International Conference on Artificial Intelligence Applications in Engineering and Technology ICAIET 2002, Kota Kinabalu, 17-18 June 2002)

Detection of Sensor Failure in a Palm Oil Fractionation Plant Using Artificial Neural Network

Arshad Ahmad[1] Mohd. Kamaruddin Abd. Hamid[2]

[1]*Department of Chemical Engineering*
Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia
Tel: +60-7-550-5410, Fax: +60-7-558-1463, Email: dr_aa@fkkksa.utm.my

[2]*Laboratory of Process Control, Department of Chemical Engineering*
Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia
Tel: +60-7-558-6229, Email: kamadean@dacafe.com

Abstract

Artificial neural network by virtue of its pattern recognition capabilities has been explored to systematically detect failures in process plants. In this paper, a two-stage approach integrating a neural network dynamic estimator and a neural network fault classifier is proposed to overcome the problem of malfunction in sensors. The process estimator was designed to predict the dynamic behavior of the normal or unfaulty operating process even in the presence of sensor failures. As such, any variables that are related or influenced by the failures under investigation cannot be used as model inputs. The difference between this estimated "normal" and the actual process measurements, termed the residuals are fed to the classifier for fault detection purposes. The classifier that was founded on feedforward network architecture then identifies the source of faults. The estimator was constructed using externally recurrent network where the estimated values are fed back to the input neurons as delayed signals. The scheme was implemented to detect sensor failure in a palm oil fractionation process. To generate the required simulation data, HYSYS.Plant dynamic process simulator was employed. The proposed scheme was successful in detecting pressure and temperature sensor failures introduced within the system.

Keywords:

Artificial Neural Network, Estimator, Fault Classifier, Palm Oil Fractionation Process

Introduction

Over the past years, the oleochemical industry has made a concerted effort to streamline operations. Their goal was simply to produce products as many as possible. Nowadays, as the market is highly competitive worldwide, production efficiency and product consistency become essential to success. Implemented of advance controllers in oleochemical processes have allowed plants to run at more economically attractive operating points, to produce more consistent products at higher capacities without risking long-term shutdowns or accidents. On the other hand, they have also created havoc in maintenance and performance degradation. There are lots of reasons why performance of advance controller may degrade. They range from instrumentation and equipment problems, including sensor and actuator faults, to process parameter changes (due to operating point changes, large disturbances, operators re-tuning inner loops, etc.) and changes in the process/feed-stock disturbances characteristics. Therefore, effective maintenance starts with an early detection of developing problems. With an early detection, the engineer has a change to stop the problem before it grows into a full malady.

Several techniques have been developed for monitoring and detection. These techniques can be broadly classified into three categories: model-based techniques, expert system and pattern recognition [1-2]. In the model-based approach, the actual behavior of the process to be supervised is compared with that of a nominal model driven by the same inputs. Faults can be detected by evaluating the difference between the estimated value of the model and the actual values of process variables. The expert system, also called a knowledge-based system, is built upon some given facts and relations so as to make an induction for system behaviors. However, it is very hard to develop an expert system for many sophisticated chemical processes, if the fault related to knowledge is not available or clear enough. Pattern recognition is based on the design of math-model free fault detection and diagnosis. From the concept of pattern mapping, the measurements and the identified fault model for each abnormal operation are needed in order to connect between patterns. Each pattern consists of measurement and corresponding fault models. The memories of the fault status are usually established via supervised or unsupervised training. When the patterns are established

by a pattern mapping, any operating condition is assigned to a class or a label from a set of fault pattern into identifiable classes based on certain similar features.

For nonlinear systems like chemical/oleochemical, building the process model is extremely difficult in general. During the past few years, artificial neural networks (ANN) were used to model non-linear processes. Based on measurement of the process, a suitable ANN can be trained to adapt the process behavior. Since ANN requires little or no prior knowledge of systems, it provides an effective tool for dealing with non-linearity because of its well-known approximation ability. This feature is particularly attracted to the faults detection scheme due to the non-linear nature of the problem. A general learning methodology for fault detection and diagnosis has been extensively studied, especially for steady-state systems [3] and for dynamic systems [4]. This paper elaborates the application of neural networks in fault detection and diagnosis. In the following sections, an introduction to ANN will be presented. This is will be followed by the case studies with results, discussions and conclusion drawn from the work.

Artificial Neural Network

Artificial Neural Network (ANN) is a computing tool that is inspired by the capabilities of human brains. An ANN is constructed of interconnected basic elements called nodes or neurons. A schematic diagram of this neuron is shown in Figure 1. Similar to their biological counterpart, these neurons are capable of processing incoming information and transferring them to other neurons.

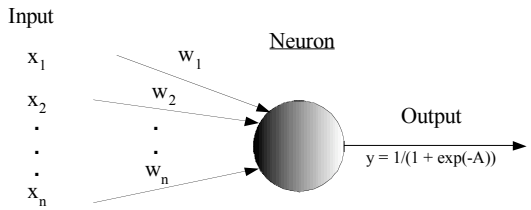

Figure 1 - An Individual Neural Network Processing Unit (Neuron).

The input signals come from either the environment or outputs of other neurons through connections as specified by the network architecture. Within each neuron, input signals are summed and transformed using an activation function before being sent to other neurons. Transformation of data via activation functions is needed to impart pattern-mapping capability to the networks. An example of such functions is the sigmoid function given by the following equation:

$$f(z) = (1 + e^{-z})^{-1} \quad (1)$$

Here, z is a weighted sum of all inputs. Associated with each connection is an adjustable value called network weights. During learning, these weights are adjusted to fulfill the training objective. Effectively, network weights serve as a measure for connection strength that controls the influence of each incoming signal on the recipient neuron.

Feedforward Network

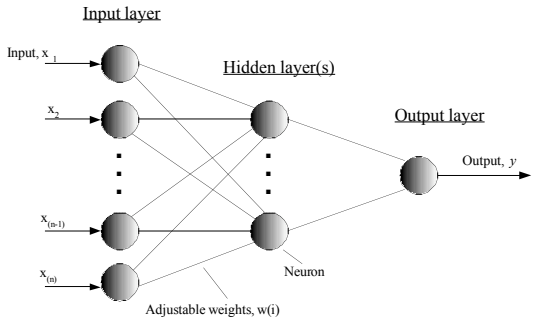

Figure 2 - A Three-layer Feedforward Neural Network.

In process engineering applications, the most commonly used network architecture is the multilayer feedforward network as displayed in Figure 2 above. This network is also known as multilayer perceptron. It is constructed of neurons arranged in several layers. There is an input layer to receive the incoming data to the network, and an output layer to deliver the processed data from the network. In between these two layers, there could be several layers known as the *hidden layers*. Except for those in the input layer, all neurons carry out information processing as mentioned earlier. The selection of input variables is carried out using various considerations. The aim is to include all strong relationships and neglecting some of the weaker links for the same of model simplicity. Experiences have shown that by feeding some of the delayed values of the outputs as inputs to the network, the prediction capabilities are improved significantly. The input-output relationships between the variables can be represented by equation (2) below:

$$\hat{y}_i = f\{u_1(k),...u_1(k-n);u_2(k),...u_2(k-n); \\ ...;y_i(k-1)...y_i(k-m)\} \quad (2)$$

where u_i is the input and $\hat{y}_i$ is the predicted output and $y_i(k-1)$ is the delayed output signals. The number of delayed signals to be used for the case of both input and output variables depends on the process. Looking from the perspective of linear modeling, decisions can be based on the model order. However, this may not be strictly followed here. The main aim is to accommodate the effect of time delays and any uncertainties associated with them.

Feedforward network architecture has been used for fault classifier.

Among the several architectures found in literature, recurrent neural networks (RNN) involving dynamic elements and internal feedback connections, have been considered as more suitable for modeling non-linear systems than feedforward networks [5]. In the last few years, various works have been presented showing that recurrent neural networks are quite effective in modeling non-linear dynamical systems [6-7]. Recurrent Elman network has been used as estimator for modeling a palm oil fractionation plant

Elman Network

For modeling purposes it is assumed that the plant to be controlled is a multivariable plant, with *m* inputs and *q* outputs, describe by a general non-linear input-output discrete time state space model:

$$x(k+1) = f\{x(k), u(k)\} \quad (3)$$

$$y(k) = g\{x(k)\} \quad (4)$$

where $f : \Re^{n+p} \rightarrow \Re^n$ and $g : \Re^n \rightarrow \Re^q$ are non-linear functions; $u(k) = \in \Re^m$, $y(k) \in \Re^q$ and $x(k) \in \Re^n$ are, respectively, the input vector, the output vector and the state vector, at a discrete time k.

Elman Network Architecture and Dynamics

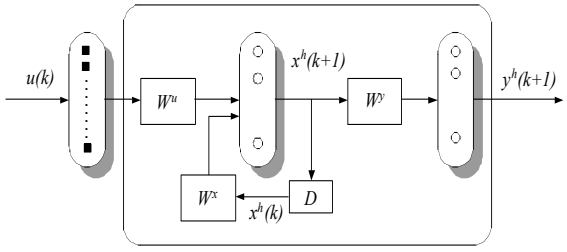

Figure 3 – Block Diagram of a Elman Network

Elman [8] has proposed a partially recurrent network, where the feedforward connections are modifiable and the recurrent connections are fixed. Additionally to the input and the output units, the Elman network has a hidden unit, $x^h(k) \in \Re^n$. $W^x \in \Re^{nxn}$, $W^u \in \Re^{nxp}$ and $W^y \in \Re^{qxn}$ are the interconnection matrices, respectively, for the context-hidden layer, input-hidden layer and hidden-output layer. Theoretically, an Elman network with *n* hidden units is able to represent a n^{th} order dynamic system. Figure 3 shows block diagram of Elman network. The dynamics of the Elman network is describe by the difference equations (5) – (7).

$$s(k+1) = W^x x^h(k) + W^u u(k) \quad (5)$$

$$x^h(k+1) = \varphi\{s(k+1)\} \quad (6)$$

$$y^h(k+1) = W^y x^h(k+1) \quad (7)$$

where $s(k) \in \Re^n$ is an intermediate variable and $\varphi(.)$ is an hyperbolic tangent function.

Network Training

An important part of the neural network model development is the training stage where the optimum connection weights are determined through some selected optimization algorithm. During training, the optimization algorithm (often referred to as learning rule) adjusts the network weights so that the error between the actual output and the target output is minimized. One commonly used error criterion is the mean squares error given by equation (8) below:

$$E = \frac{1}{N} \sum_{i=1}^{N} (t_i^{(m)} - y_i^{(m)})^2 \quad (8)$$

Here, *N* denotes the number of training data presented to the input layer, $t_i^{(m)}$ represents the desired value of the *i*th output element given the *m*th data, while $y_i^{(m)}$ is the actual output of the same element.

As an illustration, consider a backpropagation algorithm, which is a simple gradient-based technique that has been widely used. The weight update equation for this class of algorithm can be represented by equation (9) below.

$$W_{ji}^{(m)} = W_{ji}^{(m-1)} + \Delta W_{ji}^{(m)} \quad (9)$$

$W_{ji}^{(m)}$ denotes the weight of the connection between the *j*th element of the upper layer and the *i*th element of the lower layer in the *m*th learning iteration. For example, in the case of back propagation algorithm, the required weight change $\Delta W_{ji}^{(m)}$ is calculated as follows:

$$\Delta W_{ji}^{(m)} = \eta \, . \, \delta_j^{(m)} \, . \, O_i^{(m)} + \alpha \, \Delta W_{ji}^{(m-1)} \quad (10)$$

Here, η and α denote the learning rate and coefficient of the momentum term, $O_i^{(m)}$ is the output value of the *i*th element in the previous layer and $\delta_j^{(m)}$ is the gradient descent term. Other gradient-based methods also implement similar strategies, the main difference being the computation of the required weight change (i.e., equation (10)). In order to overcome some of the limitation encountered when using any first order gradient-based optimization technique, higher order methods are now more commonly used.

Selection of Network Topology

Another important stage in neural network model development is the selection of network topology, i.e., the number of neurons in each of the network layer and its

connection. Once the type of the network has been chosen, e.g., multilayer feedforward network, the orientation of network connection is defined. Similarly, the number of neurons in the output and input layers are also defined by the problem and hence, the emphasis on the topology selection is to determine the number of neurons in the hidden layer. The aim is to develop models that are accurate and robust. Neural network is known to be able to map any continuous function to any arbitrary accuracy [9]. This implies that the network can learn the relationship between any set of inputs and outputs so that when given the inputs, the outputs can be reproduced. To obtain sufficiently good approximation qualities, a network with "sufficient" neurons must be trained. In doing so, that the weights associated with all the connections within the network are optimized to achieve the desired input/output mapping. In developing the model, what is important is the ability of the model to predict the behavior on "unseen" process. Therefore, the decision on the "optimal" model structure should not be made based on the performance of the model to reproduce the training set only because the resulting model may not fulfill the robustness requirement. Although there are many methods or criteria that are available to determine the best fit of the model parameters, in this work cross validation approach is adopted.

The concept of topology selection using cross-validation is that after estimation using a given sample of data, the quality of the mapping is evaluated using a different set of data. The best mapping is defined as the one that minimizes the prediction error on a data set for which it was not trained. This approach of topology selection involves iterative efforts. Beginning with some small number of hidden neurons, the search continues until the desired performance is achieved.

Process Description

The application of neural network in fault detection and diagnosis is implemented to the palm oil fractionation process plant. The process consists of five distillation columns connected in series. A simplified schematic diagram of the plant is provided in Figure 4. The feed to the plant is Palm Kernel Oil (PKO), which contains fatty acids from C6 - C18. In the pre-cut column, light products (C8 and C10) are recovered in the overhead. The bottom stream is fed to the light-cut column where C12 is separated from the rest of its constituents. The bottom product then enters the middle-cut column where C14 and C16 are recovered leaving the C18 to be purified from the rest of the oil constituents in the still-cut column. In the residue cut column, C18 is recovered and recycled back to the previous column. All the distillation columns operate at highly vacuum condition and are generated using steam ejectors. The columns are packed with structured packing provide the desired separation properties.

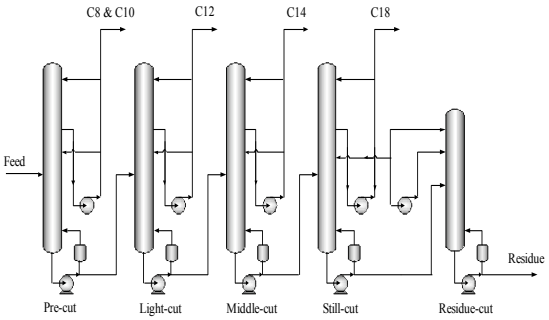

Figure 4 – Schematic Diagram of a Fractionation Process

Due to high operating temperatures, thermal oil is used for heating in the reboilers. A sectional view of a typical column is shown in Figure 5. The column is equipped with a pump-around system. Liquid collected on the trap-out tray is drawn out from a side-draw stream. The stream is split into two – a reflux stream and a stream that goes through an external cooler. The cooled stream is split again into two streams, one returning to the column as a recycle and one as the distillate. The pump-around system provides a means for the vapor in the column to be condensed through direct contact with the cooled liquid from above.

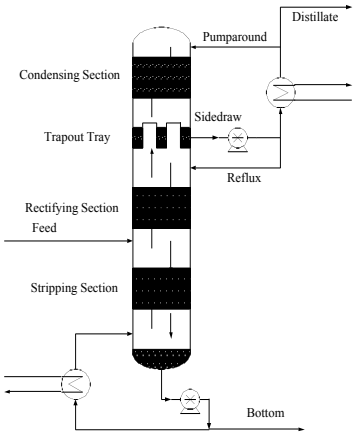

Figure 5 – A Typical Packed Column

This investigation focused on the malfunctions of the process caused by the failure of the temperature and pressure sensors in the pre-cut column. Faulty conditions are simulated using the HYSYS.Plant. Sensor failures are created causing the normal process operation to shift to a faulty operation mode. Effects of these faults are expressed by the composition of C8 in the overhead stream and the composition of C12 in the bottom stream leaving the pre-cut column. During implementation, sensor failures are categorized into high and low readings.

Modelling and Simulation

Simulation of the column was carried out using HYSYS.Plant. Due to the non-conventional nature of the palm oil distillation system, the development of the flowsheet for the simulation has not been straightforward. For example, one of the long-chained fatty acid, caprylic acid (C8) is not found in HYSYS Component Properties Library. As such, C8's properties have to be estimated by HYSYS Hypothetical Component Manager. One has to provide as much data as possible so the estimation will be realistic. HYSYS.Plant does not support packed column as such, the equivalent tray (HETP) calculation was used. Similarly, model of a direct contact internal condenser based on packed column is also not available within the standard library and modifications have also been implemented. In the simulation, UNIQUAC activity model is selected for physical property calculations due to the facts that fatty acids (carboxylic acids) are polar component with non-ideal behavior. The vapor phase fugacity is calculated using Virial model since the system displays strong vapor phase interactions. The conditions of the streams are assumed first. Then the iterative calculations are carried out automatically until the values in the calculated streams match those in the assumed stream within specified tolerances. Proper initial values should be chosen for these streams; otherwise the system might converge to different values, which is not desirable due to the non-linearity and unstable characteristics of the process [10].

The column is a special type of Sub-Flowsheet in HYSYS. A Sub-Flowsheet contains equipment and streams, and exchanges information with the Parent Flowsheet through the connected streams. From the Main Environment, the Column appears as a single, multi-feed multi-product operation. When columns are modeled in steady-state, besides the specification of the inlet streams conditions, pressure profiles, numbers of trays and the feed tray number, two specifications need to be given for columns with reboiler and condenser. These could be the duties, reflux rate, draw stream rates, composition fractions etc. We have chosen reflux ratio and overhead composition mole fractions. The tray sections of the columns are calculated with the HYSYS Tray Sizing utility. Though the tray diameter, weir length, weir height, and the tray spacing are not required in steady state modeling, they are required for an accurate and stable dynamic simulation. Model of column with a direct internal condenser is not available within the HYSYS library. As a result, modifications of a HYSYS standard column template had been carried out to produce an equivalent configuration.

Neural Network Fault Detection Scheme

The proposed fault detection scheme is hierarchical in structure. The schematic diagram of the strategy is shown in Figure 6 below. In short, there are two types of model developed using neural network. The first is the estimator that will always estimate the "normal" or fault-free process behavior. Next, there is the fault classifier that will identify the sources of fault that takes place.

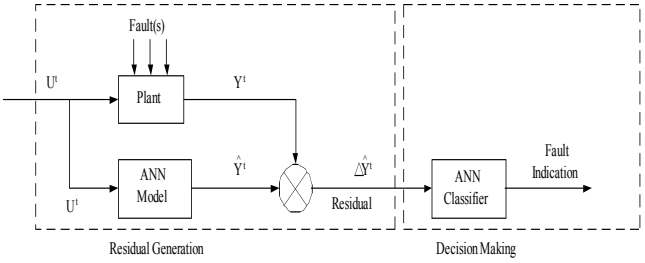

Figure 6 – The Model-Based Fault Detection Scheme

The hierarchical approach is advantageous because it alleviates the chances of misidentification of normal operation trend that is due to the manipulation of the feeds condition. In practice, there are always possibilities that the manipulation of feeds will produce process conditions that coincidentally match the fault pattern and the classifier will tend to misinterpret the situation. The use of residuals provides some protection to the system.

Process Estimator

In this study, the ANN model development efforts were carried out using the neural network toolbox available within MATLAB software. A number of important issues must be addressed when dealing with the development of the neural network estimator.

The first step was selecting the input variables and output variable for the neural network estimator. The inputs were selected based upon their availability in the actual industrial column and their effects on the top product composition. The inputs had been identified consist of 12 variables; column top temperature, column bottom temperature, column middle temperature, column top pressure, reflux flowrate and distillate flowrate as well as a one-sampling-interval-delayed signal of each these variables. The output variable is the composition of C12 (% mass) in the distillate stream.

Plant test data generation was designed and conducted in Hysys.Plant so as to capture the dynamics of the column behavior. A data set consisting of 6000 data points was obtained. The data was divided into two sets, a training set and a validation set.

The next stage concerned with the development of an artificial neural network (ANN) model that correctly mapped the input variables to output variable. Elman network with structure shown in Figure 3 had been chosen to train the data. The Elman network has *tansig* neurons in its hidden (recurrent) layer, and *purelin* neurons in its output layer. This combination is special in that two-layer networks with these transfer functions can approximate any function (with a finite number of discontinuities) with arbitrary accuracy. The only requirement is that the hidden layer must have enough neurons. More hidden neurons are needed as the function being fit increases in complexity.

Network training was implemented using gradient descent with momentum and adaptive learning backpropagation algorithm (traingdx). The network was cross-validated at every batches training and thus the cross-validation errors of the network were monitored throughout the training. Network weights and biases were selected based on the minimum cross-validation error achieved in the training.

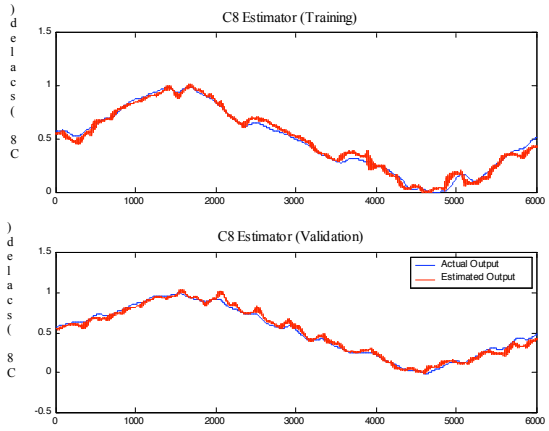

Figure 7 - Training and Validation of Neural Network Model

The results revealed that the optimum network structure was established with 19 hidden neurons. Figure 7 displays the performance of ANN in tracking the actual process data during the training and validation stages. Good performance in the validation set indicated that the network was able to represent the behaviour of the process in different operating conditions than that of the training set. In other words, the network is capable of estimating the product composition based on "unseen" data.

Fault Classifier

The fault classifier was constructed using multi layer feedforward network with single hidden layer. Network training was implemented using Levenberg-Marquardt learning algorithm. The network was also cross-validated at every batches training.

This work focused on the detection of corrupted sensor measurement. Sensor faults were simulated as if there were sudden failures of the measurement systems resulting in measurements bias. The extent of the sensor biases were used as indicators for severity of the sensor fault. Since in the operation of the palm oil fractionation process, column temperature and top column pressure are the two important variables to be controlled, these variables were considered as the focus of this work. Both positive and negative bias measurements of these sensors were simulated for fault detection. Table 1 shows the list of process faults involved.

Table 1 - List of Sensor Faults

1	Column Temperature Sensor +ve bias	F1
2	Column Temperature Sensor –ve bias	F2
3	Top Column Pressure Sensor +ve bias	F3
4	Top Column Pressure Sensor –ve bias	F4

All these faults influence not only the performance of the control loops within the plant, but also the stability of the process. For example, when F3 with 3 % bias was introduced, the reactor pressure increased exponentially to instability. Some of these impacts are shown in Figure 8 below.

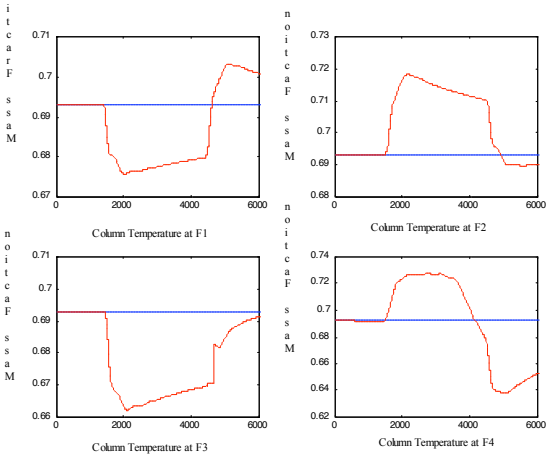

Figure 8 – Effect of Sensor Faults of Distillate C8 Composition

The output data for sensor faults were designed to spread linearly between 0.05 and 0.95. 0.05 and 0.95 were used to represent the process during normal condition and violation of process limit respectively. The network structure employed the distillate composition as the input. The outputs were the various faults F1, F2, F3 and F4. Training was accomplished using Levenberg-Marquadt algorithm and a network with 21 nodes in the hidden layer was considered optimum.

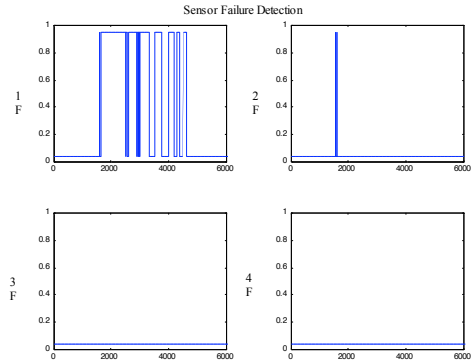

Figure 9 – Detection of Fault With +ve 3% Bias in Column Temperature (F1)

The results obtained revealed the success of the classifiers in detecting the faults introduced to the system. For example, Figure 9 illustrates the detection of Fault 1. When a 3% bias introduced to column temperature sensor the classifier F1 successfully detected the fault. This is clearly displayed in Figure 9 where the network output reached index of 0.95, as designed to indicate a process fault.

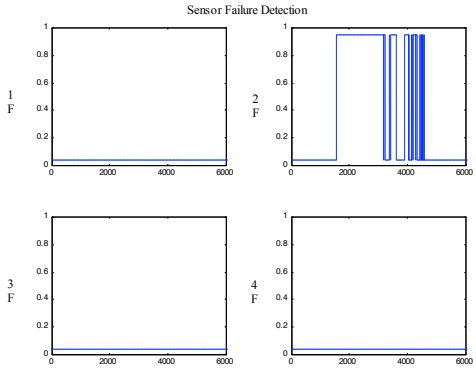

Figure 10 – Detection of Fault With –ve 3% Bias in Column Temperature (F2)

Similarly, when a –ve 3% bias introduced to column temperature sensor, the outputs of classifier F2 showed the index 0.95 to acknowledge the faulty condition as shown in Figure 10. Again, the process fault was successfully detected. Similar observations were established when other fault patterns were introduced. In all cases, the proposed scheme had been successful in detecting the failures. However, due to lack of space, the results are not shown here.

Conclusion

This paper has revealed the capabilities of neural networks in process fault detection. Fault detection is currently one of the largest application domains of neural network. Neural network models have effectively been applied as both the process estimator and fault classifier. The fault detection scheme consists of two models, the first is estimator that will estimate the normal of fault-free process behavior and the second is classifier that will identify the sources of fault that takes place. The proposed hierarchical structure has also been proven practicable. The dynamic detection scheme presented here is useful in fault monitoring because rather than having a discrete detection, continuous detection allows operators to anticipate potential process failures. With an early detection, the engineer has a change to stop the faulty condition before it grows into a full malady.

References

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

[2] Paul, L. 1981. *Failure Diagnosis and Performance Monitoring,* Marcel Dekker.

[3] Ventakasubramanian, V., Vaidyanathan, R., and Yamamoto, Y. 1990. Process Fault Detection and Diagnosis using Neural Networks – I. Steady-state Processes, *Compu. Chem. Eng.* 14: 699.

[4] Patton, R.J., Chen, J. and Siew, T.M. 1994. Fault Diagnosis in Nonlinear Dynamic System via Neural Networks, Inter. Conf. on Control '94, IEE, London, UK 2 1346.

[5] Linkens, D. and Nyongesa, Y. 1996, Learning systems in Intelligent Control: An Appraisal of Fuzzy, Neural and Genetic Algorithm Control Applications, *IEE Proc. Control Theory App.,* 134(4): 367-385.

[6] Parlos, A., Chong, K. and Atiya, A. 1994, Application of the Recurrent Multilayer Perceptron In Modelling Complex Process Dynamics, *IEEE Trans. On Neural Networks,* 2, no 2: 252-262.

[7] Draye, J., Pavisic, D. and Libert, G. 1996, Dynamic Recurrent Neural Networks: A Dynamical Analysis, *IEEE Trans. On Systems Man and Cybernetics,* Part B, vol 26, no 5: 692-706.

[8] Elman, J. 1990, Finding Structure in Time, *Cognitive Science,* 14: 1789-211.

[9] Hornik, K., Stinchcombe, M. and White, W. 1989, Multilayer Feedforward Neural Networks are Universal Approximators. *Neural Networks*, **2**, No.5, 359.

[10] HYSYS.Plant User's Manual. 2000, Hyprotech Ltd., Calgary, Canada.