of the faults number two and ten. The data concerning these faults match the same Kohonen layer elements. Other situations are classified without any problem.

E. More Simulations with Multilayer Perceptron

According to the simulations performed, the multilayer perceptron network is the most suitable architecture for fault diagnosis. This result was also observed when the example process of Hoskins and Himmelblau [12]—the system of three continuous-stirred-tank reactors in series—was studied. The multilayer perceptron network is able to form decision regions that are complex enough for practical problems. Single-layer perceptrons are limited to linearly separable decision regions and their generalization ability is not as good as with multilayer perceptrons. Counterpropagation networks offer an interesting alternative for perceptrons but they are unreliable. Because the 14-4-10 multilayer perceptron network with hyperbolic tangents in the hidden nodes and sigmoids in the output nodes (cf. Fig. 8) works best in this example problem, let's examine it a little closer.

The training data are taken at a steady state. An interesting question is, what would the network produce in a transient situation. If a fault happens suddenly it is desirable that the fault will be detected as early as possible. Fig. 11 visualizes the transitions from the normal operation by the means of principal components. The faults number 1, 6, 8, and 9 are shown. The faults are first simulated one by one and the measurement data are saved every second. Then the data are projected into 2-D space (cf. Fig. 5). Fault number 6 has been simulated for 250 seconds and the others for 100 seconds. During these times the new steady state operation points are almost reached. The effect of measurement noise can clearly be seen in Fig. 11. The faults number 1 and 9 are very far from the normal operation as can already be seen in Fig. 5 and the transition is very straightforward. In both cases already the first measurement (one second after the fault has occurred) is clearly separate from the normal operation. Fault number 8 acts very much like the previous ones. Fault number 6 does not move the process straight to the new steady state but makes a curve.

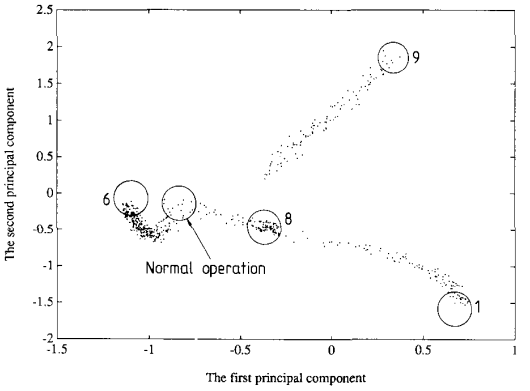


Fig. 11. The simulated transient data (faults number 1, 6, 8, and 9) presented by the two first principal components.

TABLE I

Time (s)	8. Output	Other Outputs
1	0.074	0.006 . . . 0.078
10	0.661	0.001 . . . 0.295
20	0.858	0.000 . . . 0.126
30	0.888	0.000 . . . 0.119
40	0.905	0.000 . . . 0.127
50	0.906	0.000 . . . 0.157
60	0.898	0.000 . . . 0.141

In the other fault situations the operation point moves quickly from the normal operation area but the area concerning the steady state measurements of a particular fault situation is not reached at once. During the transitions the measurement data presented by the first and second principal component are very badly overlapping. If these situations are plotted like the previous ones in Fig. 11 the result is so messy that one can hardly see anything at all.

The ability of the 14-4-10 perceptron to detect faults already in a transient situation is studied in Tables I and II. In both cases the network is trained with the steady state data (Fig. 8). First fault number 8 is examined because it seems straightforward but not as obvious as faults number 1 and 9. The time history of the outputs of the network is shown in Table I. The time in seconds after the occurrence of the fault is listed in the first column, the value of the eighth output in the second column (should be 0.9 after the fault occurs), and the range of the other outputs in the third column. The network very quickly recognizes that something is wrong in the process and after about 30 s detects that fault number 8 occurs. The process reaches the new steady state in about 300 s.

In Table II fault number 10 is examined. The time after the occurrence of the fault is again listed in the first column, then the values of the second, fifth, seventh and tenth output are listed, and finally the range of the other outputs is shown. The network again quickly detects that something is fouled up but has problems in diagnosis. In the beginning the network classifies the situation as faults number 7 and 2. About 60 s after the occurrence of the fault the network classifies the situation correctly as fault number 10. The new steady state is reached in about 300 s but after about 60 s, the operation point is near the new steady state.

TABLE II

Time (s)	2. Output	5. Output	7. Output	10. Output	Other Outputs
1	0.595	0.260	0.708	0.002	0.001 . . . 0.108
10	0.754	0.278	0.582	0.005	0.001 . . . 0.113
20	0.817	0.258	0.470	0.008	0.001 . . . 0.115
30	0.810	0.177	0.335	0.015	0.001 . . . 0.109
40	0.675	0.154	0.173	0.046	0.001 . . . 0.059
50	0.358	0.224	0.033	0.329	0.001 . . . 0.033
60	0.099	0.209	0.026	0.415	0.002 . . . 0.110
70	0.120	0.297	0.014	0.614	0.001 . . . 0.097
80	0.110	0.268	0.011	0.674	0.001 . . . 0.100
90	0.134	0.135	0.008	0.744	0.001 . . . 0.090

With faults number 1 and 9 the situation in the transient phase is similar to fault number 8 (Table I). The network classifies the measurement data correctly almost immediately when the fault occurs. In other cases (faults number 2, 3, 4, 5, 6, and 7) the situation is similar to fault number 10 (Table II). Now the network classifies the data correctly only near the new steady state. The difficulties of the classification can be seen in Fig. 5 where the data clusters of these cases are very close to each other.

V. Conclusion

Different neural network architectures for fault diagnosis have been investigated. The best one turned out to be the multilayer perceptron network. Using a hyperbolic tangent as the nonlinear element in the network improved the training rate of the network considerably.

A realistic heat exchanger–continuous stirred tank reactor system is used as a test process. The process has 14 noisy measurements and 10 typical faults. The fault detection and diagnosis problem is visualized by principal component analysis. The multilayer perceptron network is able to correctly identify the faults in all cases. An alternative, counterpropagation network seemed to learn the faults quicker, but in some cases is not able to do it at all. This, of course, for a fault diagnosis system is unacceptable.

The results in literature and in this paper must be considered preliminary, although quite promising. The next step is to experiment with real processes to gain more experience. Laboratory processes offer a properly limited environment before the complicated industrial plants are studied.

Fault diagnosis is mostly performed for steady-state data. Only some experiments were carried out for transient data. Teaching of neural network in general seems fairly slow. Also, if a new fault is introduced, teaching should start all over again. All this requires more research.

Appendix

The example process consists of a heat exchanger and a reactor (see Fig. 4). The symbols used are (numerical values in brackets):

A	surface area in the heat exchanger (40 m^2)
A_M	area of the bottom of the tank (3.0 m^2)
A_P, B_P	pump parameters (−4200 Pa s/kg, 220000 Pa)
C_1, C_2	heat capacities (100, 200 kJ/°C
c	specific heat capacity of the fluid in the tank (4.2 kJ/kg°C)
c_A	concentration of A (20 mol/m^3)
c_{A0}	input concentration of A (1200 mol/m^3)
c_B	concentration of B (1180 mol/m^3)
E	activation energy (100 kJ/mol)
F	reactor outlet flow rate (kg/s)
F_{leak}	leak flow rate (0 kg/s)
F_P	output flow rate (2.5 kg/s)
F_R	recycle flow rate (7.0 kg/s)
F_S	set point of the recycle flow rate (7.0 kg/s)
F_W	flow rate of the coolant fluid (3.5 kg/s)
F_0	input flow rate (2.5 kg/s)
g	acceleration of gravity (9.81 m/s^2)
ΔH	heat of the reaction (300 kJ/mol)
K_L, K_{L1}, K_{L2}	fluid resistance coefficients (0, 1600, 940 Pa/(kg/s)^2)
K_{V1}, K_{V2}	parameters of the control valves (6.7 10^{-4}, 6.7 10^{-4} m^3/kg)
k_R	reaction time (1/s)
k_0	preexponential kinetic constant (0.02 1/s)
L_M	level of the fluid in the tank (3.0 m)
L_S	set point of the level (3.0 m)
m	mass of the fluid in the tank (kg)
p	pressure in the joint of the outlet and recycle flows (Pa)
p_0	normal air pressure (10^5 Pa)
$\Delta p_L, \Delta p_{L1}, \Delta p_{L2}$	pressure drops in the pipes (Pa)
Δp_P	pressure difference of the pump (Pa)
$\Delta p_{V1}, \Delta p_{V2}$	pressure drops in the control valves (Pa)
Q	amount of the heat transferred in the heat exchanger (kW)
Q_R	amount of the heat created by the reaction (kW)
R	gas constant (8.3143 kJ/mol K)
T_{in}	input temperature of the coolant fluid (15 °C)
T_M	reactor temperature (58 °C)
T_{out}	output temperature of the coolant fluid (56 °C)
T_R	recycle temperature (52 °C)
T_S	set point of the reactor temperature (58 °C)
T_0	input temperature (30 °C)
ΔT	average effective temperature difference (°C)
U	heat-transfer coefficient (10 kW/°C m^2)
ρ	density of the fluid in the tank (1000 kg/m^3)

A. The Heat Exchanger

The heat exchanger of the process is of a counterflow type. It is assumed that U is constant and the system is adiabatic. The fundamental heat transfer relation is

$$Q = UA\Delta T \tag{10}$$

where ΔT is given by the approximate equation

$$\Delta T = \sqrt{(T_M - T_{\text{out}})(T_R - T_{\text{in}})}. \tag{11}$$

The balance equation for the energy of the recycle flow is

$$\frac{d}{dt}(C_1 T_R) = cF_R T_M - (cF_R T_R + UA\Delta T) \tag{12}$$

and for the energy of the coolant fluid

$$\frac{d}{dt}(C_2 T_{\text{out}}) = UA\Delta T - cF_W(T_{\text{out}} - T_{\text{in}}). \tag{13}$$

After eliminating ΔT

$$\dot{T}_R = \frac{1}{C_1}\left(cF_R(T_M - T_R) - UA\sqrt{(T_M - T_{\text{out}})(T_R - T_{\text{in}})}\right) \tag{14}$$

and

$$\dot{T}_{\text{out}} = \frac{1}{C_2}\left(UA\sqrt{(T_M - T_{\text{out}})(T_R - T_{\text{in}})} - cF_W(T_{\text{out}} - T_{\text{in}})\right). \tag{15}$$

B. The Reactor

The balance equation for the mass of the fluid in the reactor is

$$\frac{d}{dt}(\rho L_M A_M) = F_0 + F_R - F - F_{\text{leak}}, \tag{16}$$

which becomes

$$\dot{L}_M = \frac{1}{\rho A_M}(F_0 + F_P - F_{\text{leak}}). \tag{17}$$

The balance equation for the energy in the reactor is

$$\frac{d}{dt}(cmT_M) = cF_0 T_0 + cF_R T_R + Q_R - cFT_M - cF_{\text{leak}} T_M. \tag{18}$$

The energy created in the exothermic reaction is

$$Q_R = \Delta H c_A L_M A_M k_R \tag{19}$$

with

$$k_R = k_0 e^{-E/RT_M}. \tag{20}$$

When (19) and (20) are substituted in (18) the balance equation yields

$$\dot{T}_M = \frac{1}{c\rho A_M L_M}\left(cF_0 T_0 + cF_R T_R + \Delta H c_A L_M A_M k_0 e^{-E/RT_M} - cT_M F_0 - cT_M F_R\right). \tag{21}$$

The balance of component A is

$$\frac{d}{dt}(A_M L_M c_A) = \frac{1}{\rho}(F_0 c_{A0} + F_R c_A - F c_A - F_{\text{leak}} c_A) - k_R c_A A_M L_M \quad (22)$$

and of component B respectively

$$\frac{d}{dt}(A_M L_M c_B) = \frac{1}{\rho}(F_R c_B - F c_B - F_{\text{leak}} c_B) + k_R c_A A_M L_M. \quad (23)$$

After calculating derivative and substituting (17) and (20), the balance equations (22)–(23) become

$$\dot{c}_A = \frac{1}{A_m L_M}\left(\frac{F_0}{\rho}(c_{A0} - c_A) - k_0 e^{-E/RT_M} c_A A_M L_M\right) \quad (24)$$

$$\dot{c}_B = \frac{1}{A_M L_M}\left(-\frac{F_0}{\rho} c_B + k_0 e^{-E/RT_M} c_A A_M L_M\right). \quad (25)$$

C. The Pump and Pipes

It is assumed that the characteristic equation of the pump is linear

$$\Delta p_P = A_P F + B_P. \quad (26)$$

The fluid resistances of the pipes cause the pressure drops

$$\Delta p_L = K_L F^2 \quad (27)$$

$$\Delta p_{L1} = K_{L1} F_R^2 \quad (28)$$

$$\Delta p_{L2} = K_{L2} F_P^2. \quad (29)$$

The pressure drops in the control valves are

$$\Delta p_{V1} = K_{V1}\left(\frac{F_R}{a_1 A_{\max}}\right)^2 \quad (30)$$

$$\Delta p_{V2} = K_{V2}\left(\frac{F_P}{a_2 A_{\max}}\right)^2. \quad (31)$$

Because the pipe system dynamics is very fast compared with the dynamics of the reactor and the heat exchanger, the static model for flow rates is sufficient

$$\Delta p_L = p_0 + \rho g L_M - p + \Delta p_P \quad (32)$$

$$\Delta p_{V1} + \Delta p_{L1} = p - p_0 \quad (33)$$

$$\Delta p_{V2} + \Delta p_{L2} = p - p_0 \quad (34)$$

From (26)–(34) the flow rates F_P and F_R can be calculated, analytically or numerically.

D. The Controllers

Discrete PI-controllers are used to keep the recycle flow rate, the level of the reactor and the temperature of the reactor constant. The sampling time is 0.1 s.

REFERENCES

[1] M. Basseville, "Detecting changes in signals and systems—A survey," *Automatica,* vol. 24, pp. 309–326, 1988.

[2] M. L. Berenson, D. M. Levine, and M. Goldstein, *Intermediate Statistical Methods and Applications—A Computer Package Approach.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

[3] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems,* vol. 2, pp. 303–314, 1989.

[4] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[5] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—A survey and some new results," *Automatica,* vol. 26, pp. 459–474, 1990.

[6] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks,* vol. 2, pp. 183–192, 1989.

[7] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks,* vol. 1, pp. 75–89, 1988.

[8] R. Hecht-Nielsen, "Applications of counterpropagation networks," *Neural Networks,* vol. 1, pp. 131–139, 1988.

[9] ——, "Counterpropagation networks," *Applied Optics,* vol. 26, pp. 4979–4984, 1987.

[10] D. M. Himmelblau, *Fault Detection and Diagnosis in Chemical and Petrochemical Processes.* Amsterdam: Elsevier, 1978.

[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, pp. 359–366, 1989.

[12] J. C. Hoskins and D. M. Himmelblau, "Artificial neural network models of knowledge representation in chemical engineering," *Comput. Chem. Eng.,* vol. 12, pp. 881–890, 1988.

[13] R. Isermann, "Process fault detection based on modeling and estimation methods–A survey," *Automatica,* vol. 20, pp. 387–404, 1984.

[14] W. P. Jones and J. Hoskins, "Back-propagation," *BYTE,* vol. 12, pp. 155–162, Oct. 1987.

[15] T. Kohonen, "Adaptive, associative, and self-organizing functions in neural computing," *Applied Optics,* vol. 26, pp. 4910–4918, 1987.

[16] ——, "An introduction to neural computing," *Neural Networks,* vol. 1, pp. 3–16, 1988.

[17] ——, *Self-Organization and Associative Memory.* Berlin: Springer-Verlag, 1984.

[18] ——, "The 'neural' phonetic typewriter," *IEEE Computer,* vol. 21, pp. 11–22, 1988.

[19] M. A. Kramer and B. L. Palowitch Jr., "Expert system and knowledge-based approaches to process malfunction diagnosis," presented at the AIChE Nat. Meet., Chicago, 1985.

[20] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.,* vol. 4, pp. 4–22, April 1987.

[21] G. Mirchandani and W. Cao, "On hidden nodes for neural nets," *IEEE Trans. Circuits Syst.,* vol. 36, pp. 661–664, 1989.

[22] S. R. Naidu, E. Zafiriou, and T. J. McAvoy, "Use of neural networks for sensor failure detection in a control system," *IEEE Contr. Syst. Mag.,* vol. 10, pp. 49–55, April 1990.

[23] N. J. Nilsson, *Learning Machines.* New York: McGraw-Hill, 1965.

[24] O. O. Oyeleye and M. A. Kramer, "Qualitative simulation of chemical process systems: steady-state analysis," *AIChE J.,* vol. 34, pp. 1441–1454, 1988.

[25] Y. Pao, *Adaptive Pattern Recognition and Neural Networks.* New York: Addison-Wesley, Inc., 1989.

[26] L. F. Pau, *Failure Diagnosis and Performance Monitoring.* New York: Marcel Dekker, 1981.

[27] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes.* New York: Cambridge Univ. Press, 1986, pp. 547–577.

[28] S. H. Rich and V. Venkatasubramanian, "Model-based reasoning in diagnostic expert systems for chemical process plants," *Comput. Chem. Engng.,* vol. 11, pp. 111–122, 1987.

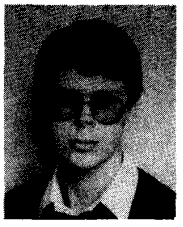
[29] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations.* Cambridge, MA: MIT Press, 1986.

[30] ——, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models.* Cambridge, MA: MIT Press, 1986.

[31] G. Stephanopoulos, *Chemical Process Control.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc. 1984.

[32] V. Venkatasubramanian and K. Chan, "A neural network methodology for process fault diagnosis," *AIChE J.,* vol. 35, pp. 1993–2002, 1989.

[33] P. D. Wasserman, *Neural Computing: Theory and Practice.* New York: Van Nostrand Reinhold, 1989.
[34] K. Watanabe, I. Matsuura, M. Abe, M. Kubota, and D. M. Himmelblau, "Incipient fault diagnosis of chemical processes via artificial neural networks," *AIChE J.*, vol. 35, pp. 1803–1812, 1989.
[35] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica,* vol. 12, pp. 601–611, 1976.
[36] *NeuralWorks Professional II: User's Guide,* NeuralWare, Inc., Pittsburgh, PA, 1989.
[37] *Matlab™ for 80386-based MS-DOS Personal Computers: User's Guide,* The MathWorks, Inc., S. Natick, MA, 1989.



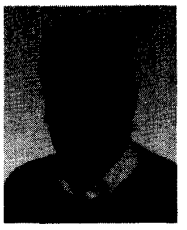
Timo Sorsa was born in Luumäki, Finland. He received the M.Sc. degree in electrical engineering from Tampere University of Technology, Tampere, Finland, in 1990.

He is a Research Engineer in the Control Engineering Laboratory at the Tampere University of Technology. His research interests are fault diagnosis and applications of neural networks in process engineering.

Heikki N. Koivo (S'67–M'71–SM'86) was born in Vaasa, Finland. He received the B.S.E.E. degree in 1967 from Purdue University, West Lafayette, IN, and the M.S. degree in electrical engineering and the Ph.D. degree in control sciences from the University of Minnesota, Minneapolis, MN, in 1969 and 1971, respectively.

In 1971 he joined the Department of Electrical Engineering of the University of Toronto, ON, Canada, as a Visiting Assistant Professor, and served there as an Assistant Professor from 1972 to 1975. From 1975 to 1979 he was an Associate Professor in applied mathematics, and from 1979 to 1984 in control engineering at Tampere University of Technology, Tampere, Finland. In 1986 he served as the Chairman of the Department of Electrical Engineering. Currently he is a Professor in automation technology. He has been awarded The Finnish Academy's Senior Research Fellowship for 1990–1991. He has authored more than 100 publications in the areas of adaptive and learning control, robotics and computer control.

Dr. Koivo is a member of the Editorial Board of *Journal of Adaptive Control and Signal Processing* for which he has just edited a special issue on expert systems in control. He is also a member of the IFAC Theory Committee and The Finnish Academy of Technical Sciences.



Hannu Koivisto was born in Nakkila, Finland. He received his M.Sc. degree in electrical engineering in 1978 from Tampere University of Technology, Finland. He then served as a Research Engineer in Tampere University of Technology from 1980 to 1985 (HVAC-control systems). He received his Licentiate of Technology degree in 1985 in control engineering from Tampere University of Technology, Finland. He is currently a Junior Research Fellow of the Finnish Academy working towards the Ph.D. degree. His research interests are adaptive and neural network control and their applications.