8-2000

Automated neural network-based instrument validation system

Xiao Xu

To the Graduate Council:

I am submitting herewith a dissertation written by Xiao Xu entitled "Automated neural network-based instrument validation system." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

J. Wesley Hines, Major Professor

We have read this dissertation and recommend its acceptance:

R. E. Uhrug, B. R. Upadhyaya, B. MacLennan

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Xiao Xu entitled " Automated Neural Network-Based Instrument Validation System." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

J. Wesley Hines, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor and
Dean of The Graduate School

Automated Neural Network-Based Instrument Validation System

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Xiao Xu

August 2000

Dedicated to my daughter, Kristine, my wife, Li, and my parents

Acknowledgments

There are so many people to whom I am grateful for making my time at the University of Tennessee so rewarding. I have benefited greatly from knowing the faulty and graduate students in the Department of Nuclear Engineering over the last five years, and hope these relationships will endure. I am particularly grateful to the members of my Dissertation Committee, Dr. J. Wesley Hines, Dr. R. E. Uhrig, Dr. B. R. Upadhyaya, and Dr. B. MacLennan of Computer Science Department, for their support, valuable comments, and encouragement. The author is most grateful to Dr. Hines for his advice and support that made the completion of this work possible.

I would like to thank Dr. Uhrig and Dr. Tom Kerlin for providing me an opportunity to continue my education here at the University of Tennessee.

This study is sponsored by several companies through the Maintenance and Reliability Center. I would like to thank the Tennessee Valley Authority for providing the data for system design and testing, and Mr. Cyrus Taft of the Electric Power Research Institute (EPRI) Instrument and Control (I & C) Center for his tireless help in collecting the data.

Finally, I would like to thank my wife, Li Huang, for her continuous support and patience.

Abstract

In a complex control process, instrument calibration is periodically performed to maintain the instruments within the calibration range, which assures proper control and minimizes down time. Instruments are usually calibrated under out-of-service conditions using manual calibration methods, which may cause incorrect calibration or equipment damage. Continuous in-service calibration monitoring of sensors and instruments will reduce unnecessary instrument calibrations, give operators more confidence in instrument measurements, increase plant efficiency or product quality, and minimize the possibility of equipment damage during unnecessary manual calibrations.

In this dissertation, an artificial neural network (ANN)-based instrument calibration verification system is designed to achieve the on-line monitoring and verification goal for scheduling maintenance. Since an ANN is a data-driven model, it can learn the relationships among signals without prior knowledge of the physical model or process, which is usually difficult to establish for the complex non-linear systems. Furthermore, the ANNs provide a noise-reduced estimate of the signal measurement. More importantly, since a neural network learns the relationships among signals, it can give an unfaulted estimate of a faulty signal based on information provided by other unfaulted signals; that is, provide a correct estimate of a faulty signal. This ANN-based instrument verification system is capable of detecting small degradations or drifts occurring in instrumentation, and preclude false control actions or system damage caused by instrument degradation.

In this dissertation, an automated scheme of neural network construction is developed. Previously, the neural network structure design required extensive knowledge of neural networks. An automated design methodology was developed so that a network structure can be created without expert interaction. This validation system was designed to monitor process sensors plant-wide. Due to the large number of sensors to be monitored and the limited computational capability of an artificial neural network model, a variable grouping process was developed for dividing the sensor variables into small correlated groups which the neural networks can handle. A modification of a statistical method, called Beta method, as well as a principal component analysis (PCA)-based method of estimating the number of neural network hidden nodes was developed. Another development in this dissertation is the sensor fault detection method. The commonly used Sequential Probability Ratio Test (SPRT) continuously measures the likelihood ratio to statistically determine if there is any significant calibration change. This method requires normally distributed signals for correct operation. In practice, the signals deviate from the normal distribution causing problems for the SPRT. A modified SPRT (MSPRT) was developed to suppress the possible intermittent alarms initiated by spurious spikes in network prediction errors.

These methods were applied to data from the Tennessee Valley Authority (TVA) fossil power plant Unit 9 for testing. The results show that the average detectable drift level is about 2.5% for instruments in the boiler system and about 1% in the turbine system of the Unit 9 system. Approximately 74% of the process instruments can be monitored using the methodologies developed in this dissertation.

Table of Contents

List of Tables

List of Figures

Nomenclature

A	decision boundary in SPRT
B	decision boundary in SPRT
B_r	incomplete beta function
Cond	condition number
cp	cumulative percentage of variation
D	process memory
d	detector measurement
E(x)	expectation of x
E_c	least square residual
E_N	SLSR's vector
EG	error goal
e^*	model residual
G	gain factor in MSPRT
K	number of data samples
lr	learning rate
m	number of mapping as well as demapping layer nodes
m_1	faulted mean of SPRT
mc	momentum constant
mse	mean squared error
msw	mean squared weight
N	number of steps for determining on hidden unit by random search
P	loading matrix
p	loadings
Q	loading matrix
q	loadings
r	residual between measurement and estimate
S	singular value matrix
S'	truncated singular value matrix

SF	stability factor
s	singular value
T	score/latent matrix
t	scores
U	score/latent/principal component matrix
u	scores
V	right singular value matrix
W	weight matrix
w	weight vector
X	measurement matrix
$\overline{X}$	mean values of X
$\hat{X}$	model estimate of X
y	desired measurement
$\hat{y}$	model estimate of y
α	false alarm probability
β	missed alarm probability
Φ	cumulative distribution function
ϕ	distribution density function of Φ
γ	performance ratio of weight decay
λ	likelihood ratio of SPRT
σ	standard deviation
ψ	distribution density function

List of Abbreviations

AANN	autoassociative neural network
AI	artificial intelligence
APE	average percentage error
AR	autoregressive
ANN	artificial neural network
ASME	American Society of Mechanical Engineers
ANFIS	adaptive neural fuzzy inference system
EMB	extended Markov blanket
EPRI	Electric Power Research Institute
EWMA	exponentially weighted moving average
GCC	general consistency checking
KFT	Kalman filtering technique
LSR	least square residual
MSET	multivariate state estimation technique
MPE	mean percentage error
MSPRT	modified sequential probability ratio test
NLPLS	non-linear partial least squares
PCA	principal component analysis
PC	principal component
PCP	principal components pruning
PCR	principal component regression
PCS	principal component space
PDF	probability density function
PE	processing element
PEM	process empirical modeling
PLS	partial least squares
RS	residual space
SLSR	standardized least square residual

SPE	squared prediction error
SPRT	sequential probability ratio test
SOM	self-organizing map
SVD	singular value decomposition
TDNN	time delayed neural network
TSVD	truncated singular value decomposition
TVA	Tennessee Valley Authority

Chapter 1

Introduction

1.1 Importance of the Research

Plant instrumentation calibration is an important requirement for power plant operation because of safety and availability considerations. When monitoring a complex process, well-calibrated sensors provide measurements that indicate the plant operating condition. Calibration is commonly regarded as the process whereby the scale of a measuring instrument is determined or adjusted on the basis of an informative or 'calibration' experiment (Aitchison and Dunsmore, 1975). It is crucial to have accurate values for control variables for correct and safe control actions. In addition, it is equally important to ensure signals be correct before they can be used for calculating other variables as system parameters. However, if sensors provide inaccurate information for monitoring and control, wrong decisions can be made. Therefore, to ensure the sensors are within a stated calibration range, a non-intrusive method for continuously monitoring sensor status is needed.

It is difficult, or impossible, for operators to visually detect small drifts in sensor instrumentation. These drifts can cause incorrect control actions and decreased process efficiency. The current method to avoid calibration drifts is to manually calibrate sensors on a periodic basis. These calibrations usually require that the instrument be taken out of service and be falsely loaded to simulate actual in-service stimuli. This can lead to equipment damage and incorrect calibration due to adjustments made under non-service

conditions; thus, a non-invasive automatic sensor calibration monitoring method would aid personnel in making on-line assessments of sensor status.

With economical consideration for power plant operations, condition-based maintenance strategies rather than periodic or corrective maintenance strategies are desired. Changing the calibration strategies to be condition-based requires that instruments be manually recalibrated only when their performance is degraded beyond a specified tolerance. Continuous calibration verification of the plant instruments will reduce unnecessary sensor calibrations and give operators more confidence in sensor measurements. Elimination of unnecessary maintenance results in cost savings and reduced plant down time while a better knowledge of the actual state of the process, due to more reliable sensor values, could result in reduced equipment damage, increased plant efficiency and increased revenue.

1.2 Statement of the Problem

The purpose of this research is to develop a sensor calibration monitoring system for on-line power plant instrument verification and calibration; and additionally, to develop a methodology for automated design of the neural network structures. This system is to monitor the condition of power plant process sensors and indicate when a sensor produces faulted measurements, which is caused by the sensor or instrument chain but not by operating condition changes. With the ability of the developed a sensor monitoring calibration system to determine when a sensor has failed or drifted, the current methods of sensor recalibration on a periodic basis will no longer be necessary. This may lead to a significant cost saving.

Recently, artificial neural networks (ANNs) have been widely studied for the purpose of instrumentation calibration verification (Upadhyaya and Eryurek, 1992; Uhrig, *et al.*, 1996; Nabeshima, *et al.*, 1995; Dong and McAvoy, 1994; Kramer, 1992). If a sensor is in calibration, the valid model estimate should match the measured signal at all operating ranges if the measurement is not drifted. A neural network model uses information from correlated signals to estimate the signal we are interested in. Therefore, the model can provide an unfaulted or near unfaulted estimate to a faulted measurement, through comparison the sensor fault can be detected. In this research, an automated artificial neural network technique is employed for sensor validation. The reason for using ANNs is based on their advantages in signal validation over the traditional model-based techniques stated in Section 2.7.

The artificial neural network structures are problem-specified. The determination of neural network structures heavily relies on the designer's experience as well as the prior knowledge on the plants. This dissertation developed an automated procedure for neural network mode construction, therefore the sensor monitoring system is transferable.

Another issue is that a single neural network model cannot handle problems with large number of inputs (signals) due to computational limitations. Therefore, in order to monitor a large number of signals, a series of small-sized neural network models is needed. This raises an issue on how to optimally divide the signals into small-size correlated groups. The most common methods are sensitivity analysis, correlation analysis, and engineering judgement. For a large number of signals to be sub-grouped, a sensitivity analysis requires an extremely large network be trained and is almost

impossible to perform, while a correlation analysis is based only on linear relationships between signals. In practice, most problems have linear or near-linear relationships, and the correlation analysis can explain most of the relationships among signals. Engineering judgement reflects the designer's prior knowledge of the system, and it is literally impossible to be applied in automated design. Therefore the correlation analysis, which can represent most of the information on system dynamics, not only provides an understanding of the physical process, but also makes the automation design feasible.

In many studies of instrument calibration verification systems, a statistical method, the Sequential Probability Ratio Test (SPRT) (Wald, 1945), is applied for sensor status checking (Upadhyaya, *et al.*, 1987; Uhrig, *et al.*, 1996; Erbay and Upadhyaya, 1997; Hines, *et al.*, 1997; Wrest, 1996; Olvera, 1993; Eryurek, 1994). This method continuously provides sensor status information by processing the residuals between the sensor measurement and the neural network estimate. In order for the SPRT to work correctly, it is required that the residuals be normally distributed. Ideally, the signal residuals represent the signal noise in the presence of unfaulted sensors. Since signal noise is somewhat normally distributed, the normal SPRT is usually quite applicable for on-line detection. However, the noise may not be normally distributed and the neural network may learn some degree of noise and incorporate it into the plant model. Therefore the residual is not pure noise and can deviate from the normal distribution. Furthermore, the spurious spikes due to system perturbation or other reasons may exist in the signals. Due to this deviation and spurious spikes, the SPRT may produce intermittent alarms.

In this dissertation, a study involves the development and evaluation of a system that integrates sensor validation and diagnostics. Furthermore, an automated methodology for variable selection and neural network design is developed. This research aims to significantly reduce system failures due to faulty signals and to enable automatic artificial neural network design without extensive knowledge of neural networks.

1.3 Contributions of the Dissertation

Traditionally, instruments are manually calibrated on a periodic basis. In recently developed systems, the utilized neural networks are designed using trial-and-error methods to minimize the network prediction error and prevent overfitting. This dissertation introduces a new methodology of automated neural network design that includes an automated variable selection and grouping, as well as an automated process to determine the number of hidden nodes. Furthermore, a modification on the traditional SPRT is performed to eliminate the occurrence of intermittent alarms due to residual noise deviating from the normal distribution or due to spurious spikes in the signals.

The contributions made by this research that will be detailed in this dissertation are:

1. the development of a methodology for automated neural network variable selection using the technique of non-linear partial least squares (NLPLS) (Geladi and Kowalaski, 1986; Hoskuldsson, 1988; Qin and McAvoy, 1992) as well as signal correlation analysis;

2. the development of a methodology for automated neural network input grouping using correlation analysis;

3. the development of a methodology to estimate the number of neural network hidden nodes in a three hidden layer autoassociative architecture;

4. the development of a modified SPRT (MSPRT) that suppress the intermittent system alarms due to non-normally distributed residuals and spurious spikes in the signals.

1.4 Organization of the Dissertation

The dissertation begins with a review in Chapter 2 of the previous and current studies involving the topics related to this work. This includes a survey of the sensor fault detection systems as well as various technologies and methods used for developing these systems. In Chapter 3, the structure of the modular sensor monitoring system is introduced. The functions of the individual modules combined into this monitoring system are introduced. Methodologies for developing these modules are also briefly discussed in Chapter 3. In Chapter 4, the monitorable variables are identified using an automation method. A new automated algorithm for variable grouping is also detailed in this chapter. The corresponding neural network models are then established using the developed automation procedure. Chapter 5 is the core chapter of this dissertation. It concentrates on the neural network design using the developed methods. It includes network type selection, network training and testing. The SPRT issues are also addressed in this chapter. Chapter 6 presents some of the detection results on artificial drifts, such as slow drift, small drop and gross failure. A discussion of these results is also made in Chapter 6. Finally, Chapter 7 summarizes this research work and addresses some difficulties of the current work and possible solutions to be addressed in future research.

Chapter 2

Literature Review

This chapter briefly reviews the literature on detection, isolation, and identification of sensor faults in plants as well as the techniques used in estimating the instrument measurements. Literature on relevant techniques, such as the artificial neural networks, partial least squares (PLS), principal component analysis (PCA) as well as other techniques are also reviewed. The purpose of this discussion is to present the current status of the topic of interest. Although the periodic manual calibration method is still in use, a number of studies have been conducted to attempt to enhance the capability of sensor status prediction for predictive maintenance purposes.

Conventional model-based techniques have been widely used in process control, monitoring and diagnostics. These techniques include polynomial fitting, statistic modeling, and state space representation modeling. More recently, applied artificial intelligence methods have been increasingly studied for the purposes of control and diagnostics. There is published research involving sensor calibration verification systems using artificial neural networks as well as other techniques in power plants (R. Dorr, *et al.*, 1997; Qin and Li, 1999; Dunia and Qin, 1998; Singer, *et al.*, 1995; Fantoni and Mazzola, 1996; Uhrig, *et al.*, 1996). There are also a number of studies on neural network optimization (Williams, 1995; Goutte, *et al.*, 1997; Hashem, 1997). The following sections discuss these topics and give an overview of the approaches used.

2.1 Probabilistic Reasoning

Ibarguengoytia *et al.* (1996) establishes a probabilistic reasoning model for the purpose of sensor validation, in which a probabilistic model is presented to detect the inconsistency in measured data. A process variable is connected with other related variables to form a lattice through an event tree called Markov blanket based on knowledge of the process. A fault detection mechanism using probabilistic propagation through the lattice finds a potential faulty sensor. Finally, constraint management is applied to distinguish a real fault from the apparent ones. The difficulty is that this technique cannot detect which sensor has the real fault if two or more sensor faults are included in the same extended Markov blanket (EMB); for example, the flow of gas and the flow of air. This technique is also incapable of detecting small drifts present in most practical problems.

2.2 Redundancy Analysis

Dorr *et al.* (1997) presents a method, which is based on simple redundancy and consists of generating residuals by comparing measurements provided by physically redundant sensors. Another method presented in this paper is to use analytical redundancy. In this method, residuals are generated by comparing each measurement with an estimate computed from models of the process. The proposed analytical redundancy technique is based on steady-state relationships between measurements. The selection of the steady-state files is based on the stability factor (SF). The stability of the measured variables is evaluated using SF, which is defined as,

$$SF = 100 \times \frac{X_{\max} - X_{\min}}{X_m}$$

where $X_{\max}$ is the maximum value of data set;

$X_{\min}$ is the minimum value of data set;

X_{m} is the mean value of data set.

The lower the SF, the more stable the variable. The calculation is performed for each variable of the file considered to collect the steady-state files for each variable. For more details on this selection, refer to (R. Dorr, *et al.*, 1997).

2.2.1 Simple Redundancy

Simple redundancy consists of using information derived from physically redundant sensors, which measure the same physical variable in order to generate residuals.

Fault detection and location are performed by comparing these residuals with thresholds, which are dependent on sensor measurement accuracy. A measurement model for simple redundancy fault detection is defined as,

$$x_i = x^* + a_i + e_i, \quad \text{for } i = 1,...,n$$

where a_i, which represent the sum of independent random errors between the n sensors of the considered set for the measurement conditions considered, are normally distributed with zero mean and standard deviation σ. Usually, σ is selected as $\varepsilon_t/2$, while ε_t represents the accuracy of the sensors and e_i represents accidental errors. These errors

are due to an abnormal degradation of the sensors or the instrument chains. It is supposed that there is no common degradation mode. Therefore, these errors are considered as independent.

The principle of sensor fault detection by simple redundancy is to compare each measurement with an estimate of the true value. The estimate is calculated by a linear combination of measurements given by redundant sensors. In this paper, assuming all sensors are of the same model, the estimate is simply the mean of the redundant sensor measurements. A detector d_m, which uses only n-1 sensors to estimate the mean of n measurements, is defined as,

$$d_m = \begin{cases} x_1 - \dfrac{x_2 + \cdots + x_n}{n-1} \\ x_i - \dfrac{x_1 + \cdots x_{i-1} + x_{i+1} + \cdots + x_n}{n-1} \\ x_n - \dfrac{x_1 + \cdots + x_{n-1}}{n-1} \end{cases}$$

In the absence of accidental errors, the mathematical expectancy and the variance of the d_m detector can be computed using,

$$d_m(i) = x_i - \frac{1}{n-1}\sum_{j=1}^{n} x_j, \quad j \neq i$$

$$d_m(i) = a_i - \frac{1}{n-1}\sum_{j=1}^{n} a_j, \quad j \neq i$$

$$E(d_m(i)) = 0$$

$$Var(d_m(i)) = \frac{n}{n-1}\sigma^2.$$

In the presence of a fault of magnitude e_i on sensor i, the mathematical expectancy of the component i of the detector is

$$E(d_m(i)) = e_i$$

and the variance of $d_m(i)$ remains unchanged.

The estimate of the failure magnitude is given by

$$\hat{e}_i = x_i - \frac{1}{n_s} \sum_{j \in C_s} x_j \quad \text{for } i \in C_f$$

where n_s number of nonfaulty sensors;

C_s set of indexes of nonfaulty sensors;

C_f set of indexes of faulty sensors.

Then search for one i such that $|d_m(i)| > \sqrt{n_s/(n_s-1)}\varepsilon_t$ which is the two standard deviation of d_m. If this i does not exist, then no sensor is faulty. Otherwise, search for the largest component j of the detector in absolute value. Sensor j is considered faulty and removed from C_s and added in C_f. The new detector is computed without this sensor. Accordingly, n_s is also updated for further calculation. This process is repeated until all faulty sensors are identified.

The major drawback of simple redundancy is that it does not enable determining which sensor is the most likely to be faulty in a set of two redundant sensors. Moreover, it is not possible to monitor single or nonredundant sensors. This drawback can be overcome with the application of artificial neural network modeling.

2.2.2 Analytical Redundancy

Analytical redundancy uses additional information provided by models such as

$$X = HX^* + a + \varepsilon$$

where the notations are the same as in the simple redundancy model. It is noticed that H must be full column rank.

This problem can be solved by a standardized least square residual (SLSR) analysis, which minimizes the criterion with respect to $\hat{X}$

$$\Phi = \frac{1}{2}\left(H\hat{X} - X\right)^T V^{-1}\left(H\hat{X} - X\right)$$

V is the covariance matrix of measurement errors and supposed to be known. The sensor status is judged by the factor of the SLSR's vector E_N, which is defined as

$$E_N(i) = \frac{E_C(i)}{\sqrt{V_E(i,i)}} \qquad \text{for } i = 1,...,m$$

where E_C is the least square residual vector and V_E is the variance-covariance matrix of LSR. They can be calculated by applying the Lagrange multiplier technique. For further

details refer to (R. Dorr, *et al.*, 1997). The detection procedure is similar to simple redundancy analysis.

The major difference between the simple and analytical redundancy analyses is that the latter technique is able to determine the sensor which is most likely to be faulty in a set of two redundant sensors. More practically, the analytical redundancy method is able to determine nonredundant sensor status which is impossible for simple redundancy method. The drawback of this technique is that a very precise model describing the physical process must be presented, which involves tremendous amount of efforts on analyzing the process, and carefully selecting the variables for modeling. The engineering judgement is heavily involved in this technique, which makes this technique unfeasible for automation.

2.3 Maximized Sensitivity

The mechanism of this approach (Qin and Li, 1999) is to make the *i*th element of a set of structured sensor residuals *r*, which is defined later in this part, insensitive to the *i*th sensor fault but most sensitive to the others. Alternative methods (Narasimhan and Mah, 1988; Dunia, *et al.*, 1996) make one index most sensitive to one fault. The maximized sensitivity technique starts with a normal process model given as

$$Bx^*(t) = e^*(t)$$

where $x^*(t)$ is a vector of normal sensor values, B is the model matrix, and e^* is the model residual which contains measurement noise, process noise, and model errors.

Under normal conditions, the model residual e^* can be assumed to be zero mean Gaussian noise. When a sensor fault occurs, the sensor measurement $x(t)$ contains the normal values of the process variables and the fault,

$$x(t) = x^*(t) + D_i f_i(t)$$

where $f_i(t)$ is a vector of the fault magnitude, D_i is a matrix of fault direction. For a single sensor fault in the ith sensor, $D_i = [0\,0\cdots 1\cdots 0]^T$, which is the ith column of the identity matrix. A set of structured residuals $r(t)$ is generated by transforming the model residual vector $e(t)$ by a matrix W

$$r(t) = We(t)$$

The matrix W is designed that each element of $r(t)$ is insensitive to one particular sensor fault and sensitive to others. The detailed derivation is omitted and can be seen in (Qin and Li, 1999).

To avoid false alarms due to noise, a filter structure, exponentially weighted moving average (EWMA), is introduced to the model residual

$$\bar{e}(t) = \gamma \bar{e}(t-1) + (1-\gamma)e(t)$$

where $0 \le \gamma \le 1$. By using PCA or PLS, the sensor values $x^*(t)$ is decomposed into

$$x^*(t) = Pt + \bar{P}\bar{t}$$

where P are orthogonal eigenvectors associated with the principal eigenvalues of the correlation matrix of $x^*(t)$ and $\overline{P}$ are the remaining eigenvectors associated with the remaining minor eigenvalues. The vectors t and $\bar{t}$ are the principal and residual components, respectively.

Through the PCA or PLS analysis, the matrix W is the eigenvector of $B^0B^{0^T}$ that corresponds to the largest eigenvalue. B^0 is the normalized vector of B, where $B = \overline{P}^T$. This finding is appropriate for either single sensor faults or multiple sensor faults. For details on multiple faulty cases, see reference (Qin and Li, 1999). Several sensor fault detection indices are applied to detect the sensor faults. Process changes are distinguishable from sensor faults with the indication of fault occurring in most of these indices.

Although this method is effective in detecting sensor status and distinguishes the process change from individual sensor faults, the major concern of this method is that it cannot detect the magnitude of the fault. Therefore it cannot provide quantitative information on sensor faults and predict current sensor readings from future data.

2.4 Unified Geometric Approach

Dunia and Qin (1998) proposed a unified geometric approach using the PCA technique to identify and detect sensor faults. Similar work has been done by (Dunia, *et al.*, 1996). The objective of this work is to define a set of faults, including sensor and process faults, in which each member is to be identified using the principal component subspace (PCS) and residual subspace (RS). The PCS includes data variations according to the principal

component model in agreement with the measurement correlations, while the RS includes data variation due to modeling errors and noise in the data.

From the geometric perspective, the sample vector for abnormal operating conditions lies in the measurement space far from the PCS. The procedure that geometrically characterizes the fault determines the location of the sample vector with respect to the PCS. The fault is detectable if the orthogonal distance between the faulty measurements and the PCS is larger than the diameter of the confidence region. The squared prediction error (SPE) is chosen to be the fault identification index. The PCA is used to reconstruct a sensor measurement from the remaining sensors to identify the fault. The drawback of this approach is that the fault direction vector should be known a priori, which requires knowledge of the process and process data that contain the faults. This in general makes the automation process difficult to be developed.

2.5 Multivariate State Estimation Technique (MSET)

Argonne National Laboratory developed a MSET model for nuclear power plant signal monitoring and fault detection (Singer, *et al.*, 1995; Gross, *et al.*, 1997; Singer, *et al.*, 1997). The MSET is a non-linear regression technique developed from the concept of the least square estimation. The least square estimation technique minimizes the sum squared error between the measurement and the estimate, which is represented as below:

$$\hat{X} = D \cdot \left(D^T \cdot D\right)^{-1} \cdot D^T \cdot X$$

where X is the sensor measurements; $\hat{X}$ is the model estimate; D is the process memory.

The least square estimation is a linear technique and requires the matrix $(D^T \cdot D)$ be invertible, i.e., it must be non-singular. However, most of the plant signals are correlated with each other to some degree, which may cause the matrix $(D^T \cdot D)$ to be non-invertable. Also the process can be nonlinear, which makes the least square estimation technique invalid.

Although the linear technique is unusable to this problem, its formalism leads to a relationship between an estimate of the process state, a current measurement and the process history that has several very useful features. Along with other reasons stated in (Singer, *et al.*, 1997), a non-linear form is developed by modification of the linear form:

$$\hat{X} = D \cdot (D^T \otimes D)^{-1} \cdot D^T \otimes X$$

where $\otimes$ is a non-linear operator which can be found in (Gross, *et al*, 1998).

Results (Gross, *et al.*, 1997; Singer, *et al.*, 1997) demonstrate that the MSET is capable of monitoring nuclear systems, detecting and identifying malfunctions and analytically replacing faulted sensors.

2.6 Other Techniques

Some other techniques, such as the generalized consistency checking (GCC), the process empirical modeling (PEM), and the Kalman-filtering technique (KFT) are also used for signal validation. The GCC method is used for the systematic cross-comparison of redundant signals. The KFT method requires prior knowledge of the process of interest. However, with the presence of no redundant sensors and no prior knowledge of the

relationship of the variables, these methods may not be applicable. Erbay and Upadhyaya (1997) applied these methods for on-line signal validation in nuclear power plants.

Other techniques include, an adaptive neural fuzzy inference system (ANFIS) applying the fuzzy logic in combination with a neural network is also applied for on-line sensor validation (Hines, *et al.*, 1997). Concerns have been raised that the number of fuzzy rules and partitions could explosively increase as the number of sensors to be monitored increased, which might prevent this technique from being applied to plant wide sensor monitoring. Furthermore, the requirement of understanding the physical process in order to make proper fuzzy rules removes this method from consideration for automated design.

2.7 Artificial Neural Networks

Artificial intelligence (AI) has become one of the most studied methods for monitoring, diagnostics and control. Neural network modeling is among the most mentioned AI techniques. Recently, a better understanding of new neural network paradigms and the existence of more powerful computers have led to the utilization of neural networks in the areas of validation, monitoring, diagnostics and control. The applicability of neural networks in the power generating industry for monitoring and validation has been demonstrated in a number of studies (Upadhyaya and Eryurek, 1992; Uhrig, *et al.*, 1996; Nabeshima, *et al.*, 1995; Dong and McAvoy, 1994; Kramer, 1992).

For effective control strategies in process industry systems, it is necessary to perform validation and monitoring of important variables. Early applications of validation and diagnostics systems utilized model-based techniques where polynomial fits were used to capture the relationships among variables (Erbay and Upadhyaya, 1997). The Kalman filtering technique uses a linear or quasi-linear model for plant state estimation (Kalman, 1960; Brown, 1992). A more recent approach, artificial neural network modeling, has broadened the capabilities of validation, monitoring, and diagnostics systems even further. This approach provides an opportunity to develop a nonlinear model between related variables.

Artificial neural networks are models inspired by the architecture of the human brain (Haykin, 1994; Aleksander and Morton, 1990). A neural network consists of a large number of highly interconnected processing elements (PEs). A PE, analogous to a neuron, has a number of input paths. It combines the values of the weighted inputs, modifies the combination with a transfer function, and produces an output. In an artificial neural network, the processing elements are organized in a sequence of layers.

Several applications of neural networks for sensor validation have been reported (Upadhyaya and Eryurek, 1992; Uhrig, *et al.*, 1996; Wrest, 1996), showing that the artificial neural network technique is capable of detecting sensor failure and of providing useful information to operators to make correct decisions.

Fantoni and Mazzola (1996) presented a detection system for nuclear power plants using artificial neural networks. This research demonstrates the feasibility of using neural

networks for signal validation in nuclear power plants. A pattern recognition module (SOM) acting as the data preprocessor is designed to cluster the data into several operating regions according to the measured plant state. In such a way, similar plant states (input patterns) are classified and assigned to the same cluster. A series of neural networks corresponding to each plant state is formed. A five-layer neural network was chosen as the network structure. Results are presented based on eight selected correlated variables. In order to filter the monitored signal, six consecutive samples in the time domain were supplied for each signal to the input layer, so one input pattern contains 48 values. Therefore, there were 48 nodes in the input layer. With a developed random fault learning method, the authors claimed that this validation system can also handle multiple sensor failures and show some results. This paper demonstrates the possibility of using neural networks for sensor validation purposes. However, the embedded filter structure using consecutive sampling data points as network input greatly enlarges the size of the neural networks. This results in a more complicated network structure and a large network weight matrix. For plant-wide sensor monitoring, a relatively large number of variables are presented in a neural network model; this filter structure makes the network training extremely difficult or impossible. In this design, only a specific group of sensors were monitored; it is not designed for plant-wide monitoring.

The generation of an accurate model, using model-based techniques, requires an effort which is proportional to the complexity of the system. Neural networks offer several advantages for signal validation, monitoring, and diagnostics over traditional techniques:

- Data-driven. It is unnecessary to define a functional form relating a set of variables. A neural network model is established by analyzing the importance between the predictors and the response variable;

- Nonlinearity. The universal approximation theorem states that a single hidden layer network is sufficient for a multilayer perceptron to uniformly approximate any continuous function, including linear or nonlinear functions (Cybenko, 1989; Haykin, 1994);

- Robustness. Neural networks are by far more fault tolerance than the traditional model-based techniques (Eryurek, 1991). Unlike traditional model-based programs, neural networks can give high performance even when there is a failure in the network structure, such as missing processing elements, or faulty weight connections (Sequin and Clay, 1992);

- Adaptation. When there is a small operating status change, the neural network can be adapted to the new operating condition through retuning using a linear transformation technique – singular value decomposition (SVD) (Masters, 1993; Jolliffe, 1986). Network retraining is not needed. This property makes the on-line sensor status monitoring available.

Neural networks are intrinsically parallel and non-algorithmic. These features make real-time processing of data and information more feasible. Artificial neural networks provide fast responses or solutions to complex problems such as plant-wide monitoring or model-based sensor validation. Once trained, the neural network models are easy to

implement and can be used as real-time failure detection tools due to their inherent robustness and parallel computational architecture.

One of the major concerns in neural networks, as in other data-driven techniques, is the overfitting problem. Because the measurements are more or less noisy, the over-pursuit of accuracy on training data causes overfitting. This is because the network is trained not only to fit the underlying function presented by the training data, but also to try to fit the noise in the function. This makes the network generalization performance unreliable. To avoid this problem, a smaller-sized neural network architecture, in regard to the number of hidden nodes, is proposed to reduce the degree of freedom of the adjustable parameters. Also the employment of a cross validation technique can avoid the overfitting problem. Details on these solutions are discussed in Chapter 5.

Chapter 3

Methodologies for System Development

3.1 Introduction

Traditional approaches to instrument calibration are expensive in terms of labor and economic cost. Manual calibrations may lead to damage of equipment or incorrect calibration. While proper adjustment is vital to maintaining proper plant operation, a less invasive technique is desirable. In this research, an on-line sensor calibration verification system is developed to monitor sensor status and replace faulty sensor instruments if needed. An autoassociative neural network (AANN) architecture is employed to provide the best estimate of the sensor values. Should a sensor failure be detected, the system acts as a virtual sensor to replace the real faulted one with its best estimate. By using this system, plants are able to operate continuously without being interrupted due to some sensors being failed. This may increase the plant operation efficiency and reduce its maintenance costs.

This chapter introduces the sensor validation system as well as the individual modules of this system. Details on the methodologies used for developing these modules are discussed. Also a statistical method was reviewed for faulty sensor detection.

3.2 System Overview

The artificial neural network (ANN) based instrument monitoring calibration system has four major components:

1. Signal estimation module utilizes an AANN architecture to provide the best estimates of signals;
2. Signal checking module determines the sensor status using a statistical decision logic;
3. Signal correction module replaces faulty signals with their best estimates when the fault becomes statistically significant;
4. Network retuning module adapts the network to the new operating conditions when plant operating conditions change.

A block diagram of this sensor calibration monitoring system is shown in Figure 3.1.

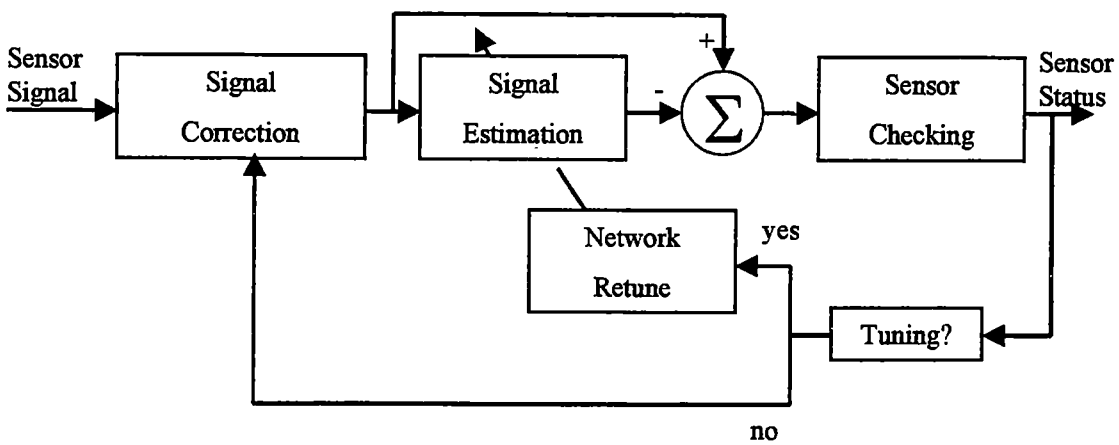


Figure 3.1 Sensor calibration system block diagram

3.2.1 Signal Estimation Module

In this research, an ANN architecture is applied to estimate sensor signals. An ANN can model a plant process at any degrees of nonlinearity (Haykins, 1994). An ANN is a data-driven plant process model that does not require prior knowledge of the plant, unlike a physical or an empirical model. The ANN model is capable of providing estimates of sensor values. In this research, two types of neural networks, the autoassociative neural

network (AANN) and the time-delayed neural network (TDNN), have been studied for architecture selection.

3.2.1.1 Autoassociative Neural Network (AANN)

In an AANN, the outputs are trained to emulate the inputs, which are current snapshot sensor values, over an appropriate dynamic operating range. Many plant variables that have some degree of correlation with each other constitute the inputs. During training, the interrelationships among the variables are embedded in the neural network connection weights resulting in a model of the process.

Figure 3.2 shows a simple AANN-based sensor monitoring module for a group of four sensors whose measurements are correlated to some degree. When a sensor that is an input to the autoassociative network is faulty due to a drift or gross failure, the network still gives a valid estimate of the correct sensor value due to its use of information from other correlated sensors. The estimated sensor value (s_n') is then compared to the actual sensor value (s_n). The difference or residual (r_n) normally has a zero mean and a variance related to the amount of noise in the sensor signal. When a sensor is faulty, the mean or variance of its associated residual changes. This change can be detected by the statistical decision logic.

3.2.1.2 Time-Delayed Neural Network (TDNN)

A TDNN uses both the current sensor measurements and the previous sensor values as input to the network. This provides the network with temporal information. This type of network uses more information and can learn process dynamics. Another advantage of

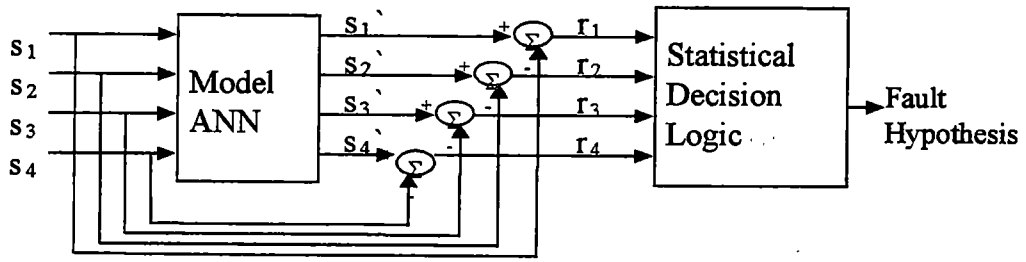


Figure 3.2 AANN-based Sensor monitoring module

using a TDNN is that the network can learn to filter the input signals since it has more than one input (a current value and several previous values) for each sensor. In order for the TDNN to be more valuable, the time delays should be chosen such that the process dynamics are represented. Otherwise, this type of network does not exhibit its advantages over other static networks.

In both network structures, a robust training procedure detailed in Chapter 5 was used to force the network to depend on all the correlated inputs (sensor measurements) to estimate each individual output. As a result, any specific network output shows virtually no change when the corresponding input has been distorted by noise, faulty data, or missing data. This characteristic allows the network to detect sensor drifts or failures by comparing sensor measurements with the corresponding network estimates of the sensor values. A simplified ANN signal error detection schematic is shown in Figure 3.3.

3.2.2 Signal Checking Module

A statistical method called the Sequential Probability Ratio Test (SPRT) (Wald, 1949) was utilized for sensor status checking based on the network residual, the difference

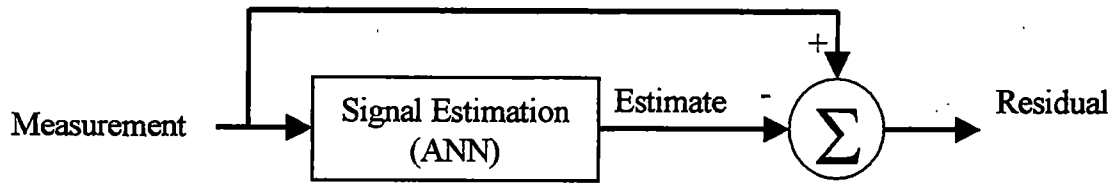


Figure 3.3 Simplified ANN signal error detection schematic

between the measurements and the estimates. The SPRT determines if the network residuals are from a normal or abnormal probability distribution. Unlike other statistical methods which require calculation of a new mean and variance at every new sample, the SPRT continuously updates the values of the probability likelihood ratio, which indicates the sensor status, by processing the updated network residual. Therefore, this method can be used for on-line sensor status checking. The SPRT system is blocked diagramed in Figure 3.4.

3.2.3 Faulty Sensor Correction Module

When a sensor measuring a control signal is classified to be faulty, the best estimate of the signal from the sensor monitoring module can be sent to the control system, for display to plant operators, or for control tasks. The best estimate also replaces the faulty sensor as input into the sensor monitoring module so that the influence of the faulty sensor on the estimate of other signals is minimized. The actual sensor output is

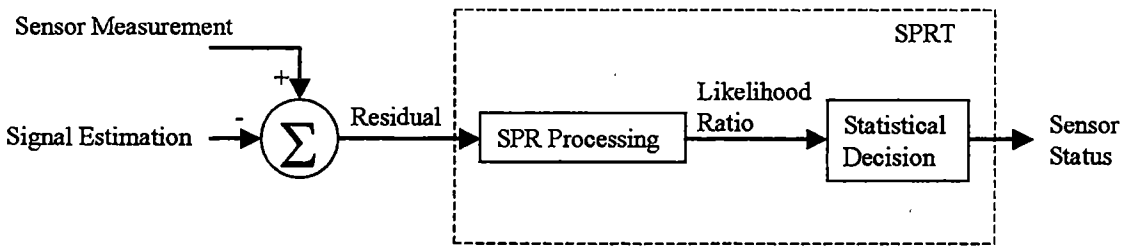


Figure 3.4 SPRT algorithm block diagram

substituted back into the network when the fault has been cleared. This method always gives the operator access to the best estimate of the parameter whether it is the unfaulted measured value or the estimated value.

3.2.4 Network Retuning Module

The neural network learns the interrelationships among the sensor measurements during training. Although the training set should include samples from all plant operating regions, sometimes the plant's operating state may change to one that was not included in the training set. This can be caused by component wear, cyclical changes, or changes in the plant configuration, among others. These changes would be characterized by several residuals deviating significantly from their normal mean of zero. When this happens, the output of the network is not reliable and the network must be retrained to operate under the new conditions. If only one residual deviates from zero, a sensor fault is hypothesized.

The neural network architecture applied in this research is a non-linear principal component analysis (NLPCA) type network, which is a three hidden layer feedforward network proposed by Kramer (1992). It consists of an input layer, three hidden layers (mapping layer, bottleneck layer and demapping layer) and an output layer. The NLPCA architecture is sketched in Figure 3.5.

The hidden layers act as feature extractors and the linear output layer combines these features to provide a desired mapping. If the features do not change when a plant or process operating condition changes, only the output layer weights need to be adjusted to

perform the desired mapping without retraining the entire network. This assumption seems to hold for small changes in operating conditions. Since the output layer is a linear layer, therefore, instead of retraining the entire network, a linear regression technique is applied to regress the model. The training time is significantly reduced, from days of training down to seconds of regression. Retraining the entire network may be necessary for major changes in plant operating conditions when adjusting the output weights does not result in satisfactory performance.

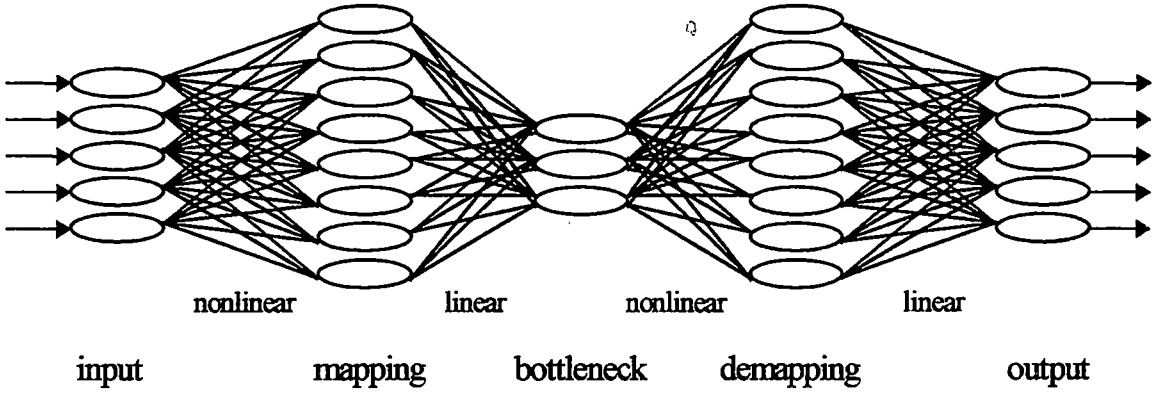


Figure 3.5 NLPCA-typed AANN structure

Retraining only the linear output layer can be achieved by solving for the output layer weights using a least squares procedure. Several methods of solving for the linear output weights exist including standard least squares methods that can cause numerical inversion problems. Other methods used are the LU or QR decompositions (Jolliffe, 1986; Farebrother, 1988). The best method is to use the singular value decomposition (SVD) technique (Press, *et al.*, 1992; Masters, 1993) that uses the most relevant information to compute the weight matrix and discards unimportant information that may be due to noise.

3.3 Neural Network Architecture Selection

The optimal selection of the number of neural network hidden units is an extremely difficult task. Among other things, it heavily depends on the complexity of the problem (Masters, 1993). The size of the neural network greatly influences the network capacity (Baum, 1988; Akaho and Amari, 1990). Current studies estimate the optimal number of network hidden units through either network training or statistic techniques. Fahlman and Lebiere (1990) use a flexible cascade-correlation learning architecture by adding hidden units one by one so as to reduce the output error of the network, the network with the least output error is selected. Similarly, Moody (1992) and Kurita (1990) use an analogy of Akaike's Information Criterion (1974) to estimate the number of hidden nodes by training the networks with different numbers of hidden units. These techniques select optimal network architectures on a trial and error basis, which is improper for automated design.

3.3.1 Beta Method

Fujita (1998) introduces a method to statistically estimate the number of hidden units without training the network. This technique is based on what is called beta function and the original derivation will be presented in this section.

Consider an n-input-1-output feedforward network that consists of a linear output unit and m non-linear hidden units. Suppose that there are K data sets containing both the input and desired output. The input data sets are represented by a $K*n$ matrix X, and the desired output data sets are represented by a $K*1$ vector y. The output of the hidden units are represented by a $K*m$ matrix H, which is generally a function of X:

$$H = f(X)$$

where f denotes the mapping $X \rightarrow H$.

In a one-hidden-layer network, let $\hat{y}$ be an actual output vector produced by the linear output unit of the network as a linear combination of the column vectors of H,

$$\hat{y} = Hw$$

where w is an m-dimensional weight vector. The sum squared output errors which should be minimized is represented as

$$\|y - \hat{y}\|^2 = \sum_{i=1}^{K} (y_i - \hat{y}_i)^2$$

According to the theory of the least squares approximation, the optimal y_0 is given by

$$y_0 = Py$$

where P is the projection matrix onto $L[H]$. If H consists of linear independent columns only, then $H^T H$ is non-singular and P is expressed as

$$P = H(H^T H)^{-1} H^T$$

Hence, the least sum of squared output errors is

$$\|y - Py\|^2 = \|P_c y\|^2 = y^T P_c y$$

where P_c=I-P and I is the identity matrix.

When a new hidden unit is added, let h be the output vector of the added hidden unit. The column space to which y belongs is expanded by one dimension from $L[H]$ to $L[Hh]$, where $[Hh]$ is an augmented matrix. This space expansion brings the decrease of the minimum value of $\|y-\hat{y}\|^2$ from $y^T P_c y$ to $y^T P_c' y$ where

$$P_c' = I - [Hh]([Hh]^T([Hh])^{-1}[Hh]^T = P_c - P_c h(h^T P_c h)^{-1} h^T P_c$$

Let Δ be the decrease of the output error, which can be expressed as

$$\Delta = y^T P_c y - y^T P_c' y = y^T P_c h(h^T P_c h)^{-1} h^T P_c y = \frac{(y^T P_c h)^2}{(h^T P_c h)}$$

The h that maximizes the Δ is the best one for reducing the output error.

The number of required hidden units for reducing the output error depends on how large the Δ of each unit can be. The expected largest value of Δ is estimated statistically, based on a certain supposition that the largest value of Δ is obtained by random search for h.

For the convenience of theoretical treatment, Δ is rewritten as

$$\Delta = \left(\frac{P_c y}{\|P_c y\|}, \frac{P_c h}{\|P_c h\|} \right)^2 \|P_c y\|^2$$

where the term within the parentheses denotes the inner product. This inner product corresponds to the inner product (u,v) of $(K\text{-}m)$-dimensional unit vectors u and v.

In order to estimate the expected largest value of Δ, let us consider the distribution of $r=(u,v)^2$ in the range of $0 \le r \le 1$. Suppose that u is constant and v is uniformly distributed on the surface of the (K-m)-dimensional hypersphere. In this case, the cumulative distribution function $\Phi(r)$ can be expressed by the beta distribution as follows:

$$\Phi(r) = cB_r\left(\frac{1}{2}, \frac{K-m-1}{2}\right)$$

Where B_r is the incomplete beta function and c is the inverse of the beta function $B(1/2,(K\text{-}m\text{-}1)/2)$. The largest value of r can be obtained through the following distribution density function ψ_N:

$$\psi_N(r_N) = N\Phi^{N-1}(r_N)\phi(r_N)$$

where s is the number of samples of v which are obtained by random sampling of unit vectors. r_N is the largest value of r in these samples. ϕ is the distribution density function of Φ.

The expected value of r_N is

$$E(r_N) = \int_0^1 r\psi_N(r)dr = 1 - \int_0^1 \Phi^N(r)dr$$

and is expressed by the inequality

$$E(r_N) > 1 - \left[\frac{K-m-1}{2cN}\right]^{\frac{2}{K-m-1}} \Gamma\left(\frac{K-m+1}{K-m-1}\right)$$

where Γ is the gamma function. The right side gives a close approximation to $E(r_N)$ for K-m>>1 and s >> 1 as follows

$$E(r_N) \approx 1 - \alpha N^{-\frac{2}{K-m-1}}$$

where α is the abbreviation of the coefficient given in the previous equation. The expected rate of the squared output error decreasing by one hidden unit added is expressed by

$$E\left(\frac{\left\|P_c' y\right\|^2}{\left\|P_c y\right\|^2}\right) = 1 - E(r_N) \approx \alpha N^{-\frac{2}{K-m-1}}$$

The squared output error $\left\|P_c^{(m)} y\right\|^2$ for m hidden units added one by one can thus be estimated as

$$E\left(\frac{\left\|P_c^{(m)} y\right\|^2}{\left\|P_c y\right\|^2}\right) \approx \alpha^m \prod_{j=1}^{m} N^{-\frac{2}{K-j-1}} \approx N^{-\frac{2m}{K}}$$

Therefore, the number of required mapping and demapping layer units is estimated approximately as

$$m \approx \frac{K \log(\|P_c y\| / EG)}{\log N}$$

where EG is the criterion of the allowable output error.

3.3.2 Principal Component Analysis (PCA)-Based Method

The principal component analysis (PCA) (Jolliffe, 1986) models the variability in a data set by identifying the dominant directions in which it varies. This technique is quite useful for data compression where the multi-dimensional data is mapped into lower dimensions with a minimal loss of information. This is done by developing a set of basis vectors based on the dominant directions and by viewing data observations not only as a set of individual or univariate measurements, but as a multivariate observation whose behavior can be observed in terms of the new basis vectors. The PCA implementation is presented below.

Let X represents an $n*m$ data matrix (n is the number of observations, m the number of variables). PCA optimally factorizes X into two matrices, T, called the scores matrix ($n*f$) and P, called the loading matrix ($m*f$), plus a residual matrix E ($n*m$):

$$X = TP^T + E$$

where f is the number of factors ($f<m$). The condition of optimality on the factorization is that the Euclidean norm of the residual matrix, $\|E\|$, must be minimized for the given number of orthogonal factors. To satisfy this criterion, the columns of P are the eigenvectors corresponding to the f largest eigenvalues of the covariance matrix of X.

PCA linearly maps data from $\Re^m$ to $\Re^f$. Take $P^T P = I$ without loss of generality, the mapping has the form:

$$t_i = y_i P$$

where y_i represents a row of Y, a single data vector, and t_i represents the corresponding row of T, or the coordinates of y_i in the feature space. The loadings P are the coefficients for the linear transformation. The information lost in this mapping can be assessed by reconstruction of the measurement vector by reversing the projection back to $\Re^m$:

$$y_i' = t_i P^T$$

where $y_i' = y_i - e_i$ is the reconstructed measurement vector. The smaller the dimensions of the feature space, the greater the resulting error.

Due to the capability of data compression without losing significant amount of information, the mechanism of PCA is also applied into the framework of neural networks to reduce dimensionality and produce a feature space map resembling the actual distribution of the underlying model parameters (Kramer, 1991; Dong and McAvoy, 1996).

Because of the property of the PCA in data compression, it is applied in this study to estimate the number of the bottleneck layer in neural networks. Estimated by the PCA-based algorithm, the bottleneck layer of the NLPCA architecture has a relatively small number of nodes. Network connections through this layer retain most of the information

presented by the signals when the signals pass through the network layers during the network training.

3.4 Neural Network Regularization

The risk of overfitting noisy and collinear data is of major concern in neural network design. A model with an excess number of free coefficients tends to generate mappings which have a lot of curvature and structure, as a result of overfitting to the noise in the training data. Similar behavior also arises with more complex non-linear neural network models. Regularization techniques control the effective complexity of the model and encourage smoother network mappings. Regularized neural network models reduce overfitting, provide stable result, and the generalization ability is improved.

Methods of regularizing neural networks have been widely studied. Larsen *et al.* (1996) proposed an adaptive regularization method by minimizing the error on a validation set. A gradient descent scheme is developed to optimize the regularization parameter adaptively. Similar work has been done by (Hansen, *et al.*, 1994; Larsen, *et al.*, 1996; Hansen, *et al.*, 1994; Petersen, *et al.*, 1996; Larsen, *et al.*, 1994). Levin *et al.* (1994) propose a principal components pruning (PCP) method to tune the neural network parameters based on principal component analysis of the node activations of successive layers of the networks. Similar to truncated singular value decomposition (TSVD) technique detailed in Section 3.6.2, the PCP removes the eigennodes which have no effect on increasing prediction error on a validation set. This method can be done off-line.

Several regularization techniques were applied in this study, including weight decay, cross validation and singular value decomposition.

3.4.1 Weight Decay

Regularization smoothes the weight matrices and makes the solution stable; in other words, makes the solution robust to noise in the data. The regularization technique adds a penalty term Ω to the objective function to give

$$J = E + \upsilon\Omega$$

Here E is one of the standard functions, such as sum squared error (SSE) or mean squared error (MSE); the parameter υ controls the extent to which the penalty term Ω influences the form of the solution. Training is performed to minimize this error function J. A function that provides a good fit to the training data will give a small value for E; while one that is very smooth will give a small value for Ω.

The most common neural network regularization method is weight decay (Press, 1992). In the form of weight decay, the penalty function is the sum of the squared of the adaptive parameters in the network.

$$\Omega = \frac{1}{2}\sum_i w_i^2$$

where the sum runs over all weights and biases.

3.4.2 Singular Value Decomposition (SVD)

A problem lurking in linear regression is the problem of collinearity of a data matrix, in which two or more independent data vectors are linearly correlated with others to a high degree. This makes the solutions of regression tasks non-unique, which leads to instability and unreliability of the solutions. The employment of the SVD technique makes the solution stable by removing noise components. This technique removes noise from a signal and does not introduce bias to the solution.

As described in Section 3.2.4, the output layer of the neural networks has a linear activation function, which linearly maps the output vector of the final hidden layer and the desired output. The neural network training process adjusts the weight parameters (weights and biases) for this connection node by repeatedly propagating the prediction error back through the previous hidden layers. Due to the possible collinear ill-conditioned training data matrix, the resulting neural network weight parameters will be weighted heavily towarding the few dominating variables, which have the largest variations. The neural network models with the non-smooth weight matrices generate inconsistent results. The SVD technique smoothes the weight matrix by removing the minor principal components due to noise in the signals.

The SVD process extracts the components from the variables. The minor components, which are associated with small eigenvalues, are considered as noise information carriers. These minor components are then discarded to remove the noise information. The SVD not only removes the collinearity of the data matrix but also serves as a noise-filter (see

Appendix D). The mathematical representation of the SVD technique is detailed as follows (Masters, 1993).

Given a weight matrix W, target vectors Y, and input vectors X, a standard regression equation can be written:

$$XW=Y$$

We can solve for the weights using the general least square solution:

$$W=(X^TX)^{-1}X^TY$$

In this solution the mean squared error is minimized. However, the problem of singularity of X^TX or near singularity arises. In practice, the vectors in X are linearly correlated to some degree, indeed some of them may be highly correlated, which causes the matrix X^TX to be ill-conditioned. A small change in the vector X results in huge change in weight matrix W. The model created by the least squares estimation may fit the training data very well, but provides very poor generalized results due to non-smooth weight matrix.

The solution to this problem is to solve for the linear output weights using the SVD method. The input matrix X is broken down into its singular value composition given by:

$$X = USV^T$$

$U = nxm$ matrix of principle components (m is the number of signals)

S = *mxm* diagonal matrix of singular values

V = *mxm* matrix of right singular values (orthonormal matrix)

The final weight values are found by reconstructing the above S matrix:

$$W = V(S')^{-1}U^T Y$$

In this study, a truncated SVD technique (TSVD) is applied for reconstruction. In TSVD, only the most relevant information is retained to compute the weight parameters. The least important information, which is stored in the smaller singular values $s_i (i = k,\ldots,m)$, is discarded because it is most likely due to noise. The amount of noise that is removed from the solution to the system is determined relative to the largest weight expected in the network. This is achieved by setting the smaller singular values $s_i (i = k,\ldots,m)$ in singular value matrix S to be zero. The value of k is chosen such that the remaining non-zero larger singular values $s_j (j=1,\ldots,k-1)$ keep most of the information (98% in this study) for good estimation. That is, k is chosen such that

$$\frac{\sum_{i=1}^{k} s_i}{\sum_{j=1}^{m} s_j} = 0.98$$

A reconstructed singular value matrix S' contains only the most important information for variable reconstruction. Minor principal components containing noise information are discarded.

By using the SVD technique, the network training process was greatly accelerated by solving for the output layer weights instead of using iterative training. Not only does this method reduce training time and obtain optimal weight parameters, but the weights are also vastly superior to what would be attained by random iterative methods. Also by removing the noise information, the possibility of neural network overfitting is limited.

Another purpose of using the SVD technique is that since it linearly regresses the network model, the weights in the network linear layer can be quickly calculated; and neural network retraining is not needed. Therefore, on-line system adaptation can be realized.

3.4.3 Robust Training Technique

Another regularization technique is the robust training. A robust network is one that provides the corrected value of an input that contains an error or missing data, without disturbing the output estimates of the other sensors in the network. Robustness with respect to faulted inputs is not necessarily a property of neural networks, since feed forward networks generally have poor extrapolation properties. Sensor failures have no precedent in the training set; therefore, the network's estimate to an erroneous input is unpredictable. An error in a single sensor may be detected, but at the risk of compromising the remaining signals in the network, and it may not be possible to distinguish the faulty sensor. The robust training technique improves the stability of the network performance and makes it possible to identify the faulty sensor.

When training a neural network, the output Y is trained to emulate the input X. A robust training set uses a modified input set X' which contains artificial faults. The input set is constructed as $X' = X + \delta I_j$ $j=1, ...,m$ (m is number of input signals), I_j is the jth column of the identity matrix, and δ is a modification factor. The modified input set X' has noise introduced in one of the signals, while the other signals remain normal. The noise is introduced into remaining signals consecutively, so that a corruption is introduced systematically into all of the signals, but not into more than one signal at a time. For each original training example, each sensor is corrupted several times using different random values of δ ranging between 1% to –1% of the signal operating range. Using the corrupted input set X', a robust network will be trained to produce the non-corrupted output.

3.4.4 Cross Validation

Usually, the plant operating signals contain a small amount of noise (typically 2 to 3 percent) from the sensors and other electronic equipment. If a network is trained to a low error value, it tends to model the noise in the data and not the overall functional relationship behind the data. This is known as overfitting the training set, and creates a network that has very poor generalization abilities. In order to avoid the overfitting problem, a cross-validation technique is employed for network training. In this technique, the data were divided into training and validation sets, the network is trained on training data and the prediction is made on validation data. When the prediction error begins to increase the training is stopped. Both the training and prediction errors are measured by the objection function J defined in Section 3.4.1.

3.5 Sequential Probability Ratio Test (SPRT)

The decision logic module implements the SPRT initially developed by Wald (1945) and later applied for sensor status checking (Upadhyaya, *et al.*, 1987; Uhrig, *et al.*, 1996; Erbay and Upadhyaya, 1997; Hines, *et al.*, 1997; Wrest, 1996; Olvera, 1993; Eryurek, 1994). This module uses the residual between the sensor value and model prediction as the input, and provides the condition of the sensor. Rather than computing a new mean and variance at each sampling time, the SPRT continuously monitors the sensor's performance by processing the residuals.

When a sensor is operating correctly, the residual should have a mean of zero, and a variance comparable to that of the sensor (due to the filtering characteristics of an AANN). If there is a sensor drift, the residual mean shifts thereby increasing the likelihood ratio. This ratio is a measure of how different the residual is from zero. If the likelihood ratio increases above a certain predefined boundary value specified through the false and missed alarm probabilities, the residuals are more likely to be from the faulted distribution than from the unfaulted distribution. The sensor is classified as a faulty sensor. When the likelihood ratio is less than the boundary value, the sensor is good. If a sensor is determined to be faulty, the likelihood ratio is reset to zero and the calculation to determine the status of the sensor begins again.

3.5.1 Decision Algorithm

This SPRT-based method is optimal in the sense that a minimum number of samples are required to detect a fault existing in the signal. The decision rule used for this test is Wald's two-sided criterion. This technique can be used for detection of sensor bias

degradation and for detection of sensor noise degradation with different concerns. Assuming that the residual signals are uncorrelated and have a Gaussian distribution, the likelihood ratio is calculated as:

$$\lambda_k = \lambda_{k-1} + \frac{m_1}{\sigma^2}\left(r_k - \frac{m_1}{2}\right)$$

where λ_k -- likelihood ratio at time k

m_1 -- the degradation mean under the degradation hypothesis H_1, which is sensor specified and determined by the calibration specifications.

$r_k = x_k - \hat{x}_k$ -- the residual between the sensor measurement and the estimate.

σ -- the normal variance of s under the normal hypothesis H_0, which is obtained from the normal condition (no fault) test data.

3.5.2 Decision Rule

The two-sided rule was originally introduced by Wald and is detailed as follows.

The decision boundaries A and B are:

$$A = \ln\left(\frac{\beta}{1-\alpha}\right) \qquad \text{and} \qquad B = \ln\left(\frac{1-\beta}{\alpha}\right)$$

where α = probability of false alarms; β = probability of missed alarms. It is important to obtain a low false alarm probability so that the users gain confidence in the system. The typical values 0.01% for α and 10% for β are adopted in this study.

Normally, the values of the probabilities are specified by the applications. The discussions on the parameter setting is given in Section 3.5.3.

When $\lambda_m < A$, sensor is OK; $\lambda_m > B$, sensor is degraded.

3.5.3 Parameter Selection

The parameters that need to be defined for implementing the SPRT are the missed alarm probability, the false alarm probability, a measure of noise in the signal (variance), and the faulted mean.

The false alarm probability is defined as the probability of classifying the sensor to be good but the sensor is indeed bad. The missed alarm probability is the probability of claiming a drifting sensor but it is in good condition. The criterion to set up the values of the probabilities is to keep the false alarm probability low, while set the missed alarm probability a little high. This setting is to avoid true false alarms being missed, while to allow some intermittent alarms to be ignored. Usually, the values of the missed and false alarm probabilities are specified by the applications. In this research, the values were set such as there is a 0.01% chance that the bad sensors will be missed, and a 10% chances that the good sensors will be classified to be faulty. The missed alarm probabilities are critically important for monitoring the sensors in control channels, especially the sensors in nuclear power plant control system.

The value of the faulted mean depends on the designed detectable sensitivity of the monitoring system. In this study, it is set to be 5% of the average individual variable operating level, that is,

$$m_1(i) = 0.05 \cdot \overline{X}(i)$$

where i is the number of signal channels to be monitored; $\overline{X}$ is the average operating level for each channel.

3.6 Summary

This chapter introduces the individual modules of the sensor validation system to accomplish the sensor monitoring tasks. Methodologies on automated network architecture design were developed. A NLPCA typed neural network architecture, that is proved to be a universal approximator, is selected as the process model. The number of network hidden nodes are estimated by the beta method as well as the PCA-based technique. Several network regularization techniques to stabilize network model performance are combined for network training. Also the SVD technique is used for network retuning when plant operating conditions change. The SPRT method statistically determines if the residuals are from the normal distribution to detect if the sensor is faulted. Parameters in methods on determining the number of network hidden nodes are to be selected in Chapter 5. Also the regularization parameter of weight decay is selected in Chapter 5. In order for AANN models to handle variables without computational difficulty for plant-wide sensor monitoring, data pre-processing is necessary. In the next chapter, Chapter 4, issues related to the data pre-processing are discussed.

Chapter 4

Data Processing

This chapter details the data pre-processing required for the neural network architecture. This procedure is needed to create a proper and meaningful data set for neural network training and validation; also to minimize any computational difficulty network training may face. In order to accomplish this procedure, the tasks that should be performed are detailed in the following sections.

4.1 Fundamental Goal

The fundamental objective of this portion of the research is to determine the signals that can be monitored by this system and, furthermore, to group signals to achieve optimal neural network performance.

When developing a sensor validation system, for plant-wide sensor monitoring, a list of process variables is constructed. Among these process variables, some of them either are calculated for performance monitoring, or have little variances. The calculated variables do not require monitoring, while the variables with little variances cannot be monitored due to poor correlation relationship with others. Therefore these variables are excluded from monitoring. Of the remaining signals, although some of them have large variations, but have little or no correlation with others. These signals provide little or no information in predicting other signals and cannot be estimated well by others; therefore they cannot be predicted well by using autoassociative network structure. The resulting monitorable variables modeled by the neural networks are put into groups of relatively small size. In

this study, it is less than or equal to 25. The grouping is accomplished such that the variables within each group are correlated with each other to a relatively large degree (greater than or equal to 0.5). The reason to choose correlation threshold to be 0.5 is that it has been found that the variables having correlation coefficients less than 0.5 were not estimated well using autoassociative network architecture. These grouped variables were then used as inputs to the neural network estimation models. Corresponding neural network models to each group of variables are then established. In this dissertation, the correlation analysis and the NLPLS-based technique are used for variable selection.

4.2 System of Interest

In this research, Tennessee Valley Authority (TVA) fossil power generating system Unit #9 provided sufficient data for the sensor monitoring system design. Figure 4.1 demonstrates the interrelationships among three sub-systems of this power generating cycle: boiler system (steam generator), turbine/generator and condenser/heat rejection (ASME PTC PM-193). Because only a few signals from the condensing system were provided, the research concentrates on two major sub-systems: the boiler system and the turbine system. A total of 224 variables were provided by the TVA for this research.

4.3 Variable Selection

The operating data were provided for the periods listed in Table 4.1. The data was sampled at one minute intervals. Some of the data are used for neural network model construction, while the other data are for model prediction performance validation.

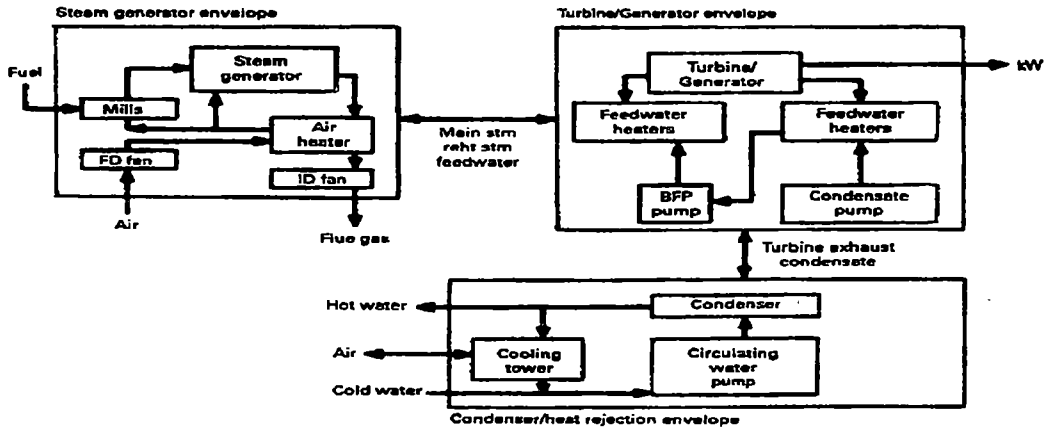


Figure 4.1 Fossil power generation system (ASME PTC PM-193)

Table 4.1 TVA Kingston Fossil Plant Operating Data

Operation Period	Data Status
01/05 – 01/22-98	for network model training and validation
03/26 – 04/10, 08/01, 08/08, 08/16, 08/22-98; 02/07-02/11, 06/07-06/11, 08/17-08/22-99	for network model prediction performance checking

For an ANN-based monitoring system to function correctly, the neural networks must be trained on the fault-free data. It is assumed that the instruments were operating within specifications over the network training period. In essence, no adjustments were made to sensors or instrument channels during this period. Therefore, in order for the network to be valid in the operating region, the data in the period of 01/05/98-01/22/98 were used for neural network training, while the remaining data were used for neural network model validation.

Of the recorded 224 variables, some are unchanged with time and 83 of them are calculated. Since the unchanged ones are not correlated with others and thus cannot be

inferred by other variables, the neural networks cannot predict them. The 83 variables calculated by the mathematical or empirical models do not need to be monitored by neural networks if the input signals are monitored. Thus, the variables that are either unchanged or calculated are excluded from neural network modeling. After removing these unmonitorable variables, the remaining 136 variables are selected for monitoring by this validation system and are listed in Table A1 in Appendix A.

However, some measured variables, which change with time, are still unmonitorable due to lack of information provided by other signals in regard to sensor validation purpose. This is explained as poor correlation between the predictive and response variables, which are the network inputs and target outputs, respectively. A non-linear partial least squares (NLPLS)-based technique, which incorporates a neural network architecture into a conventional partial least squares (PLS) model (Geladi, 1986; Hoskuldsson, 1988), as well as a correlation analysis-based technique is employed to identify these variables. A prediction error threshold is set for all the variables. The variables with prediction error from the NLPLS model larger than the threshold are classified as unpredictable and are excluded from network prediction. The NLPLS finds a few components called latent vectors that carry most information of the variables to form the network input and output, and the network is trained to map the relationship between the latent vectors. A series of single-input-single-output neural networks is constructed to model each variable. By using this type of architecture, variable grouping is not needed. The limitation is that the non-linear correlation of the variables cannot be high; otherwise, this method fails.

In this study, most of the variables have fairly high linear correlation relationships (correlation coefficient is greater than 0.5), therefore, the NLPLS model provides a reference for variable selection. The variables selected from the NLPLS model are listed in Table A2 in Appendix A. In this NLPLS model, 25 latent vectors were extracted from the signals, which is comparable to the results obtained from the correlation analysis detailed later. The response variables which have an average prediction error greater than 3% are classified to be unmonitorable. This criterion allows 105 variables to be classified as monitorable.

However, since the NLPLS modeling does not require the variables to be correlated well, the variables selected by this technique may not be appropriate for AANN modeling in regard to sensor validation because the AANN requires the inputs to be reasonably correlated well. Therefore, a method using correlation analysis is also applied to select the variables for AANN input in comparison with the NLPLS-based selections. The correlation analysis method removes the variables, which have little correlation with others, from modeling. The correlation-analysis-based variable grouping algorithm is detailed in Section 4.4.3 and is sketched in Figures 4.3a and 4.3b. This method determines that 36 variables, which are listed in Table A5 in Appendix A, cannot be monitored.

4.4 Variable Grouping

With a large number of signals/variables to be monitored in complex systems, variable grouping is needed. The purpose of this task is to construct fairly small neural network models with 20-30 input variables to avoid the computational difficulties associate with

large neural networks. The neural networks require that the input variables be correlated with each other at some degree (usually greater than 0.5); such that they can be easily trained. This task is to be done by performing the correlation analysis for all the variables. As mentioned in the previous section, this method is also considered for variable selection. It is previously found that in an AANN model, a variable poorly correlated with others cannot be predicted well (Hines, *et al.*, 1998). Therefore, this correlation analysis method optimally groups variables for neural network models to achieve better performance.

4.4.1 Non-Linear Partial Least Squares (NLPLS)

NLPLS is a technique which incorporates feedforward neural networks into the conventional partial least squares (PLS) model to provide nonlinear functionality (Kramer, 1992; Qin and McAvoy, 1992). By using the universal approximation property of neural networks (Haykin, 1994), the PLS modeling method is generalized to a nonlinear framework. The NLPLS model captures the nonlinearity and keeps the PLS projection to attain a robust generalization. By embedded nonlinear neural network architecture, the NLPLS model is able to capture the nonlinear relationship between response and predictive variables. In this research, this method along with a correlation analysis is used to select the variables which are monitorable.

The PLS model analyzes the co-variability between predictive and response variables. It identifies the dominant directions in which the response variable varies as well as the directions which have the highest correlation with the predictive variables. The PLS model is shown in Figure 4.2.

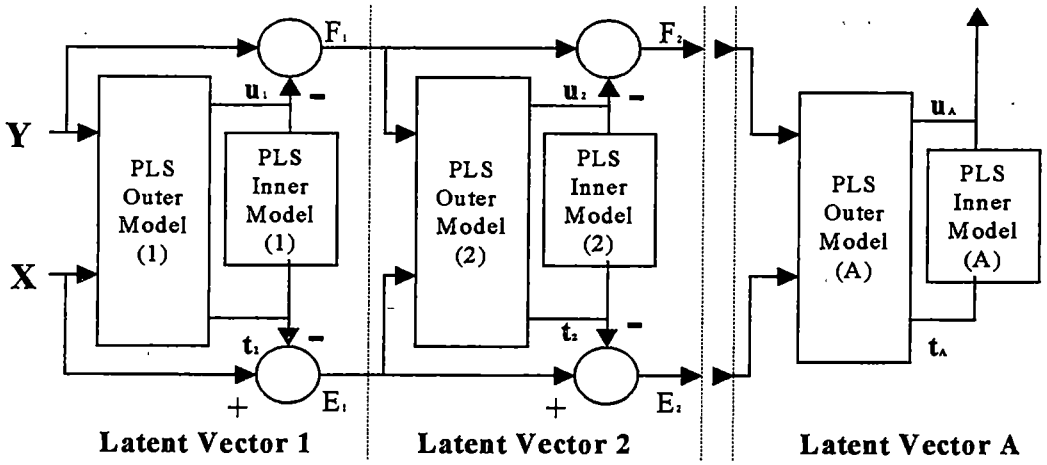


Figure 4.2 PLS model (Qin and McAvoy, 1992)

The PLS outer model, which decorrelates the input and output matrices and transforms them into latent vector space, reduces the dimension of the input X and output Y into lower dimensional latent vectors U and T, and completely de-correlates the input variables. The inner model, which can be either a linear regression model or a single-input-single-output neural network model, maps the relationship between T and U. This mapping can either be linear such that a PLS model is formed, or a neural network embedded nonlinear mapping such that an NLPLS model is constructed.

The mathematical model of the PLS is presented as:

$$X = \sum_{i=1}^{n} t_i p_i^T = \sum_{k=1}^{A} t_k p_k^T + E$$

$$Y = \sum_{i=1}^{n} u_i q_i^T = \sum_{k=1}^{A} u_k q_k^T + F$$

where n is the number of input variables, A is the number of latent vectors.

Since the columns of latent vector U are orthogonal to each other, the neural network in the inner model has no collinearity problem. The neural network is easily trained and the solution to the network weights is stable.

The PLS model iteratively extracts information from both the input and output data to the outer models until the residuals E and F contain little information (i.e., $E \approx 0$ and $F \approx 0$). These latent vectors are then used to predict the output with a very small discrepancy. The procedure on determining the latent vectors T and U is detailed in Appendix E. Mathematically this can be written as:

$$\hat{X} = \sum_{k=1}^{A} t_k p_k^T$$

$$\hat{Y} = \sum_{k=1}^{A} u_k q_k^T$$

Since data in practice are usually nonlinear in nature, it is desirable to have an approach which can model systems with some degrees of nonlinearity and still attain the robust generalization property of the PLS approach. Keeping the outer relation from the outer model in linear PLS so as to have the robust prediction property, a nonlinear PLS (NLPLS) approach using neural networks as the inner regressors is proposed:

$$u_h = f(t_h) + r_h$$

where $f(\cdot)$ stands for the nonlinear relation represented by a neural network.

The power of PLS is that it is a supervised method that extracts information from the variables by considering the input X together with the predicted output Y. It does not simply reject the higher principal components (PCs) which have small variability in the data set like the principal component regression (PCR). Therefore, the information that is predictive of outcome Y but is relatively insignificant in terms of its variance will not be lost. Combined with the neural network non-linear modeling capabilities, the NLPLS models the linear relationship among variables and the non-linearity as well. Therefore, this technique provides a better representation on modeling the output Y than simple linear regression techniques. It needs to be mentioned that since this supervised neural network has a single-input-single-output, the training process is non-exhaustive and easy. This may avoid the overfitting problem, make the solution stable, and make automation possible.

4.4.2 Correlation Analysis

Correlation analysis reveals the linear relationships among variables. Previous studies indicate that the degree of correlation between input variables is relatively significant for an autoassociative neural network to function as a complex system monitoring device (Upadhyaya, 1992; Kramer, 1991). In this research, the robust neural network training paradigm detailed in Section 3.4.3 is applied. Changes in one input variable due to drift, channel deterioration, or failure will not significantly change the corresponding value of the network output because the output is related to all the other correlated input variables through a large number of paths and weights.

The signal correlation coefficients range from –1 to +1 with the highly correlated parameters at the boundaries of the range. A value close to ± 1 means that there is a positive or negative linear relationship between the change of one parameter with respect to another. If a signal shows virtually no change with time (i.e., constant), its correlation with others would be near zero.

4.4.3 Variable Grouping Algorithm

The correlation analysis is employed to uncover the correlated relationship among all monitored variables. However, the spurious spikes in the variables will significantly influence the outcome of correlation analysis. Therefore, the spikes are filtered out using median filters with window size of three before performing the correlation analysis. A series of variables X are then selected which are highly correlated with a specified response variable Y to start variable grouping. Two algorithms are studied in this research.

Both algorithms group the variables around the most interested signals as grouping base. These interested variables are from a leak detection system in a fossil power plant of TVA, which is used for detecting water leak in the boiler system, and are listed in Table 4.2. The locations of the sensors are indicated in Figure 4.3.

The first algorithm is a method called variable exclusion, in which a variable can only be in a certain group. For instance, if signal A is bound in Group A, it cannot be in any other groups. While in the other algorithm, called variable inclusion, a variable can be in different groups. For example, signal A can be in group A, and also can appear in Group

Table 4.2 Interested Variables from Leak Detection System

Location	# Sigs	Signal Name	Unit
1	1	Combustion Air Flow A1	Pct
2	2	Combustion Air Flow B1	Pct
3	24	Unit Gross Generation	MW
4	25	Station Service Load	MW
5	34	First Stage Pressure A	psig
6	35	First Stage Pressure B	psig
7	37	Deaerator Pressure	psig
8	46	ID Fan A Suction Pressure	InH2O
9	47	ID Fan B Suction Pressure	InH2O
10	48	RH Furnace Press After Econ	InH2O
11	124	Superheat Outlet Temperature #1	DegF
12	125	Superheat Outlet Temperature #2	DegF

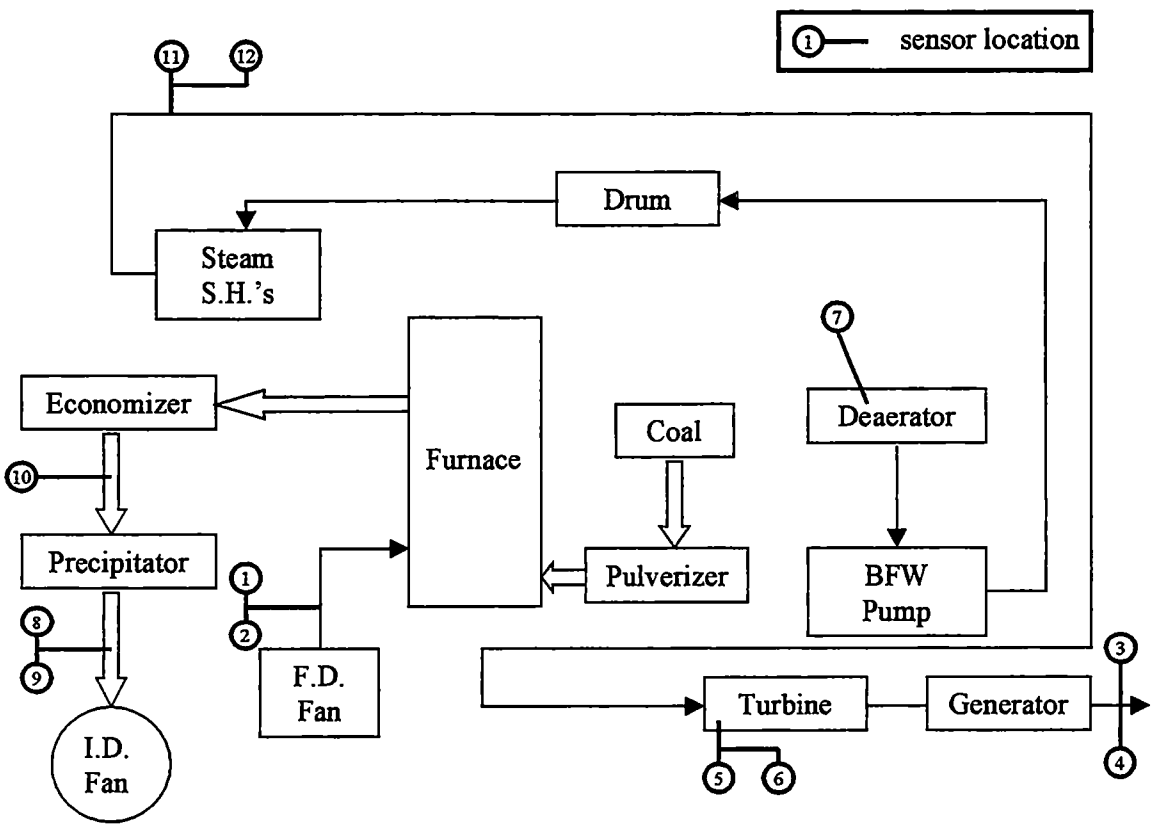


Figure 4.3 Kingston power plant Unit 9 Schematic

B or any other groups. Both algorithms start with an interested variable Y_i from Table 4.2, and variables X that are well correlated with Y_i are pre-selected to form the first group. Then the pairwise correlation analysis is performed on X within each group to remove the variables having little correlation with others. This process is plotted in Figures 4.4 and 4.5. Due to computational difficulty, the number of variables in a group is limited. In this study, a maximum of 25 variables is allowed in each group. The variables are so grouped that these variables are correlated well with each other. This will aid in possible and easy network training. Another interested variable Y_j $(j \neq i)$ is selected to pre-construct another group. This procedure is repeated until all groups are formed and all monitorable signals are covered in the groups. It can be seen from the block diagrams of these algorithms in Figures 4.4 and 4.5 that the difference between the two algorithms is that a variable can be in different groups for the latter algorithm, which allows the variables share important information for better prediction.

The algorithms are applied to group the data of January 1998. The grouping results are tabulated in Tables A3 and A4 in Appendix A. By using the proposed grouping method, the pairwise variables are correlated well (above 0.5). A demonstration of the correlation for Group 4 (NN_4) is tabulated in Table 4.3.

The results from both algorithms show that there are some common variables that cannot be inferred by others and therefore are unpredictable, like variable #8 (Coldwell Tank Makeup Flow). They also indicate that the exclusion method throws out a lot of important information for predicting some variables, such as variable #61 (Unit CEMS

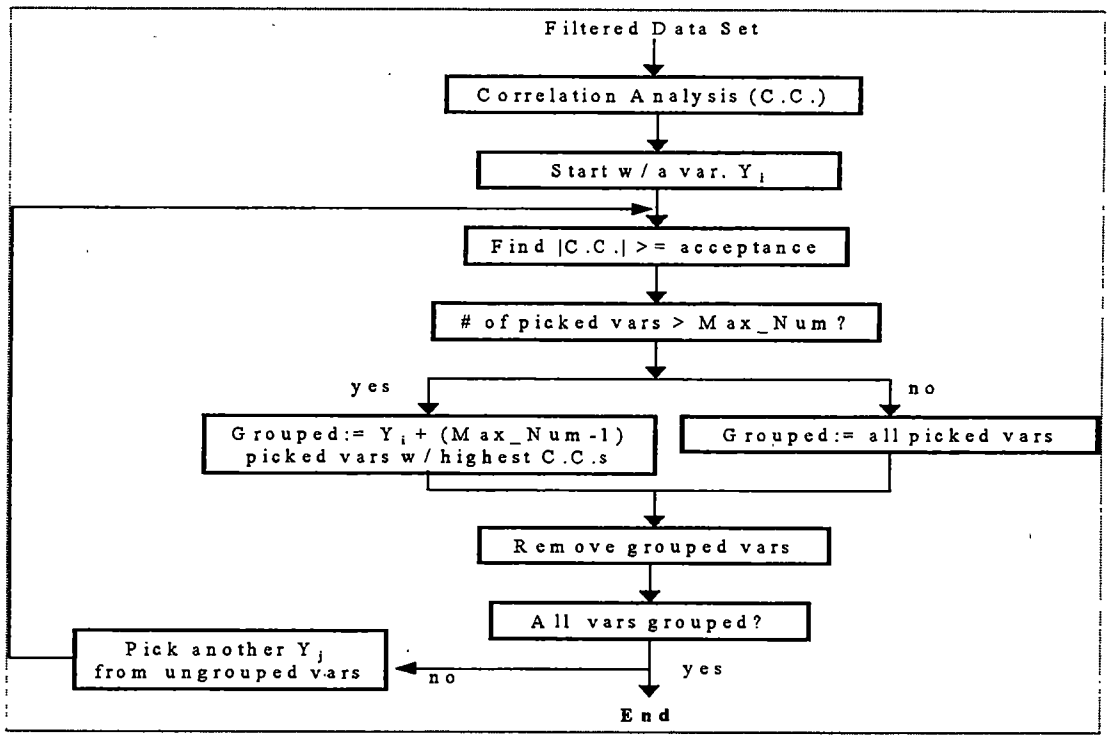


Figure 4.4 Variable exclusion algorithm

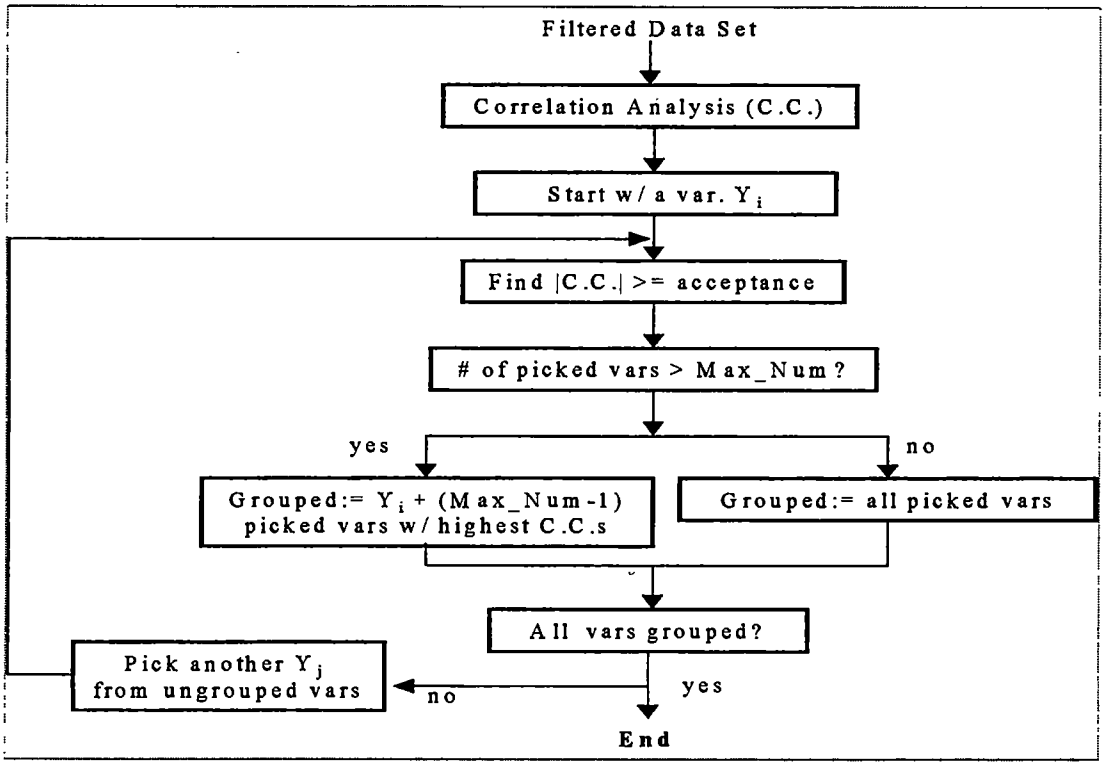


Figure 4.5 Variable inclusion algorithm

RH NOx PPM). This variable cannot be monitored since the group has only three variables according to the exclusion method but it can be monitored using the inclusion method. This is because the other variables, which provide useful information on predicting this variable, are thrown out and become unavailable using the exclusion method. Therefore, the exclusion method is improper. The 36 unpredictable variables from the inclusion method are listed in Table A5 in Appendix A.

Table 4.3 Correlation Coefficients for Group 4 (NN_4)

Sig #	75	85	87	119	123	124	125	136
75	1.00	1.00	0.90	0.69	0.66	0.51	0.65	-0.55
85	1.00	1.00	0.89	0.68	0.66	0.51	0.65	-0.54
87	0.90	0.89	1.00	0.77	0.75	0.66	0.75	-0.66
119	0.69	0.68	0.77	1.00	0.99	0.93	0.99	-0.57
123	0.66	0.66	0.75	0.99	1.00	0.95	1.00	-0.56
124	0.51	0.51	0.66	0.93	0.95	1.00	0.95	-0.59
125	0.65	0.65	0.75	0.99	1.00	0.95	1.00	-0.56
136	-0.55	-0.54	-0.66	-0.57	-0.56	-0.59	-0.56	1.00

To finalize the grouping procedure, the variables listed in Table A5 should be removed and the remaining variables, which are considered monitorable, are to be regrouped. The final grouping results are presented in Tables A6 through A20. This system is able to monitor 100 signals from a total number of 136 signals, or 74% of the signals can be monitored. There are a total of 15 neural network modules for monitoring these variables based on the grouping result. Some variables that provide most useful information in predicting others are included in multiple groups.

4.5 Network Input Scaling

Virtually all networks are to be trained more efficiently if their inputs and outputs are restricted to a "reasonable" range (Bishop, 1995). This entails pre-processing, or scaling, the input data before it is presented to the network, and post-processing the outputs of the network to restore the output values. There are two reasons why processing the data is advantageous, or in some cases, required.

1. The non-linear activation function used in network architecture requires a bounded input value to prevent premature saturation of the network, which greatly slows or prohibits training. In addition, the sigmoidal function tends to emphasize the importance of intermediate output values, while obscuring fine differences when the outputs are near their extreme high and low values. This means that predictions that approach the limits of the network's output will be less accurate than the intermediate predictions (Masters, 1993).

2. A typical plant system has parameters that differ in magnitude and/or deviation greatly, such as temperature contrasted to pressure. Depending on the units in which the parameters are expressed, they have values that may differ by several orders of magnitude. The network would need to find a solution for the weights in which some weight values had markedly different values from others. This would greatly affect the network's stability and generalization ability.

3. Plant signals have wide operating ranges from one sensor to another, for example, a flow meter may normally operate between 1500 and 2000 KLb/hr levels; while a sensor

monitoring the RH Furnace pressure may operate between 2 and 3 InH_2O levels. A PCA technique applied to this type of data may totally ignore the importance of the pressure signal due to the small variance. However, the pressure signal should be considered equally important as the flow signal. Therefore, the unit variance scaling of the data before performing the PCA technique is needed. Otherwise, the resulting PCA outcomes could be invalid.

From the above discussion, it is apparent that the input parameters should be scaled to a narrow range near the center of the activation functions. The input parameters to the network are linearly scaled using a so-called *z-score* scaling method, which scales the data to values that have a mean of zero and a variance of unity. The slope and intercept parameters are used to restore the outputs back to their original values.

4.6 Summary

This chapter discusses the issues related to the data pre-processing. NLPLS is an inferential neural network model. It does not require variables be well-correlated. Therefore, the variables selected from this model may not be proper for AANN architecture, which requires variables to be well-correlated. The result shows that the NLPLS determines 105 variables are monitorable in comparison with 100 variables from correlation analysis.

Chapter 5

Sensor Monitoring Validation System Design

This chapter applies the techniques detailed in the previous chapters to a TVA fossil power plant for plant-wide sensor monitoring validation system design. The first few sections discuss the issues of neural network structure design and training, including the selection of the number of the network hidden nodes without network training, as well as the network regularization. The later sections concentrate on the issues of detecting sensor drifts and replacing faulty sensors with their best estimates from the neural networks.

5.1 Neural Network Structure and Design

The objective of this study is to design a neural network that has the best generalization capability with a simplest structure. A simpler network has less possibility of overfitting, less computational difficulty due to complex structure, and stabler network performance. The ideal network for this application would have the ability to provide dependable sensor estimates, even though the sensor may be faulty. Once the best network architecture is determined, training techniques were incorporated to optimize the performance of the network.

The structure of the neural networks depends directly on the complexity of the problem. In this study, an automation procedure of network design was developed. The following sections discuss the issues related to this procedure.

5.1.1 Neural Network Structures

In this dissertation, an NLPCA-typed AANN was used for signal modeling. The NLPCA has been proven as a universal approximator (Cybenko, 1989; Kramer, 1992; Haykin, 1994). In this structure, the bottleneck layer performs as a PCA feature extractor, which produces the features to retain the maximum amount of information from the original input variables, for a given degree of data compression.

The three hidden layers form a feature detection architecture in which the bottleneck layer plays the key role in the identity mapping. Essentially, the bottleneck layer functions as a NLPCA filter that uses a lower dimension to explain a maximum amount of information while eliminating minor or noise components. The outputs of the bottleneck layer are non-linear principal components, which have a clear interpretation in theory (Bishop, 1995). The first two layers in the network map from the input data space to the non-linear principle component score space, and the last two layers map from the non-linear principle component score space to the output data space. The mapping-bottleneck-demapping combination forces the network to develop a compact representation of the training data that better models the underlying system parameters.

The number of hidden nodes in a feedforward neural network is significant in characterizing the performance of the network. It greatly influences network capacity, generalization ability and output response. However, the number of hidden nodes should not be too small to learn the underlying functions or too large to provide poor generalization of the output. Therefore, an optimal number of hidden nodes exists, which depends on the complexity of a given learning task.

The beta method-based technique explained in Section 3.3.1, which automatically estimates the number of the NLPCA mapping and demapping layer nodes, is developed. A PCA-based method is also employed for determining the number of the bottleneck layer nodes of the NLPCA.

Another type of neural network is the time delayed neural networks (TDNN). In a TDNN, besides the current signal values, previous ones are also used to provide temporal information for network to get a better estimation. A simple block diagram of TDNN with four variables x_1, x_2, x_3, x_4, plus two delayed signals in x_2, and one in x_3 is plotted in Figure 5.2. This diagram shows clearly that the delayed information is used to predict the present values in a TDNN.

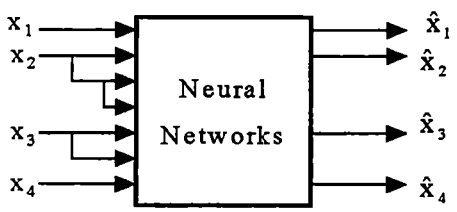


Figure 5.2 Simple block TDNN structure

In order for a TDNN to achieve its best performance, delayed signals should provide useful information, that is, a good correlation between the current data and the previous data must exist. Otherwise, the TDNN cannot guarantee any improved performance compared to an AANN. (*Hines, et al.*, 1998) Also the increased collinearity in its input data matrix could degrade its generalization ability.

The proper time lag can be determined by a cross-correlation analysis. This technique determines the amount of information provided by delayed signals in estimating a

response signal. Figure 5.3 shows a cross-correlation relationship between Combustion Air Flow A1 and other signals in the group of NN_1 for the time span of four hours. It demonstrates that there is no higher correlation at any lagged time than that at the current time. This indicates that there is no additional information provided from the previous measurements to improve the network performance. There is virtually no helpful information provided for network prediction. Therefore, the TDNN does not perform better than the AANN. A faster sampling rate probably provides a better dynamic representation of the physical process. It was found that all the monitorable signals have the same behavior, implying that the TDNN is not helpful for getting an improved performance.

5.1.2 Determination of Neural Network Parameters

One issue related to the neural network structure is the selection of the number of the hidden nodes. Generally, the number of hidden layer nodes of a neural network is related

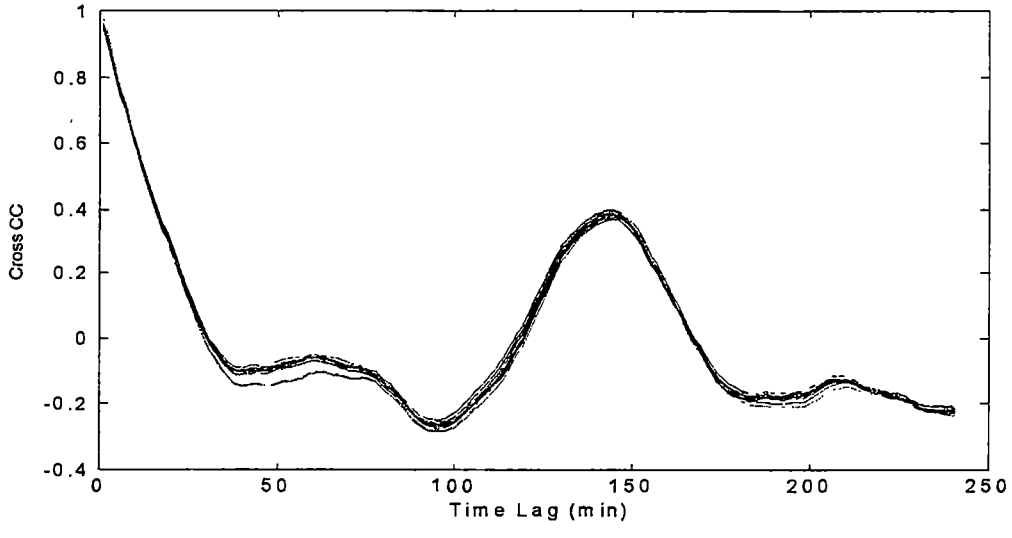


Figure 5.3 Cross correlation between Combustion Air Flow A1 and others

to the complexity of the process. A network without a sufficient number of hidden nodes would have a reduced accuracy, due to its limited representational capacity with too few degrees of freedom in adjustable parameters. If there are too many nodes, the network is prone to overfitting, or learning the stochastic variations in the data rather than the underlying functions. One of present approaches for determining the number of mapping layer nodes is the simplified cross-validation scheme. In such an approach, the number of hidden nodes is determined such that the network has a minimum test error. Therefore, this method requires the network be trained and determines the solution on repeated trials.

The result of variable grouping implies that the neural network structure for each variable group should not be identical. A neural network modeling with more variables may be complicated in its structure than with fewer variables. Due to the various complexity of the neural network structures, there is no unique solution to the network parameters. The best solution must be chosen upon individuals' discretion. To automate this procedure, some techniques were developed.

5.1.2.1 Modified Beta Method

As detailed in Section 3.3.1, the beta method has its advantage in determining the hidden units without training the networks. However, it introduces a parameter N, the total number of candidates that are randomly searched for the optimum hidden unit, which is not known *a priori*. Furthermore, the number of hidden nodes depends on the number of training samples. These factors make the original beta-method inapplicable. In this

dissertation, a modified beta method is developed such that it becomes data-independent and no prior information is needed.

In this algorithm, the parameter, K, which is the number of training patterns, and $\log(N)$, which is the natural logarithm of the random search steps, are closely related. For example, with more training patterns, the random search for the solution takes more steps. Based on this observation, the method is modified by replacing this ratio with a parameter C. Through this replacement, the modified beta method becomes

$$m \approx C * \log(\|P_c z\| / EG)$$

The root least square error of $\|P_c z\|$ is calculated through a linear regression method using the SVD method. The selection of C depends on the complexity of the problem and is originally determined by experiments, referring to (Fujita, 1998). When applied for automated design, an expression for estimating C should be derived.

It is observed that the determination of C is based on the roughness of the training surface, a simple way of expressing this is by checking the number of principal components of the data matrix. That is,

$$C = f(\textit{data matrix})$$

The function $f()$ is designed to be the number of principal components kept such that 95% of the variation in the data set is retained. For most network groups, the value of C

is around 5 and the networks provide good results with the calculated number of hidden nodes.

5.1.2.2 PCA-Based Method

In the NLPCA-type neural network, since the bottleneck layer functions as a non-linear principal component analysis filter, it requires as many nodes as there are non-linear factors in the parameters that are modeled. Therefore, the number of bottleneck nodes can be statistically determined by the evaluation of the principle components in a data set.

It has been discussed in Chapter 3 that the PCA technique is applicable to estimate the number of network nodes in the bottleneck layer. In retaining most of the variation in the data set, the PCA finds a small number of variables through the principal components (PCs) retaining most information of the dynamic process. By performing on the covariance matrix $X'X$ of the data set X, PCA finds a set of PCs. The number of PCs to be kept is determined by the amount of underlying information excluding noise in the data.

Several methods on selecting number of PCs are reported in (Jolliffe, 1986). In this dissertation, a method, to select a cumulative percentage of information which it is desired that the selected PCs should contribute, is applied. This method is explained below.

The number of PCs m is selected such that the cumulative percentage reaches a desired value. The value of m should be chosen that a maximum amount of noise is removed but a minimum amount of true signal is lost. In this research, since the sensor measurements

are noisy, therefore m is selected such that the cumulative percentage of variation of data is equal or greater than 95%. The definition of cumulative percentage of variation cp accounted for by the first k PCs' is,

$$cp_k = 100 \sum_{j=1}^{k} l_j \Big/ \sum_{j=1}^{p} l_j$$

where l is the singular values of input covariance matrix $X'X$; p is the dimension of $X'X$. This expression reduces to $cp_k = \frac{100}{p} \sum_{j=1}^{k} l_j$ in the case of a correlation matrix.

Choosing a cut off value cp^* based on the desired cumulative percentage threshold, and retaining m PCs corresponding to cp^*, where m is the smallest integer, k, for which $cp_k > cp^*$, provides a rule which preserves in the first m PCs most of the information in the data set.

5.2 Network Training

The methodologies for neural network training are discussed in this section. Among the topics covered are, the removal of the outliers in the training data; the training algorithm selection; the accelerated training methods; the training error goal determination; the robust training technique; and the iterative validation method.

5.2.1 Outlier Removal

The neural network's performance mainly depends on the robustness and quality of the collected data for training and estimation. Therefore any corrupted data results in questionable network performance. One common type of corrupted data that is easy to

detect is the spurious spikes or outliers. Because of their extremely large magnitudes relative to the rest of the signals, outliers tend to negatively influence the output of the network.

In order to minimize the influence of outliers on the network's performance, the training data set are pre-processed with a median filter to filter out the spurious spikes of the data usually caused by the noise in the data collection system. A zero phase shift median filter was used on the input parameters, which gives the median value for a window size of 3. After filtering, the data was scaled by the z-score method to prevent network from premature saturation and to accelerate the training process.

5.2.2 Training Algorithm Selection

The major criterion for selecting the training paradigm is the size of the training set. Three backpropagation methods were investigated: gradient descent with momentum & adaptive learning rate (GDX) (Vogl, *et al.*, 1988), conjugate gradient with Fletcher-Reeves updates (CGF) (Scales, 1985), and Levenberg-Marquardt (LM) (Hagan and Menhaj, 1994).

Although gradient descent backpropagation (GDX) is computationally simple and reliable, it has very slow convergence properties. Backpropagation is used to calculate the derivatives of performance (*perf*) with respect to the weights and bias variables *W*. Each variable is adjusted according to gradient descent with momentum:

$$\Delta W = mc \cdot dW_{old} + lr \cdot mc \cdot \frac{dperf}{dW}$$

where *mc* is the momentum constant and *lr* is the learning rate, which are to be selected.

It can be seen that the change of weights and bias matrix W depends on the parameter settings of *mc* and *lr*. Without these parameters being properly selected, the network training may not converge as expected. The conjugate gradient backpropagation with Fletcher-Reeves updates has faster convergence properties than the GDX algorithm. In the CGF algorithm, each variable is adjusted according to the following:

$$dW = -gW + dW_{old} \cdot Z, \qquad Z = \frac{norm\_sqr}{norm\_sqr_{old}}$$

where gW is the gradient, *norm_sqr* is the norm square of the gradient.

In contrast with the GDX, the adjustment of the variables only depends on the gradient and the change of gradient and no chosen parameters are required. This makes the algorithm easier to implement.

The Levenberg-Marquardt (Bishop, 1995) algorithm is an optimization method that trades off between the reliable convergence of the gradient descent paradigm and the quick convergence of Newton's method. Implementing this method requires the calculation of a pseudo Hessian matrix. This matrix incorporates second order information that aids convergence, but is based on assumptions of being near the minimum and the error surface having a quadratic function. The mathematical manipulations of the pseudo Hessian matrix make the LM training paradigm very memory intensive and slow for large training sets and network architectures.

Due to the relatively large size of the neural networks required for a plant wide monitoring system, the Levenberg-Marquardt training algorithm is impractical due to its memory intensive requirements. We also want to avoid any training parameter choices, so the GDX algorithm is excluded for consideration. By comparing the three training algorithms, The conjugate gradient backpropagation with Fletcher-Reeves updates was selected as the optimal training algorithm.

5.2.3 Ill-Conditioning and Regularization

By modeling sensor signals in an autoassociative way, a signal is predicted through its relationship with other correlated variables. The problem with using these variables as predictors is that they are not only highly correlated with the response variable, but they are also correlated with each other. If the degree of correlation is extremely high, meaning they are almost linear dependent, the data matrix becomes ill-conditioned and the problem of drift detection becomes ill-posed (Hadamard, 1923; Hines, et al., 1999; Gribok, et al., 1999). Hadamard defined a well-posed problem as a problem which satisfies the three following conditions:

- The solution for the problem exists.
- The solution is unique.
- The solution is stable or smooth under small perturbations of the data; i.e. small perturbations in the data should produce small perturbations in the solution.

5.2.3.1 Ill-Conditioned Problem

If any of these conditions are not met, the problem is termed ill-posed and special considerations must be taken to ensure a reliable solution. The degree of an ill-posed matrix is measured by the condition number *Cond* which is defined as:

$$Cond = \frac{\max(s)}{\min(s)}$$

where s is the singular values of the data covariance matrix. A matrix with condition number less than 10 is considered to be in good condition, less than 100 to be in fairly good condition, larger than 100 is classified to be ill-conditioned.

The condition numbers of the input matrices of all networks are listed in Table 5.1. The results indicate that all network input matrices are ill-conditioned, which causes inconsistency in the neural network model.

Table 5.1 Condition numbers for each network model

Network Tag	Condition Number
NN_1	3.9615e+004
NN_2	3.8640e+004
NN_3	3.8900e+004
NN_4	1.8049e+004
NN_5	5.5154e+004
NN_6	2.0478e+004
NN_7	1.0565e+005
NN_8	3.9260e+004
NN_9	2.5291e+005
NN_10	4.0335e+004
NN_11	622.2410
NN_12	720.4839
NN_13	1.5410e+004

The large condition number of the network input matrices means that a network trained with this set of data would generalize badly on future data. The application of regularization techniques resolves this ill-posed problem by minimizing an objective function J when it is embedded in neural network training. Several network regularization techniques have been discussed in Section 3.4. This section primarily discusses a simple form of weight decay.

Regularization combines the objective error to be minimized (the mean-squared error of the network, *mse*) and the prior knowledge of the process (mean-squared weights of the network, *msw*) which will stabilize the network estimation. By nature, a process or system achieves its equilibrium state with minimum energy. In neural network business, the energy is represented in terms of weights. Therefore a term of *msw* is embedded into the objective function for minimization. A regularization objective function in terms of mean squared of the network error and the mean squared of the network weights has the form:

$$J = \gamma mse + (1-\gamma)msw$$

where

$$mse = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2, \quad msw = \frac{1}{n}\sum_{j=1}^{n} w_j^{\,2}.$$

γ is the performance ratio and needs to be determined, $0 \le \gamma \le 1$. γ=0 means the objective function is totally dependent on prior knowledge, while γ=1 means the

objective function is unregularized. If the value of performance ratio is too large, the network is under-regularized, the solution tries to provide more accurate estimate but has large uncertainties. On the other hand, the neural network is over-regularized when the performance ratio is small. The over-regularized network model tends to provide a more stabilized result but introduces a larger bias, and the resulting estimate has a large discrepancy from the true value. In this study, to balance these two situations, a value of γ is set to be 0.75, which indicates that 75% of mean squared error and 25% of mean squared weights are contributed to the objective function. By doing so, the model tends to predict more accurately while does not loss consistency.

5.2.3.2 Evaluation of Regularization

To evaluate the consistency of the sensor validation system under small perturbations of the data, a bootstrap technique (Efron, 1982) was used. The bootstrap technique is a statistical method used for evaluating the stability of the regression coefficients or fitting values. For a training data set of size n, the bootstrap technique samples values from both predictor and response variables at random with replacement, thus providing a bootstrap sample of size n with some original values duplicated and some missing. This bootstrap sample is used to map the predictor variables onto the response variables using the same fitting procedure as for the original sample. When the method is repeated a number of times, the bootstrap procedure produces a set of fitted values whose variability can be estimated and whose sampling distribution can be plotted.

To demonstrate the power of regularization for stabilizing the model development, a linear regression model was used to detect a Venturi meter drift in a nuclear power plant by predicting the feedwater flow rate (Gribok, *et al.*, 1999). Variables that are highly correlated with the feedwater flow rate were engaged for modeling. This shows an ill-conditioned problem, and the models from different trials provided inconsistent results. The flow rate drift at a selected check point was between 20 to 60 KLb/hr, while the actual drift at that point was 40 KLB/hr. A bootstrap technique was applied to illustrate the inconsistency, and the probability density function (PDF) of the model predictions is plotted in Figure 5.4. After regularization, the models all provide a consistent value of about 40 KLb/hr at different trials, which is the true drift of the feedwater flow rate, see Figure 5.5.

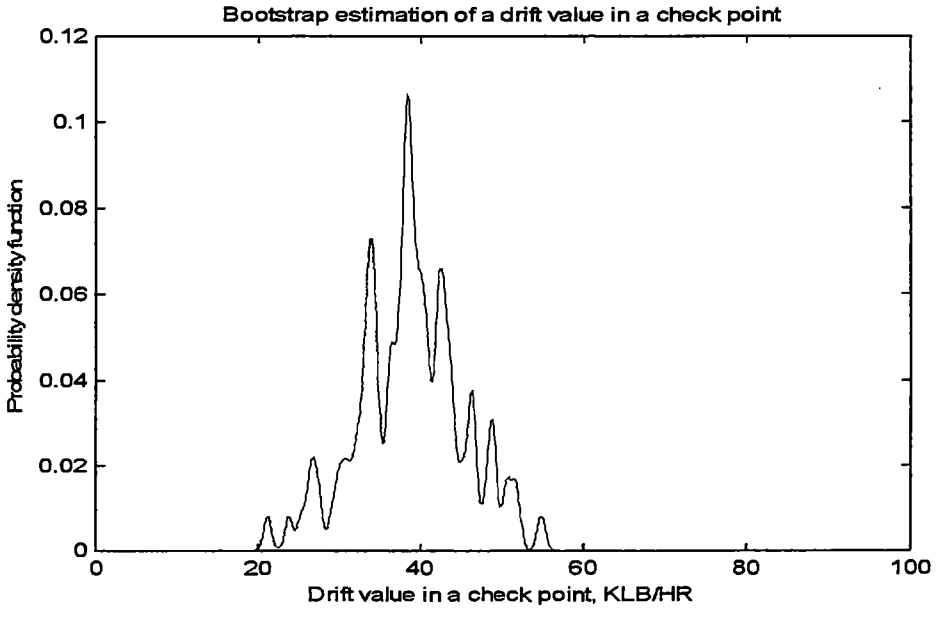


Figure 5.4 Instability of drift estimation due to perturbations (Gribok, *et al.*, 1999)

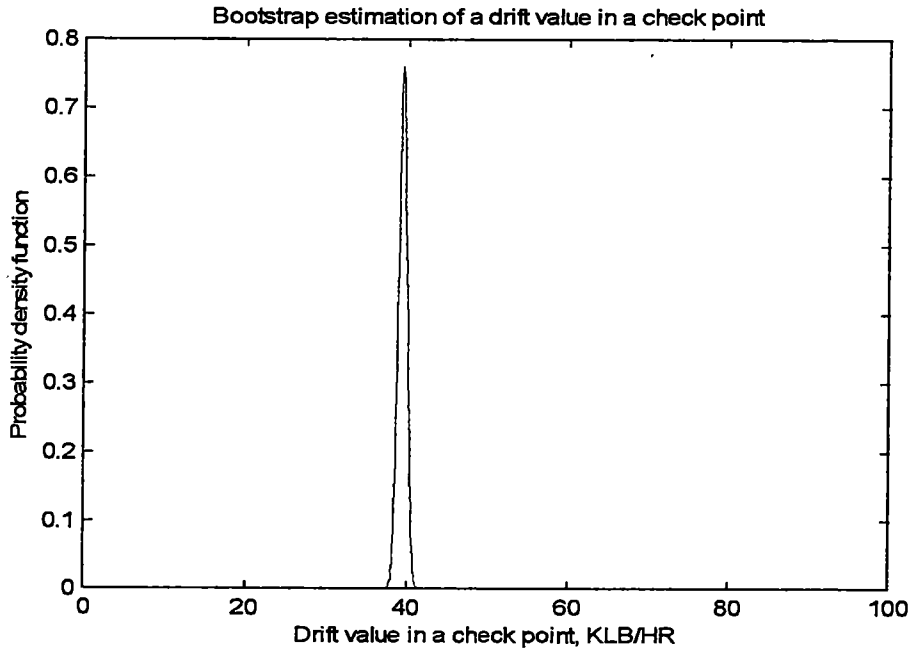


Figure 5.5 Regularized estimated drift value at the check point (Gribok, *et al.*, 1999)

5.2.4 Training Paradigm

The initial training/validation data set consisted of about 5% of the total patterns randomly selected from the twelve-day operating period listed in Table 4.1 (01/05-01/16-98). The network is trained in a cross-validated fashion explained in Section 3.4.4. When the network prediction error of the validation set increases, the network stops training, the test data patterns that have the largest errors are added to the training data set until both training and testing error goals are achieved. Figure 5.6 shows the network training paradigm developed for this research. Through this training process, the data is ensured to cover the entire operating input space of the process, including any process transients. A robust training paradigm is used to force the network to rely on all the sensor information to estimate each specific sensor's value.

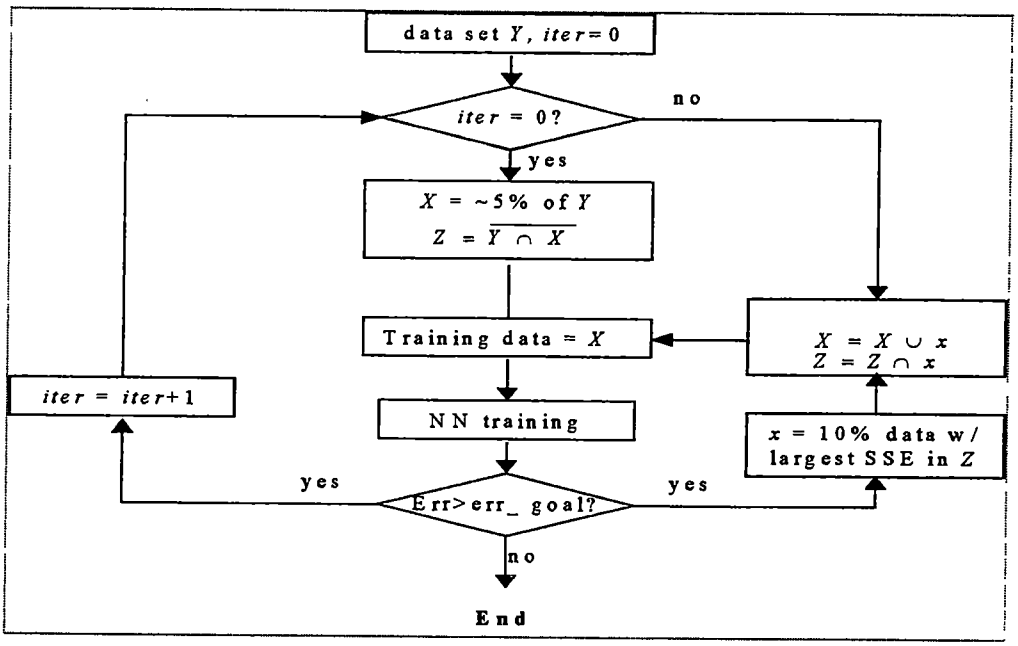


Figure 5.6 Network training paradigm

Validation involved selecting and constructing several sets of data to exercise the input space and evaluate the network performance. This is an iterative process since poor performance on the validation set may require merging it with the training set and retraining the network. Thus an iterative training and testing procedure is used to achieve optimal performance.

5.2.5 Stopping Criteria

In order to avoid the overfitting problem discussed in Section 3.4.4, the network training should be stopped when the prediction error of validation data set begins to increase, although the error of training data set continues to decrease. This is known as cross-validation for network training. In this technique, the data were divided into training and validation sets, the network is trained on training data and the prediction is made on

validation data. Both the training and prediction errors are measured by the objective function J defined in Section 5.2.3.

5.2.6 Network Performance Evaluation

Network performance was evaluated with a test set of normal operating data and a test set with an artificial error placed in one of the signals. The mean percentage error (MPE) between the actual sensor values and the network's estimated values was used for performance comparisons.

5.3 Issues on Sequential Probability Ratio Test (SPRT)

5.3.1 SPRT Detection Problem

The ideal network residual represents the pure noise if there is no sensor degradation. Therefore, it is applicable for on-line detection. In practice, the neural network may learn some degree of noise, even the spurious spikes in signals, and incorporate it into neural network models. Therefore the residual contains not just the noise and may deviate from the normal distribution. As stated in Section 3.7, the SPRT algorithm requires normally distributed residual signals. Therefore, the SPRT may produce some system intermittent alarms due to this deviation.

Since the SPRT accumulates the residuals for updating the likelihood ratio, these spikes may cause the value of the likelihood ratio to pass the threshold and produce false alarms not because of the signal degradation. The term of (m/σ), which has the equivalent meaning to the signal-to-noise ratio (SNR), has a large impact on the detection decision, and determines the accumulation rate of the likelihood ratio. Spikes in sensor signals

with large (m/σ), which indicates less-noisy and well-instrumented data, will cause false alarms to be produced quickly. This would lead operators to take wrong actions. In order to avoid this situation, a modified SPRT (MSPRT) to eliminate the intermittent alarms was developed.

5.3.2 Modified SPRT (MSPRT)

Since (m/σ) controls the speed of the accumulation of the likelihood ratio, a filter structure is added to reduce the speed to avoid false alarms due to spurious spikes in the residuals. The MSPRT is represented as

$$\lambda_k = \lambda_{k-1} + \frac{m}{G\sigma^2}\left(r_k - \frac{m}{2}\right) = \lambda_{k-1} + \frac{m^2}{G\sigma^2}\left(\frac{r_k}{m} - \frac{1}{2}\right)$$

where G is a gain factor to suppress the accumulation rate. Although the MSPRT may delay the sensor validation system to initiate a false alarm, this will not degrade the SPRT capability for detecting real signal failures. The determination of G is based on the variance of the resulting residuals between estimates and target values of the training data. To avoid spike-caused intermittent alarms due to prediction error, G is chosen to be 9 such that the term of $(\sqrt{G}\sigma)$ represents three standard deviations of the resulting residual. This means that there is a 99% confidence that the training result will not cause intermittent alarms even through the training may not be perfect.

Another modification is on the execution algorithm. Instead of resetting λ_k to zero at each time when λ_k reaches or exceeds the upper bound threshold B, the value of λ_k stays

at the upper bound. The MSPRT will make clearer decisions than the current version of SPRT by suppressing the intermittent alarms.

An example of the comparison of the currently used SPRT and the MSPRT is plotted in Figures 5.7 and 5.8. In this example, the SPRT intends to detect a 2.5% mean drift while a drift of 1% per day was artificially added for 10 days. Figure 5.7 is the residual normal probability plot. It shows from the bottom plot of Figure 5.7 that the distribution of the residual deviates from the normal distribution which is the straight dot-dash line, indicating that the residual is not normally distributed. Figure 5.8 shows that the SPRT initiates a false alarm due to spikes in the residual at 31st minute with 0.02% drift, whereas the MSPRT eliminates the false alarms due to spikes and produces false alarms when the true degradation occurs at 3876th minute with 2.7% drift.

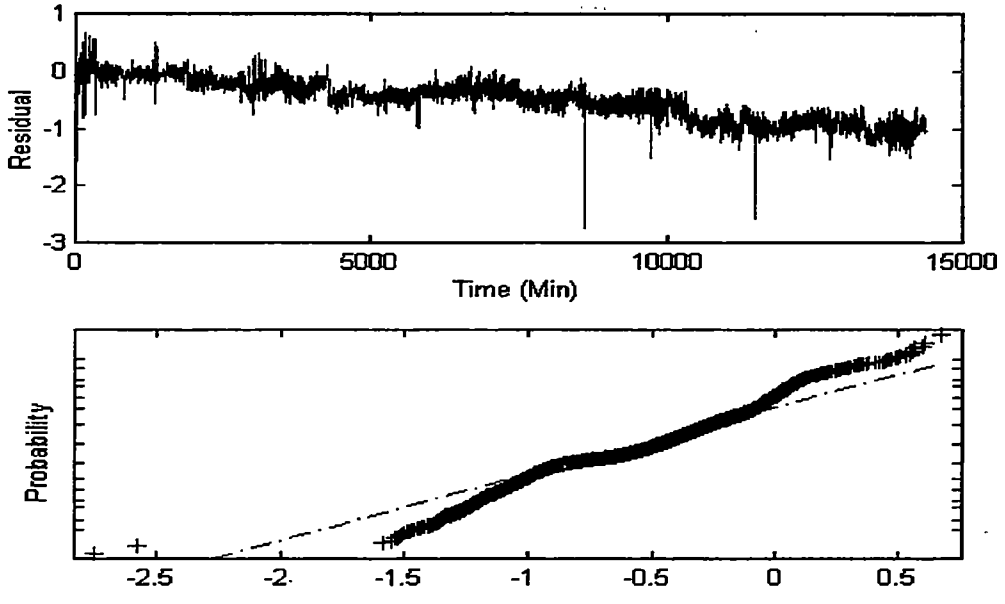


Figure 5.7 Residual normal probability plot

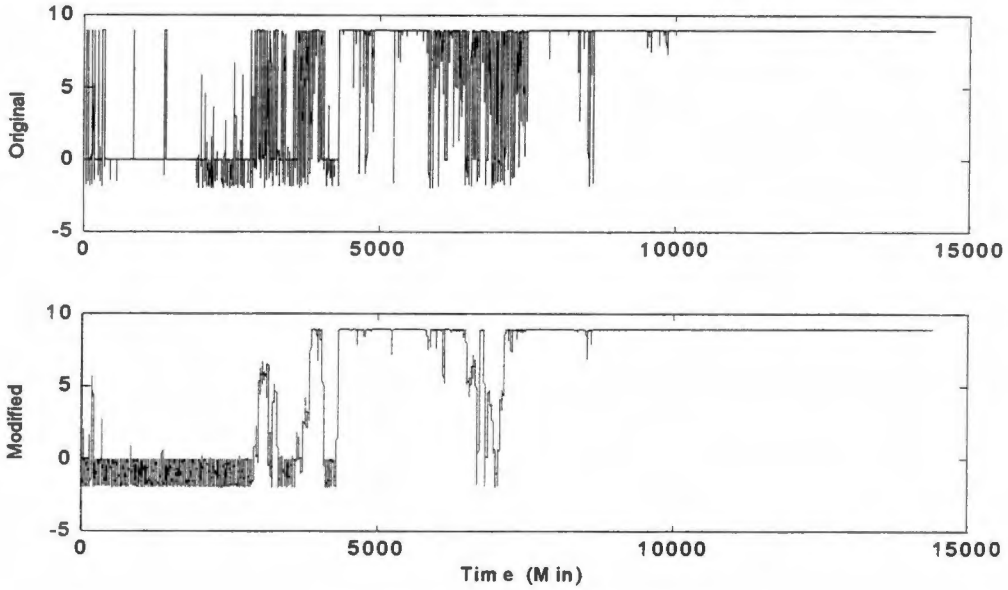


Figure 5.8 SPRT vs. MSPRT

5.4 Summary

In this chapter, techniques discussed in Chapter 3 were applied for the sensor validation system design. Selection on network structures between AANN and TDNN was performed through the cross correlation analysis. For a TDNN to have a superior performance over an AANN, there should be a high correlation between variables at a lagged time. This study shows that no such time lag exists. Therefore, an AANN instead of TDNN was selected as the neural network structure.

Modification on beta method was made to develop an automated methodology to estimate the number of network hidden nodes in mapping and demapping layers. An MSPRT is also developed to suppress the intermittent alarms from SPRT due to prediction error or signal noise. Next chapter presents some results using the developed methodologies.

Chapter 6

Results and Summary

In this chapter, techniques developed in this research were applied to the real operating data from the TVA Kingston Power Plant Unit 9. A bootstrap technique is presented to illustrate the ill-conditioned problem as well as the regularization effect on stabilizing the neural network performance with the presence of the collinear data. The results of sensor validation system performance are presented. Also a sensitivity analysis is performed to verify the robustness of the networks. Network performance is validated with artificially induced drifted signals. In this chapter, all results are plotted in figures in Appendices B and C unless otherwise stated.

6.1 Study of Historical Information

In real processes, historical information of signals may provide better understanding of the process. Time delayed neural network (TDNN) architecture predicts current value using historical information and tends to provide a better result than AANN does. However, for a TDNN to be advantageous over an AANN, the historical information must be useful for network modeling. The determination of usefulness of the historical information is discussed in this section.

A group of 15 signals listed in Table 6.1 is used for neural network modeling to investigate the usefulness of the historical information, in which the cross correlation among these signals is plotted in Figure 6.1. This plot indicates that other than the zero time lag, which is at the current sampling time, there is no higher correlation between

measured steam flow and other signals. It is noticed that although the correlation between the Measured Steam Flow and the Ambient Air Temperature is maximum at a time lag of 3rd minute, there is no truly helpful information from the Ambient Air Temp. in predicting Measured Steam Flow, see Figure 6.2. Therefore, there is no additional information in the delayed measurements or historical information for improving network performance, as described in Section 5.1.1. Therefore, in this situation, the TDNN architecture will not provide better prediction than the AANN architecture. The comparison result in this section illustrates this fact.

This paragraph provides the training results of both the AANN and the TDNN using signals listed in Table 6.1 with regularization are shown in Figures B1 through B4 in Appendix B. In these figures, the red signals represent the sensor measurements, while

Table 6.1 Signals for History Study

Sig #	Tag Name	Signal Name
1	9FAIRA1	Combustion Air Flow A1
13	9FHWPUMPS	Hotwell Pumps Discharge Flow
23	9FSTEAM	Measured Steam Flow – Raw
30	9O2RHA	Excess O2 A in Reheat Furnace
31	9O2RHB	Excess O2 B in Reheat Furnace
34	9P1STSTGA	First Stage Pressure A
38	9PDRUM	Selected Drum Pressure
46	9PIDFASUCT	ID Fan A Suction Pressure
48	9PRHECON	RH Furnace Press After Econ
50	9PSHAHT	SH Furnace Press After HT SH
57	9RHCO2PCT	Unit CEMS RH CO2
73	9TAMBIENT	Ambient Air Temperature
74	9TAPHAAIRIN	Air Preheater A Air Inlet Temp
75	9TAPHAGASIN	Air Preheater A Gas Inlet Temp
119	9TRHABSPR	RH Attemp A Before Spray Temp

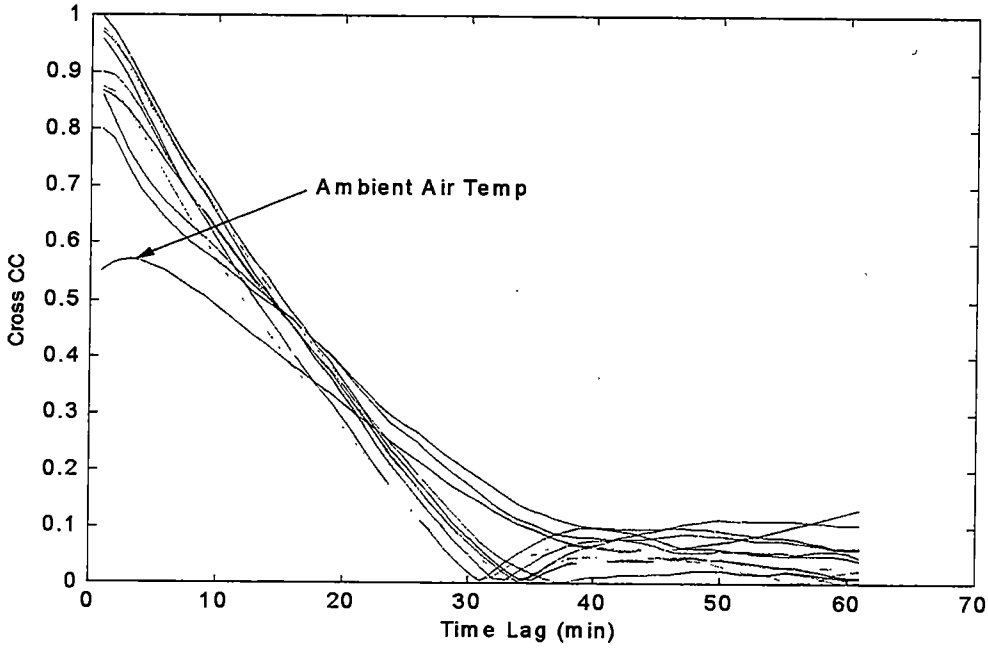


Figure 6.1 Cross correlation between Measured Steam Flow and others

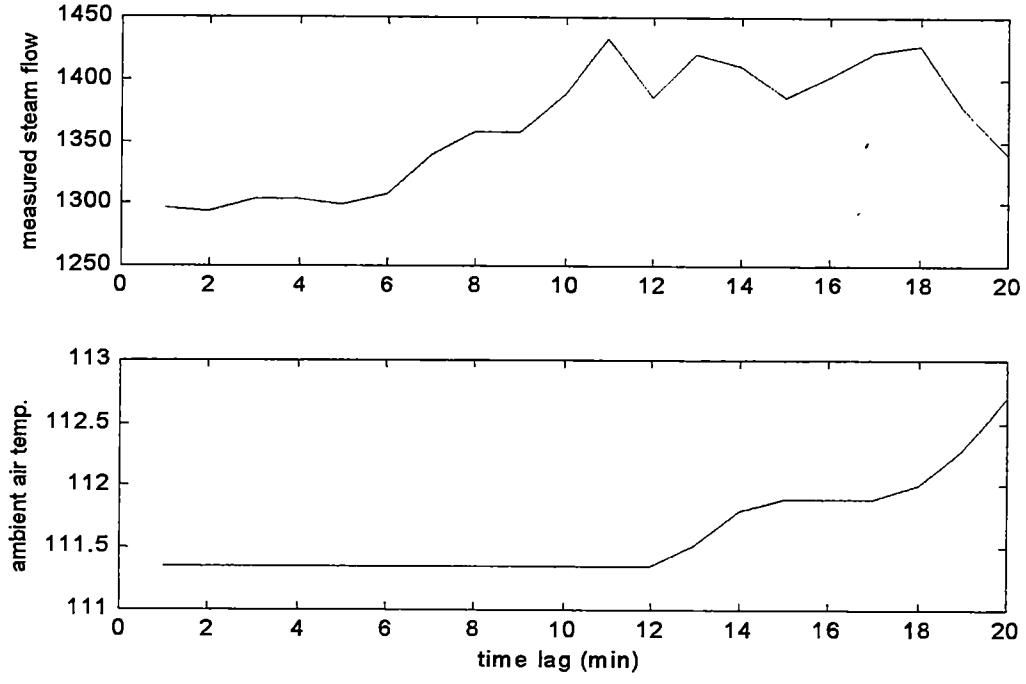


Figure 6.2 Measured Steam Flow and Ambient Air Temp. vs. time lag

the blue ones show the network estimates. The mean percent error (MPE) value is a measure of network performance and is listed in Table 6.2. The plotted residuals indicate that the TDNN does not perform better than the AANN, except only three signals (#30, #50, and #75) are trained slightly better in TDNN than in AANN. For example, the MPE values for Measured Steam Flow and for Selected Drum Pressure with AANN are 0.79% and 0.19%, respectively; while the TDNN provides values of 0.94% and 0.2%, respectively.

It has also been noticed that, in general, the noise level is higher in the TDNN predictions than in the AANN predictions, refer to Table 6.3. This increase in variance is likely due to a problem of collinearity and is presented in several texts on regression (MacGregor, *et al.*, 1991; Qin, 1997). The TDNN has more redundant information and has a more serious collinearity problem; therefore, the variability is increased.

Table 6.2 Demo Group Training Performance

Sig #	MPE in AANN (%)	MPE in TDNN (%)
1	1.1805	1.2037
13	1.2962	1.3488
23	0.7909	0.9379
30	3.1972	3.1630
31	2.0006	2.2110
34	0.8216	1.0072
38	0.1921	0.2023
46	1.2984	1.5036
48	2.3511	2.3739
50	1.6971	1.6866
57	0.6371	0.7030
73	0.8723	1.0351
74	0.5155	0.6236
75	0.3259	0.3222
119	0.3840	0.3998

Table 6.3 Signal Noise Level in Demo Group

Sig #	AANN (%)	TDNN (%)
1	0.1501	0.1058
13	0.0450	0.0650
23	0.0640	0.1080
30	0.2865	0.3732
31	0.2231	0.3210
34	0.0695	0.1144
38	0.0066	0.0097
46	0.1665	0.1478
48	0.1188	0.1810
50	0.0958	0.1584
57	0.0704	0.0764
73	0.0621	0.1206
74	0.0380	0.0479
75	0.0162	0.0232
119	0.0409	0.0495

The result shows that with no useful delayed information, the TDNN performs no better, or even worse, than the AANN due to the collinearlity problem. In this research, it was determined that this situation commonly exists in all the variables to be monitored. Therefore, the TDNN architecture is not necessary for this research.

6.2 Signal Prediction

In this section, the results of network prediction are reported. The results demonstrate the neural networks function not only as a signal predictor but as a noise filter as well. In this section, the results from NN_1 through NN_4 are presented and discussed. The robustness of the system, that is, how well the system resists the abnormal changes in a signal measurement in providing a best estimate, is also evaluated through a sensitivity analysis.

6.2.1 Neural Network Prediction and Noise Reduction

The neural network models built on the data groups developed in Chapter 4 were trained and tested. The robust training technique detailed in Chapter 5 and the TSVD technique of Chapter 2 were applied for network training. These techniques provide the robustness of the networks and reduce the variance of the network predictions. The sensitivity analysis is performed to evaluate the robustness of the network models. Assuming all operating data are fault-free, artificial faults were added to a data set, which is independent of the training data, to verify the validity of the monitoring system. The MSPRT is applied for detecting and locating the sensor faults and provides a reference to the sensor correction/replacement module.

The developed automation techniques were applied for network structure design. Since the variables in the leak detection system are included in networks NN_1 through NN_4, this section concentrates on the results from these 4 networks. By using the modified beta method as well as the developed PCA-based method, the resulting network structures are listed in Table 6.4.

Table 6.4 Neural Network Structures for NN_1 through NN_4

NN No.	# of input/output	# of mapping/demapping layer nodes	# of bottleneck layer nodes
NN_1	25	28	9
NN_2	25	27	8
NN_3	25	31	13
NN_4	8	29	5

It has been verified that since the variables in each group are most highly correlated with each other (refer to Table 4.3); therefore, fewer network hidden nodes generated by the PCA-based technique are needed to retain most of the information for capturing the process dynamics. The network performances of the variables in NN_1 are plotted in Figures C1 through C6 in Appendix C. Also we notice that although there are only 8 variables in NN_4, the mapping and demapping layers still needs 29 nodes due to relative low correlation between these variables with an average correlation coefficient of 0.6.

The results indicate that the neural networks are capable of regenerating the signals. However, not all the signals can be estimated well. Some of them are more accurately estimated than others. The signals having large prediction errors were operating at small magnitudes, such as the Combustion Air Flow. The accuracy of estimation depends heavily on the variable operating range. This is because the z-score scaling method scales all the variables into a comparable range. Thus the scaling coefficients for the variables with lower operating magnitudes (VL) are much larger than those with higher operating magnitudes (VH). In the network training process, all scaled variables are trained down to the same error goal. Therefore, the VLs after being back to the original scale will have larger errors than VHs.

The average percentage errors (APEs) for signals in NN_1 are listed in Table 6.5. Also the noise analysis shows that in general, signal estimates have less noise than sensor measurements. Although the actual noise level (NL) in a signal is difficult to obtain, a comparable result can be generated on the same basis by applying the same filtering structures to both the measurements and the estimates. By the filtering analysis, we can

get a comparable result for both the measurement and the filtered signal on the same basis. In this study, the noise levels in the signals are obtained resulted by applying a median filter with a selected window size. The noise level NL (%) is defined such that it is comparable to the training APE.

$$NL = \sum_{k=1}^{n} \frac{|residual_k|}{x_k} \times 100 = \sum_{k=1}^{n} \frac{|x_k - \hat{x}_k|}{x_k} \times 100$$

where n is the number of sample points; x_k is the kth data point while $\hat{x}_k$ is the filtered kth point.

The results listed in Table 6.5 use window size of 25. From the measured and the filtered signals plotted in Figures 6.3 and 6.4, it is observed that the measured signals have a lot of roughness due to noise. The filter smoothes the signals by eliminating the noises in the signals.

The results indicate that the signals from the turbine system are better estimated than the ones from the boiler system. For instance, the APEs for the combustion air flows A1 and A2 in the boiler system are 3.4% and 3.1%, respectively; while that for measured steam flow in the turbine system is 0.8%. This is compliant with the previous finding that the measurements in the boiler system are noisier and less accurate than those in the turbine system. More importantly, the results indicate that the neural networks acts as a noise filter as well.

Table 6.5 NN_1 Network Training Performance

Sig #	APE (%)	Measured NL (%) (WinSize=25)	Estimated NL (%) (WinSize=25)
1	3.3718	2.3169	1.8555
2	3.0651	2.3119	1.9016
9	0.7828	2.1008	1.9770
10	0.7625	1.9506	1.8333
13	1.5133	2.3918	1.6544
23	0.7687	2.1814	2.0710
24	0.9749	1.9285	1.9012
25	0.6058	1.1405	1.1433
34	0.8802	2.4002	2.2623
35	0.8785	2.4008	2.2623
36	0.5467	2.1269	2.0275
40	0.6807	2.1691	2.0592
41	0.5487	2.1044	2.0250
42	0.5499	2.1670	2.0951
43	0.5801	2.1030	2.1359
44	0.5596	1.6205	1.7052
45	0.5453	2.1438	2.0417
53	2.4688	4.3145	3.1942
88	0.1323	0.3745	0.3719
91	0.1775	0.3091	0.3265
92	0.1326	0.4157	0.4211
103	0.2458	0.4548	0.4744
105	0.2747	0.3938	0.4326
106	0.4006	0.4691	0.4850
110	0.4570	0.4052	0.4471

The results indicate that the signals from the turbine system are better estimated than the ones from the boiler system. For instance, the APEs for the combustion air flows A1 and A2 in the boiler system are 3.4% and 3.1%, respectively; while that for measured steam flow in the turbine system is 0.8%. This is compliant with the previous finding that the measurements in the boiler system are noisier and less accurate than those in the turbine system. More importantly, the results indicate that the neural networks acts as a noise filter as well.

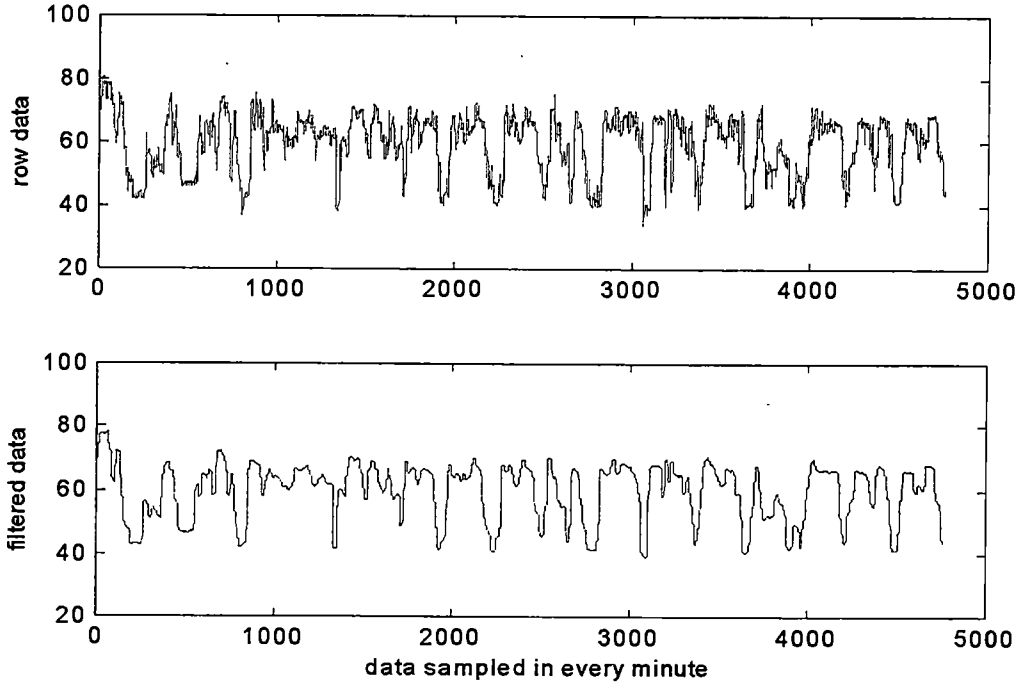


Figure 6.3 Measured and filtered Combustion Air Flow (Pct) (WinSize = 25)

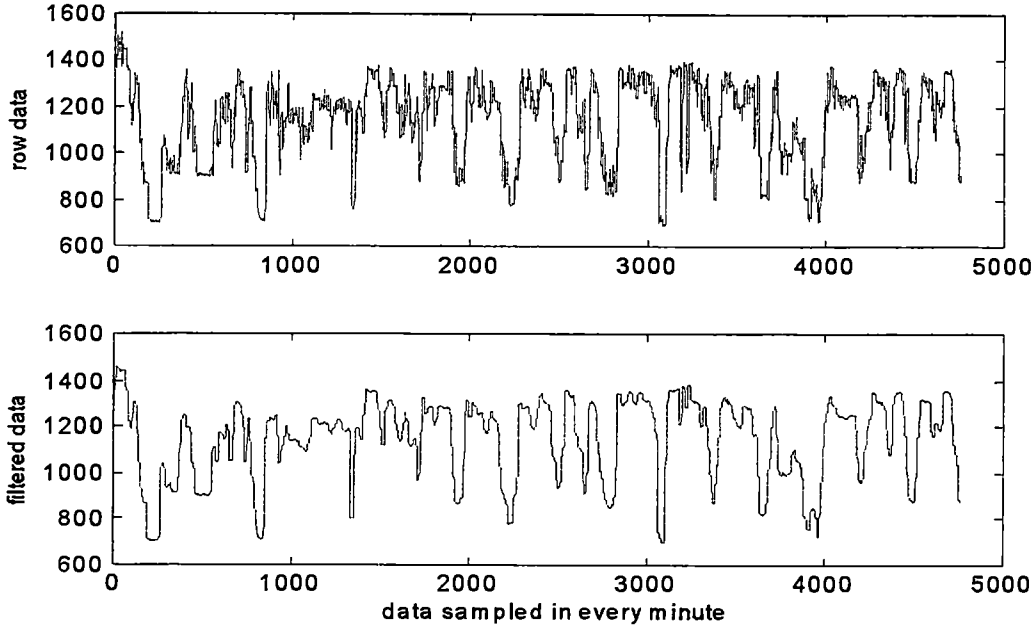


Figure 6.4 Measured and filtered Feedwater Flow (Klb/hr) (WinSize = 25)

6.2.2 Neural Network Generalization

To verify the validity of the neural network models for signal prediction, generalization is performed with the data on which the network has not been trained. In this section, the neural network performance on variables in leak detection system is reported.

The results, plotted in Figures C7 through C18, show that the neural network models designed through the automation procedure are capable of estimating the signals with an acceptable prediction error. It can also be seen that the MSPRT suppresses the intermittent alarms due to prediction errors. The network model predicts very well on some signals, like SH Outlet Temp #1 and #2, with prediction errors of about 0.1%. Both the standard and MSPRTs do not trigger at any time.

Generally, the prediction errors are larger than the training errors due the untrained data. Table 6.6 presents with prediction and the training average percent errors (APEs) for the variables in the leak detection system. It can be seen that the prediction APEs for most variables are larger than the training APEs. However, even through the neural networks have not been trained on these data, they still generalize well, i.e., reproduce the signals with small and acceptable errors. The result shows that the network is not only trained but also able to predict the future data as well if there is no significant change in the signals. If there is a considerable change in the system, the network has to be retrained to capture the system dynamics. This situation will be discussed in Section 6.5.

Another finding from the result is that the scaling method is also a factor to influence the network prediction accuracy. For neural network training, it is difficult to set different

Table 6.6 NN Performance on Variables from Leak Detection System

# Sigs	Training APE (%)	Prediction APE (%)
1	3.2966	3.2193
2	2.9251	3.2901
24	0.8939	1.0322
25	0.7627	0.8203
34	0.8240	0.9784
35	0.8230	0.9781
37	0.8665	0.8652
46	1.9341	1.9343
47	3.3317	3.3259
48	2.2792	2.2785
124	0.1030	0.1030
125	0.0946	0.0946

error goals for individual variables. With signals operating at different levels, larger prediction errors may be produced by the neural networks during training for the sensor measurements with low operating magnitudes. This may cause SPRT trigger early to produce intermittent alarms for these sensor measurements. In order to avoid the intermittent alarms due to prediction errors, large tolerance would be set up for the detection system and thus leads to low sensitivity to faults of these sensors. For example, the network has large prediction APEs on the Combustion Air Flow signals at about 3.2%, while it has small APEs on the power generation signals at about 1%. Therefore, the sensor validation system can detect smaller, or more sensitive to, drifts on sensors measuring the power generation signals than on sensors measuring the combustion air flow signals. After being trained, the network is tested for robustness to assure the network outputs are not affected much by the exterior perturbation. Otherwise, sensor faults cannot be identified.

6.2.3 Sensitivity Analysis

In this section, a sensitivity analysis is performed to analyze the robustness of the network models. A robust network should be insensitive to small disturbances in the process or the instrument channels. The sensitivity analysis provides the changes on each network output by perturbing one of the network inputs with a certain amount of change at a time. If the changes of the network outputs are large, indicating the network is sensitive to external disturbances, then it is not robust. In this procedure, each network input was perturbed by a small amount of disturbances and the changes in the network outputs were analyzed. Figure 6.5 represents the changes in network output in Group NN_1 with each input being perturbed by 5%. The results indicate that with the perturbation in sensor measurement, the network estimate also changes. Some signals from the boiler system, like combustion air flow and hotwell pump discharge flow (signals #1 and #5 shown in Figure 6.5, respectively), have a total change of more than 10% overall network outputs, respectively, which is higher than the changes of others. This also proves that the signals in the turbine system have less noise and are more accurately measured than in the boiler system. However, the average individual change of most outputs is less than 1%, which indicates that any particular disturbed sensor signal does not affect the network output much. The network provides a robust signal estimate because of the application of the regularization technique and the robust training algorithm.

6.3 Sensor Fault Detection

This section presents the system detecting ability on sensor different types of faults. The

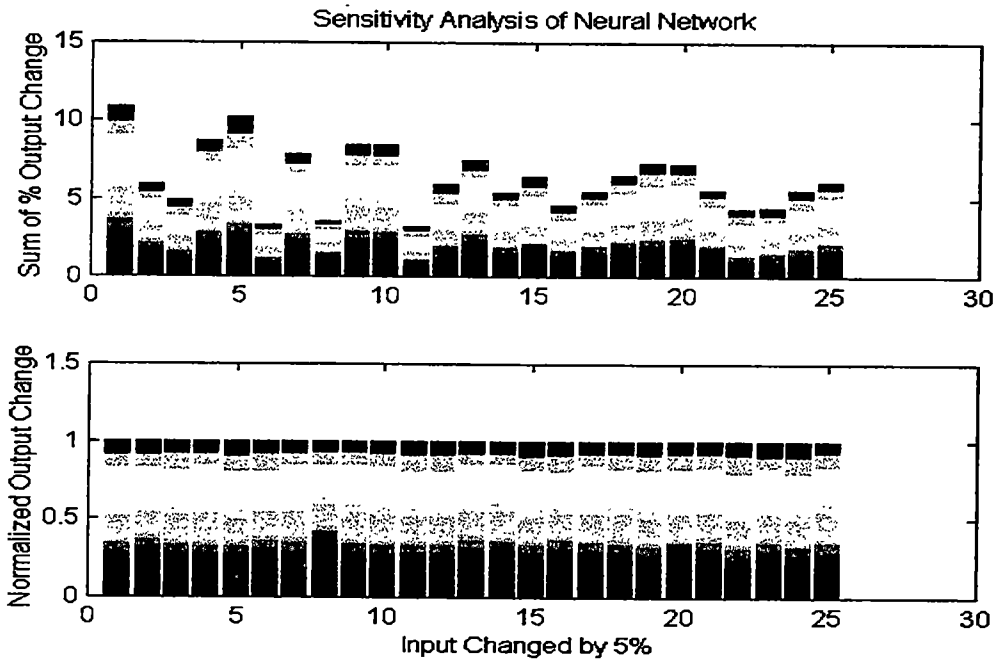


Figure 6.5 Sensitivity analysis with 5% perturbation in NN_1 input

common types of sensor faults include small drift, such as ramp drift and step drop; and gross failure. In this study, all sensors are assumed in calibration range, so no fault occurs in these signals. In order for the neural network-based sensor validation system to identify and detect the faults, artificial drifts were introduced in each of the inputs for system performance verification. All the types of possible faults were simulated on the designed sensor verification system. Also the data that were not used for training were applied to test the capability of the neural network models on generalization.

6.3.1 SPRT Parameter Setting

As discussed in Chapter 5, the MSPRT eliminates the possible intermittent alarms due to noise or prediction error. The gain factor in the MSPRT algorithm was statistically determined to filter out these intermittent alarms and produces false alarms when a true

drift is detected. It is also found that the determination of the faulted mean parameter m in SPRT depends on the training accuracy; that is, the more accurate the estimation, the smaller the m, and vice versa.

In this study, since no operating ranges of the sensors are provided, therefore, the determination of the detecting faulted mean m in SPRT algorithm is based on the mean operating level of the individual signal. This is determined by

$$m_i = 0.05\overline{X} \qquad i = 1,...,p$$

which is 5% of the mean operating level of the signal. Where $\overline{X}$ are the mean operating values of the data; while p is the number of signals.

6.3.2 Small Drift Detection

In this section, both the small ramp drift and the step drop were simulated for small drift detection. Results for a 1% per day ramp drift introduced in some of the signals operating during 01/11-01/22/1998 are presented in Figures C19 through C32.

Figure C19 presents the detection result for a 1%/day artificial drift on the Feedwater Flow, the MSPRT detects the fault at the 2693rd minute or 1.87% drift of the signal, and initiates a false alarm. While the SPRT initiates a false alarm at the 73rd minute or 0.05% drift, which is obviously triggered due to noise or prediction error but not actual drift. Figure C20 shows the detection result for the Measured Steam Flow. The MSPRT detects the real fault beginning at the 2719th minute or 1.89% drift, while the SPRT detects a fault occurring as early as in the 55th minute or 0.04% drift which is apparently

an intermittent alarm. The detection results on signals in the leak detection system are tabulated in Table 6.7. It is shown that the fault occurring in a signal with more accurate prediction can be detected much earlier than with less accurate prediction due to the noisy and inaccurate measurement.

Tests on small abrupt failures were also performed by artificially introducing a 5% step drop starting at the 10000^{th} minute to each of the signals. Some detection results are plotted in Figures C33 through C35. It has been shown that the MSPRT successfully detects the sensor abrupt failure immediately after it occurs for all the signals, and eliminates early intermittent alarms in most signal channels, which have relatively small prediction errors.

The results indicate that although the MSPRT delays the alarming time, it clears the intermittent alarms due to noise or spikes in most of the signal channels, and triggers alarms only when a true drift occurs. Therefore, the MSPRT improves the reliability of

Table 6.7 Detection Level on Interested Variables (1%/day ramp drift)

# Sigs	Detected drift (%)
1	4.77
2	3.53
24	2.20
25	1.87
34	1.20
35	1.20
37	2.74
46	4.00
47	5.65
48	3.37
124	0.93
125	0.68

the alarming system and provides more confidence on the detection results. These results also show that even though a drift is occurring in the signal, the network estimate does not change significantly. The sensor monitoring system successfully identifies and detects a sensor fault.

6.3.3 Gross Fault Detection

A gross fault is defined in this study as a drastic change in a signal value. Gross faults in this research were simulated in two cases:

- a large percentage drop in individual sensors
- a total failure in individual sensors to their maximum or minimum values, representing a gross fault "high" or gross fault "low", respectively. Depending on how "gross" the signal fails, the network may or may not remain stable.

A large drop in a signal's value may cause other outputs in the network to vary, since each input contributes to each output to some degree. A large fault can create false alarms in other channels. The residuals may change somewhat such that they are greater than the pre-set faulted mean values of the SPRT's. While the other variable residuals may vary, the amount of variation is only a small fraction of that of the faulty signal.

6.3.3.1 Large Drop Case

An artificially introduced 20% drop in individual sensors were simulated for sensor fault detection. It is found that with a large drop in one signal, the others closely related to the

faulted signal were also affected. In this part, a simulated result of signals affected by the faulted Feedwater Flow #1 signal is reported. Figure C36 presents the estimation on the faulted Feedwater Flow #1 and the detection result. It is shown that the validation system locates the fault and detects it immediately after the fault occurs. More importantly, this system almost fully recovers the signal from the faulty measurement. Furthermore, we can see that some other network outputs were also affected due to the correlated relationships. Figure C37 indicates that the network estimate of Unit Gross Generation signal was affected and classified faulty by the validation system. To avoid this mislocating problem, a sensor correction module is designed to replace the faulted measurement with its best estimate. This corrects the faulty input to the neural networks and the impact on other closely related signals is minimized. Section 6.7 presents the performance of the sensor replacement module.

6.3.3.2 Total Failure Case

Another type of gross fault is the total failure of the sensors. When it happens in one sensor, it is found that the other signals closely related to the faulted signal were degraded to a large degree. Figure C38 shows a total failure in Feedwater Flow #1 occurred at the 10000^{th} minute. The SPRT initiates a false alarm right after the fault occurs. Although the signal information is totally lost, the sensor validation system is still able to recover about 99.1%. After the replacement, the detection system successfully recovers the lost signal and the SPRT then finds no abnormality. The result is plotted in Figure C39.

Since the degradation in each affected channel and the true fault in the faulted channel were detected almost at the same time, so it is difficult to isolate the true fault by simply checking the false alarms. Other method is needed to locate the true faulty channel. Figure C40 shows that due to the total failure of Feedwater Flow signal, the Unit Gross Generation signal, which is estimated from the information provided by other signals, including Feedwater Flow, is degraded at a large degree. The SPRT also considers this as a fault and generates false alarms. We can see from the result that the SPRT initiates a false alarm for the affected Unit Gross Generation signal almost at the same time as it detects the faulted Feedwater Flow signal. Therefore, the true fault cannot be isolated by the alarms. However, the degraded magnitudes for the true fault and for the affected channels are different. Table 6.8 presents the average prediction error (APE) in percentage for each of the signals in NN_1 with the total failure of sensor measuring Feedwater Flow #1 (Signal #9). We can see from the table that although many signals are affected due to a sensor fault, except sensor #9, the others are within 3.5%. Therefore, based on the amount of changes in signals, the true fault can be distinguished.

6.4 Detectable Drift Level

The drift detectable level represents the sensitivity of the detection system. It is determined based on a statistical confidence interval. The smallest detectable drift for all monitorable signals are calculated as follows:

$$D = \frac{\sigma}{m} * 100$$

where σ - standard deviation of the prediction error ;

m - mean of the signal operating level.

Table 6.8 Network APE with A Total Sensor Failure

Sig #	APE (%)
1	-0.8897
2	2.1002
9	-100.0000
10	2.3624
13	2.5916
23	2.8252
24	2.8066
25	1.8254
34	3.3803
35	3.3748
36	2.6617
40	2.8640
41	2.6196
42	2.7358
43	2.7147
44	2.3723
45	2.6540
53	-0.2522
88	0.4436
91	0.2984
92	0.5492
103	0.5041
105	0.4049
106	0.2649
110	0.1816

Some results are presented in Table 6.9. This table shows that although the networks are successfully trained, the detection level for each signal varies significantly. The average detection level for signals in the boiler system in NN_1 is higher than in the turbine system. This is because that the signals recorded from the turbine system are much less

noisy and more accurately measured than the signals from the boiler system. The average smallest detectable level is ~2.5% for the signals in the boiler system and ~0.9% in the turbine system. However, due to the introduction of the filtering structure in the SPRT, the actual detection level of this system is higher. For example, the smallest detectable level for feedwater flow is 1.02%, while the actual detection level is 1.87%.

Table 6.9 Smallest Detectable Level of Signals in NN_1

Sig #	Smallest Detectable Level (%)
1	3.9162
2	3.6551
9	1.0230
10	0.9954
13	2.2192
23	0.9933
24	1.2494
25	0.8722
34	1.0836
35	1.0822
36	0.6875
40	0.8306
41	0.6944
42	0.6939
43	0.7511
44	0.7477
45	0.6899
53	2.9521
88	0.2252
91	0.2389
92	0.1645
103	0.3153
105	0.3602
106	0.5334
110	0.6080

6.5 Faulty Sensor Replacement

When a sensor is found faulty, it may not be corrected immediately due to operational constraints. Thus, it is desirable for the system to provide the correct value of the failed sensor so that the plant could continue operating without interruption. This is extremely important when the faulty signal is a control variable in a feedback control system. In addition, this would minimize degradation of the sensor monitoring system because the faulty input is replaced with a fault-free signal. The correction module designed in this study is capable of immediately replacing faulty sensor signals with their best estimates. Moreover, as described in previous section, the faulted sensor signal may affect the detecting outcomes for other good sensors. The faulty sensor replacement module replaces the faulty signal with its best estimate as virtual input to the neural networks. Since the network input becomes fault-free or less faulty, therefore, the influence on other signals is virtually eliminated.

In this research, it is assumed that only one sensor is faulty at a time, no multiple faults should occur at a same time. The robust training algorithm is designed under this assumption. The sensor replacement module continuously corrects the faulty signal and assures no multiple sensor fault happening at the same time. Two types of sensor faults, small drift and gross failure, were simulated.

6.5.1 Small Drift Case

It is found that when a small drift occurs in one sensor signal, others were not quite affected with the appreciation of the robustness of the networks. In this study, a small ramp of 1% per day was introduced to the raw measurement of Feedwater Flow #1 to

simulate the drift at the 5000th minute. The sensor validation system detects the fault and starts frequently alarming at the 3621st minute entering the drift, which is about 2.5% drift and the result is plotted in Figure C41. After the drifted signal was replaced by the neural network output, we can see from Figure C42 that the false alarms were eliminated. The system is back to normal. In this case, the other highly correlated signals were slightly affected, but the SPRT outcomes for these signal channels indicate no faults.

6.5.2 Gross Failure Case

As mentioned previously, the gross sensor failure will have a large impact on the detecting outcomes on other good sensors. This can be observed from the results in Figures C36 and C37. Therefore, repeated replacement procedures may be needed to maximally recover the lost signal.

Figure C43 demonstrates the corrected Feedwater Flow signal after the signal with 20% step drop is replaced. It is found that 99.5% of the lost signal is recovered. Figure C44 shows the detecting result for Unit Gross Generation signal after the faulted signal is replaced. It is proved that the replacement module is capable of correcting the mislocating problem caused by relatively large drifts.

Figure C45 presents the corrected Feedwater Flow signal with a total failure in the sensor after the replacement. Figure C46 shows the corresponding Unit Gross Generation after the faulted signal is replaced. We can see that when the sensor fails at the 10000th minute, the SPRT detects the fault immediately. The replacement module started replacing the faulted signal with its best estimate from the network. Although the sensor

is totally failed, the system is still able to recover 99.5% of the original signal. However, less detectable sensitivity should be set in avoiding mislocating problems on good sensors. The capability of signal recovery shows the robustness of the network models.

6.6 Model Change and Adaptation

Neural networks are not guaranteed to function as expected when operating outside the training region. Although they have good generalization abilities inside the training space, they must be retrained when expected to operate in new regions. There are several circumstances when the operating region may change:

- Component replacement;
- Component degradation or failure;
- Process parameters change, such as a change of flow rate;
- Cyclic or seasonal changes;

All of these conditions may have impact on the performance of the sensor monitoring system. When plant conditions change, the network must be able to adapt itself to the new operating conditions. With a slight change of the operating condition, the network may not need to be retrained, but only to be retuned using the linear TSVD regression technique detailed in Section 3.4.2, which is fast in implementation and can satisfy an on-line real-time adaptation requirement.

To demonstrate the neural network adaptation under new process condition, a network is trained with the data from one operating condition and tested with a set of data from a different operating condition. It is found that the operating condition changed from the operating period of winter 1998 to summer 1999. The measurement of Feedwater Flow during these two periods is plotted in Figure 6.8. It can be clearly seen the feedwater flow rate increased during summer 99 compared with that during winter 98. In this study, the data during winter 98 was used for network training and the data during summer 99 was used for network adaptation purpose.

Figure 6.6 shows that the winter 98 data used for network training does not cover the dynamic range of the summer 99 test data. The network trained with the winter 98 data is not expected to generalize well on the summer 99 operating condition that is beyond the network training region. The result can be seen later in this section.

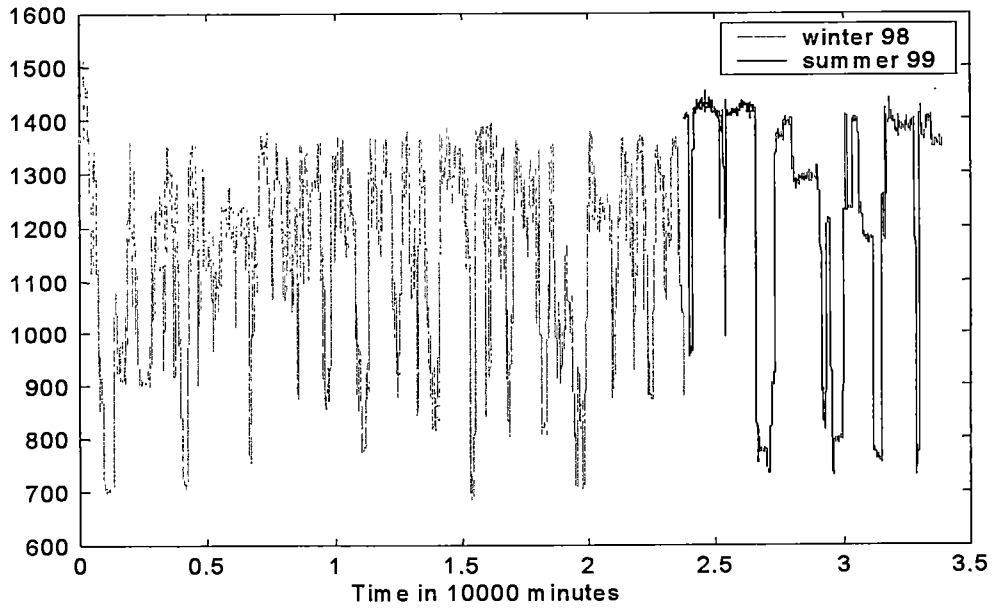


Figure 6.6 Operating condition change of Feedwater Flow (Klb/hr)

Figure C47 presents the network performance during these periods on Feedwater Flow. It indicates that the network model fits well on the winter 98 data which was used for training but performs badly on the summer 99 data which is beyond the model dynamic range. In this case, the model does not match the current plant operating status and needs to be modified. The SPRT initiates false alarms due to this mismatch.

When there is a slight change in operating conditions, complete network retraining is unnecessary. The TSVD technique linearly regresses the neural network model to accommodate the new operating condition by combining the current operating data and the previous operating data. By doing so, the network learns the inherent correlations for all dynamic operating regions. The result plotted in Figure C48 was obtained by using the TSVD algorithm to retune the network to include summer 99 operating period. The network's estimation performance in summer 99 operating period starting at the 23788^{th} minute, i.e., the beginning of the summer 99 operating period, has improved drastically. No false alarm occurs due to the mismatch. With the utilization of the TSVD technique, no network retraining is required and on-line model adaptation is possible.

The correlation inherent in the neural network inputs leads to an ill-conditioning problem (Hadamard 1923) that can cause increased noise level of model output (Qin, 1997) or instability of the model estimate (Hansen, 1989). Network regularization techniques such as the truncated singular value decomposition method can be utilized to correct these problems (Hines 1999). These techniques force the model to depend on all of the input information equally and reject perturbations due to signal noise. These techniques also result in sensor estimates that have less noise than their corresponding measured signals.

6.7 Summary

This study shows that an AANN-based sensor validation system is capable of monitoring plant-wide sensor performance because the neural networks can model relatively large number of variables. Another advantageous property of a neural network over other model-based techniques is that it can establish a process model with little prior knowledge on the process; and needs no extensive knowledge of interested process to develop a proper model.

The developed automation methodology requires little knowledge on neural network structure design, making the neural networks be more practical. A PCA-based technique was developed to estimate the number of nodes in the bottleneck layer of NLPCA architecture or in the hidden layer of single hidden layer AANN architecture. This technique extracts the useful information from the predictors and forms a few principal components (PCs) which attain sufficient information to capture the dynamics of the process. These PCs can be interpreted as a data compressor to greatly reduce the dimensionality of the data through input-output mapping. This simplifies network structure and reduces the possibility of overfitting. This also helps stabilizing the network performance by eliminating the collinearity problem. Furthermore, this technique reduces the noise level of the signals to be modeled.

The modified beta method estimates the number of nodes in mapping/demapping layers of an NLPCA architecture through a statistical method and removes the influence of data length as well as the random search steps which is difficult to determine. A properly selected coefficient C provides enough number of hidden nodes for easier network

training. However, experiment indicates that the determination of C is not crucial for networks to generalize. The application of a regularization method in this research ia similar to the Optimal Brain Damage (Cun, *et al.*, 1990) pruning scheme and disconnects the unnecessary weight connections between the hidden nodes.

The sensitivity analysis discovers that a robust training scheme provides the robustness of the network model. The scheme interrupts the data by adding noise into it one at a time. This assumes that no more than two signals degrade at the same time. Should multiple faults happen, the sensor correction/replacement module iteratively replaces the first detected faulty signal with its best estimate from the neural network model as the virtual sensor signal and ensures only one faulty signal exists at a time.

The TSVD mechanism is simple to manipulate. Since the output layers of neural network models are linear, the SVD technique is able to solve for the weight matrix by linear regression. Therefore, instead of retraining the whole network model, the SVD allows part of the model to be quickly retuned for on-line adaptation when a small change in plant operating conditions occurs.

The MSPRT uses a filter structure to eliminate the intermittent alarms caused by the spurious spikes as well as the model prediction errors. Therefore the MSPRT eliminates the requirement of residuals to be normally distributed. It is capable of dealing with the residuals containing not only the noise but also some true signal contents.

Also, the cross correlation analysis provides sufficient evidence on whether a TDNN can perform better than an AANN does. In order for TDNN to have a superior performance, delayed signals must contain helpful information for prediction.

The variables must have some degree of correlation for an AANN to be trained. The collinearity of the network inputs causes a network stability problem, and results in an increased variance of network outputs. Network regularization techniques combined with the singular value decomposition method was utilized to lessen the impact of these problems.

The result shows that this sensor validation system can detect the signal drift at an average level ~2.5% of the mean system normal operating level for signals in the boiler system and ~1% in the turbine system.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

The AANN-based sensor validation system with a robust training technique is capable of providing reliable estimates of faulty sensor measurements. Sensor faults are detectable by comparing the difference between the sensor measurements and the corresponding network estimates. This sensor validation system can detect sensor drifts at an average level of ~2.5% of the sensor's mean operating level in the boiler system and ~1% in the turbine system. This result is consistent with the fact that some signals in the boiler system are noisy and not accurately measured, such as the air combustion flow. This sensor validation system is capable of monitoring 74% of the instrument channels, the unmonitorable sensor channels have little correlation with others and therefore, cannot be inferred by others. Most of the unmonitorable channels are in the boiler system.

The correlated neural network inputs cause an ill-conditioned problem, which unstabilizes the network prediction. A bootstrap result demonstrates that the regularization technique reduces the severity of the ill-conditioned problem and tends to stabilize the network performance. Although the regularization introduces a bias to a solution, it greatly reduces the inconsistency of the model estimations, and therefore, greatly increase the reliably of the model.

The developed methodology of designing the optimal neural network architecture requires no prior knowledge to calculate the parameters in the algorithms, and greatly

simplifies the network structure design. This makes the neural networks more practical without requiring extensive neural network design experience.

Small faults, like slow drifts and small offsets, in one sensor only affect other sensor measurements to a minor degree due to the robustness of the networks. The faulty sensor can be easily detected and isolated by the MSPRT-based fault detection module. However, in gross fault situations, other closely related sensors may be affected to a large degree, and the true fault is difficult to locate and isolate using the MSPRT-based fault detection module. A method for identifying the faulty sensor is to measure the degree of change of each network output. The sensor channel with the largest variance, which is the faulty sensor, is getting replaced.

When operating conditions change, the neural network models must be modified to accommodate themselves to the new operating conditions. The use of the TSVD technique allows the network to adapt itself to the new operating conditions without retraining the entire network. This feature allows continuous on-line monitoring to be realized.

Although the MSPRT delays the detection time when compared with the traditional SPRT because of the introduced filtering structure, it effectively suppresses the early intermittent alarms due to noise or prediction errors and triggers alarms when a true fault is detected. The benefits of reducing false alarms outweigh the detriment of a slightly delayed detection time.

7.2 Recommendations for Future Work

This section analyzes the possible difficulties of the developed methodology. Some alternative solutions are recommended and the possible problems the alternatives may face are also discussed.

7.2.1 Network Regularization

Due to the collinearity of the network model inputs, the network performance may be unstable and regularization is needed. Over-regularized networks yield large prediction errors, while under-regularized ones generate predictions with relatively large uncertainties. Therefore, an optimal network model requires the regularization parameter be properly determined. However, properly selecting the regularization parameter, which controls the network performance, involves human interaction which is not desired in this automated process design. Unlike the regularization of linear models, regularization of nonlinear models, such as neural networks, is extremely difficult to control quantitatively. It should be pointed out that in the situation of sensor validation, the selection of the regularization parameter is not as crucial as some inferential sensing applications (Hines, *et al.*, 1999; Gribok, *et al.*, 1999). The most important issue in the sensor validation application is the robustness of the models, of which all the outputs, including the corresponding output to the faulty input, are not affected much by a faulty input.

7.2.2 Different Architecture - NLPLS

The sensor validation AANN architecture is relatively complicated, and leads to difficult training and computational limitation. Network architecture simplification is desired. Although the number of latent vectors must be determined, a NLPLS architecture, which

decorrelates the relationship between the network inputs, greatly reduces the complexity of the model and eliminates the collinearity problem. More importantly, regularization is inherently embedded in a NLPLS model. Because of the simple input/output mapping relationship, network variable grouping becomes unnecessary. Also the inferential type structure uses all variables but the predicted one for variable prediction, which increases the robustness of the sensor monitoring system.

However, the biggest concern for this architecture is the retuning procedure under the situation of operating condition changes. Unlike the retuning procedure of the AANNs studied in this dissertation, an NLPLS model is retuned sequentially through all ANN-structured inner models. This would require model retraining and it may become impracticable for on-line adjustment.

7.2.3 Residual Normalizing

In this research it is reported that the SPRT produces intermittent alarms due to non-normally distributed residuals. The residuals deviate from Gaussian due to spurious spikes which may be removed by filtering. Therefore, instead of modifying the SPRT algorithm, the residuals may be normalized through a spectrum transform filtering algorithm (Gross and Hoyer, 1995) so that they are normally distributed about the mean. Another approach may be to generate an autoregressive (AR) model from the residuals and to use its output as the SPRT input.

Bibliography

Bibliography

Aitchison, J. and Dunsmore, I. R. (1975), "Statistical prediction analysis," Cambridge University Press.

Akaike, H. (1974), "A new look at the statistical model identification," *IEEE Transactions on Automatic Control, AC_19*, pp. 716-723.

Akaho, S. and Amari, S. (1990), "O the capacity of three-layer networks," *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA Vol. III.

Aleksander, I. and Morton H. (1990), "An introduction to neural computing," London, Chapman & Hall.

Baum, E. B. (1988), "On the capacity of multilayer perceptrons," *Journal of Complexity*, No. 4, pp. 193-215.

Bishop, C. M. (1995), "Neural networks for pattern recognition," Oxford University Press Inc., New York, NY.

Brown, R. G. (1992), "Introduction to random signals and applied Kalman filtering," John Wiley & Sons, New York.

Cun, Y. L., *et al.* (1990), "Optimal brain damage", *Advances in Neural Information Processing System 2, Proceedings of the 1989 Conference*, San Mateo, CA, Morgan Kaufmann Publishers.

Cybenko, G. (1989), "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, 2, 303-314.

Dong, D. and McAvoy, T. (1994), "Sensor data analysis using autoassociative neural networks," *Proceedings of the World Congress on Neural Networks*, San Diego, CA, Vol. 1, pp. 161-166.

Dong, D. and McAvoy, T. (1996), "Nonlinear principal component analysis-based on principal curves and neural networks," *Computers in Chemical Engineering*, 20, pp. 65-78.

Dorr, R., *et al.* (1997), "Detection, isolation, and identification of sensor faults in nuclear power plants," *IEEE Transactions on Control Systems technology*, Vol. 5, No. 1.

Dunia, R., *et al.* (1996), "Identification of faulty sensors using principal component analysis," *AIChE Journal*, Vol. 42, No. 10.

Dunia, R. and Qin, S. (1998), "A unified geometric approach to process and sensor fault identification and reconstruction: the unidementional fault case," *Computers Chem Eng*, Vol. 22, No. 7.

Efron, B. (1982), "The jacknife, the bootstrap, and other resampling plans," *Society for Industrial and Applied Mathematics*, Philadelphia, PA.

Erbay, A. S. and Upadhyaya, B. R. (1997), "A personal computer-based on-line sensor validation system for nuclear power plants," *Nuclear Technology*, Vol. 119, pp. 63-75.

Eryurek, E. (1991), "Development and application of multi-layer neural networks for estimation of power plant variables," MS Thesis, The University of Tennessee.

Fahlman, S. E. and Lebiere, C. (1990), "The cascade-correlation learning architecture," *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann.

Fantoni, P. A. and Mazzola, A. (1996), "Multiple-failure signal validation in nuclear power plants using artificial neural networks," *Nuclear Technology*, Vol. 113, pp 368-374.

Fujita, O. (1998), "Statistical estimation of the number of hidden units for feedforward neural networks," *Neural Networks*, Vol. 11, pp. 852-859.

Geladi, P., and Kowalski, B. R. (1986), "Partial least squares regression: a tutorial," *Analytica Chimica Acta*, 185.

Gribok, A. V., Attieh, I., Hines, J. W., and Uhrig, R. E. (1999), "Regularization of Feedwater Flow Rate Evaluation for Venturi Meter Fouling Problems in Nuclear Power Plants", *Ninth International Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-9)*, San Francisco, CA.

Gross, K. and Hoyer, K. (1995), "Spectrum-transformed sequential testing method for signal validation applications, " *9th Power Plant Dynamics, Control & Testing Symposium Proceedings*, Knoxville, TN, Vol. 1.

Goutte, C., and Hansen, L. K. (1997), "Regularization with a pruning prior," *Neural Networks*, Vol. 10, No. 6.

Gross, K., *et al.* (1997), " Application of a model-based fault detection system to nuclear power signals," *9th International Conference on Intelligent Systems Applications to Power Systems*, pp. 66-70, Seoul, Korea.

Gross, K., *et al* (1998), "Industrial process surveillance system," Patent # 5764509, University of Chicago.

Hadamard, J. (1923), "Lectures on Cauchy's problem in linear partial differential equations", Yale University Press, New Haven.

Hagan, M. T. and Menhaj, M. (1994),"Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, 1994.

Hansen, L. K., *et al.* (1994), "Adaptive regularization," *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV*, Piscataway, New Jersey, pp. 78-87.

Hansen, L.K., *et al.* (1994), "Pruning from adaptive neural networks," Technical University of Denmark.

Hashem, S. (1997), "Optimal linear combinations of neural networks," *Neural Networks*, Vol. 10, No. 4.

Haykin, S. (1994), "Neural networks," MacMillan Publishing Company, Englewood Cliffs, NJ.

Hines, J. W., Gribok, Andrei V., Attieh, I., and Uhrig, R. E. (1999), "The Use of Regularization in Inferential Measurements", Presented at *the Enlarged Halden Programme Group (EHPG) Meeting*, Loen, Norway.

Hines, J. W., Uhrig, R. E., and Xu, X. (1998), "Sensor Validation and Instrument Calibration Monitoring," Midterm Report prepared by the University of Tennessee.

Hoskuldsson, A. (1988), "PLS regression methods," *J. Chemomet*, 2.

Jolliffe, I. T. (1986), "Principal component analysis," Springer-Verlag, New York.

Kalman, R. E. (1960), "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, 82.

Kramer, M. A. (1991), "Nonlinear principal component analysis using autoassociative neural networks," *AICHE Journal*, Vol. 37, No. 2, pp. 233-243.

Kramer, M. A. (1992), "Autoassociative neural networks," *Computers in Chemical Engineering*, 16:(4), pp. 313-328.

Kurita, T. (1990), "A method to determine the number of hidden units of three-layered neural networks by information criteria," *Transaction of the Institute of Electronics Information and Communication Engineers, J73-D-II*, pp. 1872-1878.

Larsen, J., *et al.* (1994), "Generalization performance of regularized neural network models," *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV*, Piscataway, New Jersey, pp. 42-51.

Larsen, J., *et al.* (1996), "Design and regularization of neural networks: the optimal use of a validation set," *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VI*, Piscataway, New Jersey, pp. 62-71.

Larsen, J., *et al.* (1996), "Regularization of neural networks," *Proceedings of the 4th Interdisciplinary Workshop*, Technical University of Denmark, pp. 59-66.

Levin, A., *et al.* (1994), "Fast pruning using principal components," *Advances in Neural Information Processing 6*, Morgan Kaufmann, San Diego, CA.

MacGregor, J.F., Marlin, T.E., Kresta, J.V., and Skagerberg B. (1991), "Some comments on neural networks and other empirical modeling methods," *Proceeding of the Chemical Process Control - IV Conference*, South Padre Island, TX, Feb. 18-22.

Masters, T. (1993), "Practical neural network recipes in C++", Academic Press, San Diego, CA.

MATLAB, High performance numeric computation and visualization software, (1995), The Mathworks Inc., Natick, MA.

Moody, J. E. (1992), "The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems," *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann.

Nabeshima, K., Susuki, K., and Turkan, T. (1995), "Real-time nuclear power plant monitoring with hybrid artificial intelligence systems," *9th Power Plant Dynamics, Control & Testing Symposium*, Vol. 2, pp. 55.01, University of Tennessee, May 24-26.

Navasimhan, S. Mah and R. (1988), "Generalized likelihood ratios for gross error identification in dynamic processes," *AIChE Journal*, Vol. 34.

Olvera, J. R. (1993), "Instrument calibration verification," Master Thesis, The University of Tennessee.

Petersen, M., *et al.* (1996), "Pruning with generalization based weight saliencies: γ OBD, γ OBS," *Advances in Neural Information Processing Systems 8, Proceedings of the 1995 Conference*, Cambridge, Massachusetts: MIT Press, pp. 521-528.

Press, W., *et al.* (1992), "Numerical recipes in C", Cambridge University Press.

Qin, S. (1997), " Neural networks for intelligent sensors and control --- Practical issues and some solutions," In *Neural Systems for Control*, Chapter 8, Edited by O. Omidvar and D. L. Elliott, Academic Press.

Qin, S. and McAvoy, T. (1992), "Nonlinear PLS modeling using neural networks," *Computers chem. Engng.*, Vol. 16, No.4, pp 379-391.

Qin, S. and Li, W. (1999), "Detection, identification, and reconstruction of faulty sensors with maximized sensitivity," *AIChE Journal*, Vol. 45, No . 9.

Scale, L.E. (1985), "Introduction to non-linear optimization," Spring-Verlag, New York, NY.

Sequin, C. H. and Clay, R.D. (1992), "Fault tolerance in artificial neural networks," *Proc. Int. Conf. on Neural Networks*, pp. I-703-708, San Diego, CA.

Singer, R., *et al.* (1995), "A pattern-recognition-based, fault-tolerant monitoring and diagnostic technique," *Seventh Symposium on Nuclear Reactor Surveillance and Diagnostics Proceedings*, Avignon, France.

Singer, R., *et al.* (1997), "Model-based nuclear power plant monitoring and fault detection: theoretical foundation," *9th International Conference on Intelligent Systems Applications to Power Systems*, pp. 60-65, Seoul, Korea.

Uhrig, R.E., Hines, J.W., Black, C., Wrest, D.J., Xu, X. (1996), ""Instrument surveillance and calibration verification system", Report Prepared by the University of Tennessee for Sandia National Laboratories, Contract No. AQ-6982.

Upadhyaya, B.R. and Eryurek, E. (1992), "Application of neural networks for sensor validation and plant monitoring," *NUCLEAR TECHNOLOGY*, Vol. 97, pp. 170-176.

Upadhyaya, B.R., Wolvaardt, F.P., and Glockler, O. (1987), "An integrated approach for sensor failure detection in dynamic systems," Research Report prepared for the Measurement & Control Engineering Center, Report No. NE-MCEC-BRU-87-01.

Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T., Alkon, D. L. (1988), "Accelerating the convergence of the backpropagation method," *Biological Cybernetics*, Vol. 59, pp. 257-263.

Wald, A. (1945), "Sequential tests of statistical hypothesis," *Ann. Math. Statist.*, Vol. 16, pp.117-186.

Williams, P. (1995), "Bayesian regularization and pruning using a Laplace prior," *Neural Computation*, 7(1) , 117-143.

Wrest, D. J. (1996), "Instrument surveillance and calibration verification through plant wide monitoring using autoassociative neural networks," MS Thesis, The University of Tennessee.

Appendices

Appendix A. Tables of Variable Grouping

Table A1 Variables for Monitoring

# Sigs	Signal Name	Unit
1	Combustion Air Flow A1	Pct
2	Combustion Air Flow B1	Pct
3	BFP A Discharge Flow - Raw	Klb/hr
4	BFP B Discharge Flow - Raw	Klb/hr
5	BFP C Discharge Flow - Raw	Klb/hr
6	BCW Pump Leakoff Flow - Raw	Klb/hr
7	Condensate Flow - Raw	Klb/hr
8	Coldwell Tank Makeup Flow	Klb/hr
9	Feedwater Flow #1 - Raw	Klb/hr
10	Feedwater Flow #2 - Raw	Klb/hr
11	FWH 3 Drain Flow - Raw	Klb/hr
12	Building Heating Steam Flow	Klb/hr
13	Hotwell Pumps Discharge Flow	Klb/hr
14	Pulv A PA Flow Rate	Klb/hr
15	Pulv B PA Flow Rate	Klb/hr
16	Pulv C PA Flow Rate	Klb/hr
17	Pulv D PA Flow Rate	Klb/hr
18	Pulv E PA Flow Rate	Klb/hr
19	Pulv F PA Flow Rate	Klb/hr
20	Reheat Spray A Flow	Klb/hr
21	Superheat Spray A Flow	Klb/hr
22	Superheat Spray B Flow	Klb/hr
23	Measured Steam Flow - Raw	Klb/hr
24	Unit Gross Generation	MW
25	Station Service Load	MW
26	Coldwell Tank Level	Inches
27	Selected Deaerator Level	Inches
28	FW Htr 1 Level	Inches
29	FW Htr 3 Level	Inches
30	Excess O2 A in Reheat Furnace	%Vol
31	Excess O2 B in Reheat Furnace	%Vol
32	Excess O2 A in Superheat Furnace	%Vol
33	Excess O2 B in Superheat Furnace	%Vol
34	First Stage Pressure A	psig
35	First Stage Pressure B	psig
36	Cold Reheat Pressure at Turbine	psig
37	Deaerator Pressure	psig
38	Selected Drum Pressure	psig
39	FW Entering Economizer Pressure	psig

40	FW Heater 1 Extraction Pressure	psig
41	FW Heater 2 Extraction Pressure	psig
42	FW Heater 3 Extraction Pressure	psig
43	FW Heater 4 Extraction Pressure	psig
44	FW Heater 5 Ext Press - Abs	psia
45	Hot Reheat Pressure at Turbine	psig
46	ID Fan A Suction Pressure	InH2O
47	ID Fan B Suction Pressure	InH2O
48	RH Furnace Press After Econ	InH2O
49	RH Furnace Pressure A	InH2O
50	SH Furnace Press After HT SH	InH2O
51	SH Furnace Press After Econ	InH2O
52	SH Furnace Pressure A	InH2O
53	SH Windbox Pressure	InH2O
54	Throttle Pressure at Stop Vlv A	psig
55	Throttle Pressure at Stop Vlv B	psig
56	Turbine Exhaust Pressure	InHgAbs
57	Unit CEMS RH CO2	Pct
58	Unit CEMS RH CO	PPM
59	BAILEY CO MONITOR - REHEAT FURN	PPM
60	Unit CEMS RH NOx LB/MMBTU	LB/MMBTU
61	Unit CEMS RH NOx PPM	PPM
62	Unit CEMS SH CO2	Pct
63	Unit CEMS SH CO	PPM
64	BAILEY CO MONITOR - SUPERHEAT	PPM
65	Unit CEMS SH NOx LB/MMBTU	LB/MMBTU
66	Unit CEMS SH NOx PPM	PPM
67	Pulv A Flow	Klb/hr
68	Pulv B Flow	Klb/hr
69	Pulv C Flow	Klb/hr
70	Pulv D Flow	Klb/hr
71	Pulv E Flow	Klb/hr
72	Pulv F Flow	Klb/hr
73	Ambient Air Temperature	DegF
74	Air Preheater A Air Inlet Temp	DegF
75	Air Preheater A Gas Inlet Temp	DegF
76	Air Preheater A Gas Outlet Temp	DegF
77	Air Preheater B Air Inlet Temp	DegF
78	Air Preheater B Gas Inlet Temp	DegF
79	Air Preheater B Gas Outlet Temp	DegF
80	BFP Suction Header Temperature	DegF
81	Circ Water Inlet Temp, East	DegF

82	Circ Water Inlet Temp, West	DegF
83	Circ Water Outlet Temp, East	DegF
84	Circ Water Outlet Temp, West	DegF
85	Cold Reheat Temp at Turbine	DegF
86	FW Heater 1 Drain Temperature	DegF
87	FW Heater 1 Extraction Temp	DegF
88	FW Heater 1 Water In Temp	DegF
89	FW Heater 1 Water Out Temp	DegF
90	FW Heater 2 Drain Temperature	DegF
91	FW Heater 2 Water In Temp	DegF
92	FW Heater 3 Drain Temperature	DegF
93	FW Heater 3 Extraction Temp	DegF
94	FW Heater 3 Water In Temp	DegF
95	FW Heater 4 Extraction Temp	DegF
96	FW Heater 5 Drain Temperature	DegF
97	FW Heater 5 Extraction Temp	DegF
98	FW Heater 5 Water Out Temp	DegF
99	FW Heater 6 Drain Temperature	DegF
100	FW Heater 6 Extraction Temp A	DegF
101	FW Heater 6 Extraction Temp B	DegF
102	FW Heater 6 Water Out Temp	DegF
103	FW Heater 7 Drain Temperature	DegF
104	FW Heater 7 Extraction Temp A	DegF
105	FW Heater 7 Water Out Temp	DegF
106	FW Heater 8 Drain Temperature	DegF
107	FW Heater 8 Extraction Temp A	DegF
108	FW Heater 8 Extraction Temp B	DegF
109	FW Heater 8 Water In Temp	DegF
110	FW Heater 8 Water Out Temp	DegF
111	Hot Reheat Temp at Int Vlv A	DegF
112	Hot Reheat Temp at Int Vlv B	DegF
113	Pulverizer A Outlet Temperature	DegF
114	Pulverizer B Outlet Temperature	DegF
115	Pulverizer C Outlet Temperature	DegF
116	Pulverizer D Outlet Temperature	DegF
117	Pulverizer E Outlet Temperature	DegF
118	Pulverizer F Outlet Temperature	DegF
119	RH Attemp A Before Spray Temp	DegF
120	RH Outlet Header Temperature A	DegF
121	Reheat Outlet Temperature #1	DegF
122	Reheat Outlet Temperature #2	DegF
123	SH Outlet Header Temperature A	DegF

124	Superheat Outlet Temperature #1	DegF
125	Superheat Outlet Temperature #2	DegF
126	Throttle Temperature at Turbine	DegF
127	Auxiliary Air Damper AA Position	Pct
128	Auxiliary Air Damper AB Position	Pct
129	Auxiliary Air Damper BC Position	Pct
130	Auxiliary Air Damper CC Position	Pct
131	Auxiliary Air Damper DD Position	Pct
132	Auxiliary Air Damper DE Position	Pct
133	Auxiliary Air Damper EF Position	Pct
134	Auxiliary Air Damper FF Position	Pct
135	RH Furnace Tilt Position	Deg
136	SH Furnace Tilt Position	Deg

Table A2 Variable Selection via NLPLS

No Sig	Signal Name
1	Combustion Air Flow A1
2	Combustion Air Flow B1
3	BFP A Discharge Flow - Raw
4	BFP B Discharge Flow - Raw
6	BCW Pump Leakoff Flow - Raw
7	Condensate Flow - Raw
9	Feedwater Flow #1 - Raw
10	Feedwater Flow #2 - Raw
13	Hot Reheat Steam Flow - Raw
14	Hotwell Pumps Discharge Flow
18	Pulv D PA Flow Rate
25	Measured Steam Flow - Raw
26	Unit Gross Generation
27	Station Service Load
30	FW Htr 1 Level
32	FW Htr 3 Level
34	Excess O2 B in Reheat Furnace
37	First Stage Pressure A
38	First Stage Pressure B
39	Cold Reheat Pressure at Turbine
40	Deaerator Pressure
41	Selected Drum Pressure
42	FW Entering Economizer Pressure
43	FW Heater 1 Extraction Pressure
44	FW Heater 2 Extraction Pressure
45	FW Heater 3 Extraction Pressure
46	FW Heater 4 Extraction Pressure
47	FW Heater 5 Ext Press - Abs
48	Hot Reheat Pressure at Turbine
49	ID Fan A Suction Pressure
50	ID Fan B Suction Pressure
51	RH Furnace Press After Econ
53	SH Furnace Press After HT SH
54	SH Furnace Press After Econ
56	SH Windbox Pressure
57	Throttle Pressure at Stop Vlv A
58	Throttle Pressure at Stop Vlv B
59	Turbine Exhaust Pressure
60	Unit CEMS RH CO2

63	Unit CEMS RH NOx LB/MMBTU
64	Unit CEMS RH NOx PPM
65	Unit CEMS SH CO2
68	Unit CEMS SH NOx LB/MMBTU
69	Unit CEMS SH NOx PPM
73	Pulv D Flow
76	Ambient Air Temperature
77	Air Preheater A Air Inlet Temp
78	Air Preheater A Gas Inlet Temp
79	Air Preheater A Gas Outlet Temp
80	Air Preheater B Air Inlet Temp
81	Air Preheater B Gas Inlet Temp
82	Air Preheater B Gas Outlet Temp
83	BFP Suction Header Temperature
84	Circ Water Inlet Temp, East
85	Circ Water Inlet Temp, West
86	Circ Water Outlet Temp, East
87	Circ Water Outlet Temp, West
88	Cold Reheat Temp at Turbine
89	FW Heater 1 Drain Temperature
90	FW Heater 1 Extraction Temp
91	FW Heater 1 Water In Temp
92	FW Heater 1 Water Out Temp
93	FW Heater 2 Drain Temperature
94	FW Heater 2 Water In Temp
95	FW Heater 3 Drain Temperature
96	FW Heater 3 Extraction Temp
97	FW Heater 3 Water In Temp
98	FW Heater 4 Extraction Temp
99	FW Heater 5 Drain Temperature
100	FW Heater 5 Extraction Temp
101	FW Heater 5 Water Out Temp
102	FW Heater 6 Drain Temperature
103	FW Heater 6 Extraction Temp A
104	FW Heater 6 Extraction Temp B
105	FW Heater 6 Water Out Temp
106	FW Heater 7 Drain Temperature
107	FW Heater 7 Extraction Temp A
109	FW Heater 7 Water Out Temp
110	FW Heater 8 Drain Temperature
111	FW Heater 8 Extraction Temp A
112	FW Heater 8 Extraction Temp B

113	FW Heater 8 Water In Temp
114	FW Heater 8 Water Out Temp
115	Hot Reheat Temp at Int Vlv A
116	Hot Reheat Temp at Int Vlv B
117	Pulverizer A Outlet Temperature
118	Pulverizer B Outlet Temperature
119	Pulverizer C Outlet Temperature
120	Pulverizer D Outlet Temperature
121	Pulverizer E Outlet Temperature
122	Pulverizer F Outlet Temperature
123	RH Attemp A Before Spray Temp
124	RH Outlet Header Temperature A
125	Reheat Outlet Temperature #1
126	Reheat Outlet Temperature #2
127	SH Outlet Header Temperature A
128	Superheat Outlet Temperature #1
129	Superheat Outlet Temperature #2
130	Throttle Temperature at Turbine
131	Auxiliary Air Damper AA Position
132	Auxiliary Air Damper AB Position
133	Auxiliary Air Damper BC Position
134	Auxiliary Air Damper CC Position
135	Auxiliary Air Damper DD Position
136	Auxiliary Air Damper DE Position

Table A3 Grouping with Exclusion Method

G #	Variables Grouped																									
1	1	2	25	45	41	36	42	43	23	44	40	35	34	10	9	92	24	88	53	103	105	106	110	91	13
2	37	80	94	90	89	7	3	98	107	47	46														
3	48	4	78																						
4	124	125	123	119	87	85	75	136																	
5	5	54	55																						
6	6																								
7	8																								
8	11	73																							
9	12	22	77	74																					
10	14	113	67																						
11	15	114	68																						
12	16	115	118																						
13	17	70	51	39	38	50	96	72	29	84															
14	18	117	71	104	120	121	122	112	111	95	93														
15	19	79																							
16	20																								
17	21	102	99																						
18	26	126																							
19	27																								
20	28																								
21	30	33	32	31	60	56	109	86	69	62	57														
22	49	133	132	134	131																				
23	52																								
24	58																								
25	59																								
26	61	65	66																						
27	63																								
28	64																								
29	76																								

30	81 82 83
31	97 108
32	100
33	101
34	116
35	127 130 128 129
36	135

Table A4 Grouping with Non-Exclusion Method

G #	Variables Grouped																									
1	1	2	25	45	41	36	42	43	23	44	40	35	34	10	9	92	24	88	53	103	105	106	110	91	13	
2	37	80	94	92	42	44	43	45	41	36	9	40	24	88	34	35	91	10	103	105	23	90	89	106	25	
3	46	47	48	70	38	119	29	39	87	85	3	98	4	107	78	94	80	53	37	89	110	7	106	13	91	
4	124	125	123	119	87	85	75	136																		
5	5	54	55																							
6	6																									
7	8																									
8	11	73																								
9	12	73	22	77	74																					
10	14	113	67																							
11	15	114	68																							
12	16	115	118																							
13	17	70	51	13	35	34	53	1	9	4	7	40	23	10	107	37	42	41	36	45	44	43	39	25	24	
14	18	117	71	104	120	121	122	112	111	95	93															
15	19	118	16	79	115																					
16	20																									
17	21	47	32	46	30	31	70	50	1	69	78	38	57	39	2	107	102	13	99	51	4	110	62	53	3	
18	26	22																								
19	27																									
20	28																									
21	33	30	32	31	46	47	60	48	84	56	109	70	102	99	78	29	50	96	72	2	1	69	51	86	119	
22	49	133	132	134	131																					
23	52																									
24	58																									
25	59																									
26	61	60	65	48	46	47	98	38	107	39	110	3	80	4	94	106	89	103	90	105	25	91	88	34	35	
27	63																									
28	64																									
29	66	65	61	60																						

30	76	79																							
31	81	82	83	84	109	56	22																		
32	97	56																							
33	100	93																							
34	101	12																							
35	108	97	109	56	83	84																			
36	116																								
37	126	22	26																						
38	127	130	128	129	51	50	75	87	85	119	70	2	1	13	23	25	34	35	40	9	36	42	43	45	41
39	135	132																							

Table A5 Unmonitorable Variables Through Grouping

# Sigs	Signal Name	Unit
5	BFP C Discharge Flow - Raw	Klb/hr
6	BCW Pump Leakoff Flow - Raw	Klb/hr
8	Coldwell Tank Makeup Flow	Klb/hr
11	FWH 3 Drain Flow - Raw	Klb/hr
12	Building Heating Steam Flow	Klb/hr
14	Pulv A PA Flow Rate	Klb/hr
15	Pulv B PA Flow Rate	Klb/hr
16	Pulv C PA Flow Rate	Klb/hr
19	Pulv F PA Flow Rate	Klb/hr
20	Reheat Spray A Flow	Klb/hr
22	Superheat Spray B Flow	Klb/hr
26	Coldwell Tank Level	Inches
27	Selected Deaerator Level	Inches
28	FW Htr 1 Level	Inches
49	RH Furnace Pressure A	InH2O
52	SH Furnace Pressure A	InH2O
54	Throttle Pressure at Stop Vlv A	psig
55	Throttle Pressure at Stop Vlv B	psig
58	Unit CEMS RH CO	PPM
59	BAILEY CO MONITOR - REHEAT FURN	PPM
63	Unit CEMS SH CO	PPM
64	BAILEY CO MONITOR - SUPERHEAT	PPM
66	Unit CEMS SH NOx PPM	PPM
67	Pulv A Flow	Klb/hr
76	Air Preheater A Gas Outlet Temp	DegF
79	Air Preheater B Gas Outlet Temp	DegF
97	FW Heater 5 Extraction Temp	DegF
100	FW Heater 6 Extraction Temp A	DegF
101	FW Heater 6 Extraction Temp B	DegF
108	FW Heater 8 Extraction Temp B	DegF
113	Pulverizer A Outlet Temperature	DegF
114	Pulverizer B Outlet Temperature	DegF
116	Pulverizer D Outlet Temperature	DegF
118	Pulverizer F Outlet Temperature	DegF
126	Throttle Temperature at Turbine	DegF
135	RH Furnace Tilt Position	Deg

Table A6 Variables in NN_1

# Sigs	Signal Name	Unit
1	Combustion Air Flow A1	Pct
2	Combustion Air Flow B1	Pct
9	Feedwater Flow #1 - Raw	Klb/hr
10	Feedwater Flow #2 - Raw	Klb/hr
13	Hotwell Pumps Discharge Flow	Klb/hr
23	Measured Steam Flow - Raw	Klb/hr
24	Unit Gross Generation	MW
25	Station Service Load	MW
34	First Stage Pressure A	psig
35	First Stage Pressure B	psig
36	Cold Reheat Pressure at Turbine	psig
40	FW Heater 1 Extraction Pressure	psig
41	FW Heater 2 Extraction Pressure	psig
42	FW Heater 3 Extraction Pressure	psig
43	FW Heater 4 Extraction Pressure	psig
44	FW Heater 5 Ext Press - Abs	psia
45	Hot Reheat Pressure at Turbine	psig
53	SH Windbox Pressure	InH2O
88	FW Heater 1 Water In Temp	DegF
91	FW Heater 2 Water In Temp	DegF
92	FW Heater 3 Drain Temperature	DegF
103	FW Heater 7 Drain Temperature	DegF
105	FW Heater 7 Water Out Temp	DegF
106	FW Heater 8 Drain Temperature	DegF
110	FW Heater 8 Water Out Temp	DegF

Table A7 Variables in NN_2

# Sigs	Signal Name	Unit
9	Feedwater Flow #1 - Raw	Klb/hr
10	Feedwater Flow #2 - Raw	Klb/hr
23	Measured Steam Flow - Raw	Klb/hr
24	Unit Gross Generation	MW
25	Station Service Load	MW
34	First Stage Pressure A	psig
35	First Stage Pressure B	psig
36	Cold Reheat Pressure at Turbine	psig
37	Deaerator Pressure	psig
40	FW Heater 1 Extraction Pressure	psig
41	FW Heater 2 Extraction Pressure	psig
42	FW Heater 3 Extraction Pressure	psig
43	FW Heater 4 Extraction Pressure	psig
44	FW Heater 5 Ext Press - Abs	psia
45	Hot Reheat Pressure at Turbine	psig
80	BFP Suction Header Temperature	DegF
88	FW Heater 1 Water In Temp	DegF
89	FW Heater 1 Water Out Temp	DegF
90	FW Heater 2 Drain Temperature	DegF
91	FW Heater 2 Water In Temp	DegF
92	FW Heater 3 Drain Temperature	DegF
94	FW Heater 3 Water In Temp	DegF
103	FW Heater 7 Drain Temperature	DegF
105	FW Heater 7 Water Out Temp	DegF
106	FW Heater 8 Drain Temperature	DegF

Table A8 Variables in NN_3

# Sigs	Signal Name	Unit
3	BFP A Discharge Flow - Raw	Klb/hr
4	BFP B Discharge Flow - Raw	Klb/hr
7	Condensate Flow - Raw	Klb/hr
13	Hotwell Pumps Discharge Flow	Klb/hr
29	FW Htr 3 Level	Inches
37	Deaerator Pressure	psig
38	Selected Drum Pressure	psig
39	FW Entering Economizer Pressure	psig
46	ID Fan A Suction Pressure	InH2O
47	ID Fan B Suction Pressure	InH2O
48	RH Furnace Press After Econ	InH2O
53	SH Windbox Pressure	InH2O
70	Pulv D Flow	Klb/hr
78	Air Preheater B Gas Inlet Temp	DegF
80	BFP Suction Header Temperature	DegF
85	Cold Reheat Temp at Turbine	DegF
87	FW Heater 1 Extraction Temp	DegF
89	FW Heater 1 Water Out Temp	DegF
91	FW Heater 2 Water In Temp	DegF
94	FW Heater 3 Water In Temp	DegF
98	FW Heater 5 Water Out Temp	DegF
106	FW Heater 8 Drain Temperature	DegF
107	FW Heater 8 Extraction Temp A	DegF
110	FW Heater 8 Water Out Temp	DegF
119	RH Attemp A Before Spray Temp	DegF

Table A9 Variables in NN_4

# Sigs	Signal Name	Unit
75	Air Preheater A Gas Inlet Temp	DegF
85	Cold Reheat Temp at Turbine	DegF
87	FW Heater 1 Extraction Temp	DegF
119	RH Attemp A Before Spray Temp	DegF
123	SH Outlet Header Temperature A	DegF
124	Superheat Outlet Temperature #1	DegF
125	Superheat Outlet Temperature #2	DegF
136	SH Furnace Tilt Position	Deg

Table A10 Variables in NN_5

# Sigs	Signal Name	Unit
1	Combustion Air Flow A1	Pct
4	BFP B Discharge Flow - Raw	Klb/hr
7	Condensate Flow - Raw	Klb/hr
9	Feedwater Flow #1 - Raw	Klb/hr
10	Feedwater Flow #2 - Raw	Klb/hr
13	Hotwell Pumps Discharge Flow	Klb/hr
17	Pulv D PA Flow Rate	Klb/hr
23	Measured Steam Flow - Raw	Klb/hr
24	Unit Gross Generation	MW
25	Station Service Load	MW
34	First Stage Pressure A	psig
35	First Stage Pressure B	psig
36	Cold Reheat Pressure at Turbine	psig
37	Deaerator Pressure	psig
39	FW Entering Economizer Pressure	psig
40	FW Heater 1 Extraction Pressure	psig
41	FW Heater 2 Extraction Pressure	psig
42	FW Heater 3 Extraction Pressure	psig
43	FW Heater 4 Extraction Pressure	psig
44	FW Heater 5 Ext Press - Abs	psia
45	Hot Reheat Pressure at Turbine	psig
51	SH Furnace Press After Econ	InH2O
53	SH Windbox Pressure	InH2O
70	Pulv D Flow	Klb/hr
107	FW Heater 8 Extraction Temp A	DegF

Table A11 Variables in NN_6

# Sigs	Signal Name	Unit
18	Pulv E PA Flow Rate	Klb/hr
71	Pulv E Flow	Klb/hr
93	FW Heater 3 Extraction Temp	DegF
95	FW Heater 4 Extraction Temp	DegF
104	FW Heater 7 Extraction Temp A	DegF
111	Hot Reheat Temp at Int Vlv A	DegF
112	Hot Reheat Temp at Int Vlv B	DegF
117	Pulverizer E Outlet Temperature	DegF
120	RH Outlet Header Temperature A	DegF
121	Reheat Outlet Temperature #1	DegF
122	Reheat Outlet Temperature #2	DegF

Table A12 Variables in NN_7

# Sigs	Signal Name	Unit
1	Combustion Air Flow A1	Pct
2	Combustion Air Flow B1	Pct
3	BFP A Discharge Flow - Raw	Klb/hr
4	BFP B Discharge Flow - Raw	Klb/hr
13	Hotwell Pumps Discharge Flow	Klb/hr
21	Superheat Spray A Flow	Klb/hr
30	Excess O2 A in Reheat Furnace	%Vol
31	Excess O2 B in Reheat Furnace	%Vol
32	Excess O2 A in Superheat Furnace	%Vol
38	Selected Drum Pressure	psig
39	FW Entering Economizer Pressure	psig
46	ID Fan A Suction Pressure	InH2O
47	ID Fan B Suction Pressure	InH2O
50	SH Furnace Press After HT SH	InH2O
51	SH Furnace Press After Econ	InH2O
53	SH Windbox Pressure	InH2O
57	Unit CEMS RH CO2	Pct
62	Unit CEMS SH CO2	Pct
69	Pulv C Flow	Klb/hr
70	Pulv D Flow	Klb/hr
78	Air Preheater B Gas Inlet Temp	DegF
99	FW Heater 6 Drain Temperature	DegF
102	FW Heater 6 Water Out Temp	DegF
107	FW Heater 8 Extraction Temp A	DegF
110	FW Heater 8 Water Out Temp	DegF

Table A13 Variables in NN_8

# Sigs	Signal Name	Unit
1	Combustion Air Flow A1	Pct
2	Combustion Air Flow B1	Pct
29	FW Htr 3 Level	Inches
30	Excess O2 A in Reheat Furnace	%Vol
31	Excess O2 B in Reheat Furnace	%Vol
32	Excess O2 A in Superheat Furnace	%Vol
33	Excess O2 B in Superheat Furnace	%Vol
46	ID Fan A Suction Pressure	InH2O
47	ID Fan B Suction Pressure	InH2O
48	RH Furnace Press After Econ	InH2O
50	SH Furnace Press After HT SH	InH2O
51	SH Furnace Press After Econ	InH2O
56	Turbine Exhaust Pressure	InHgAbs
60	Unit CEMS RH NOx LB/MMBTU	LB/MMBTU
69	Pulv C Flow	Klb/hr
70	Pulv D Flow	Klb/hr
72	Pulv F Flow	Klb/hr
78	Air Preheater B Gas Inlet Temp	DegF
84	Circ Water Outlet Temp, West	DegF
86	FW Heater 1 Drain Temperature	DegF
96	FW Heater 5 Drain Temperature	DegF
99	FW Heater 6 Drain Temperature	DegF
102	FW Heater 6 Water Out Temp	DegF
109	FW Heater 8 Water In Temp	DegF
119	RH Attemp A Before Spray Temp	DegF

Table A14 Variables in NN_9

# Sigs	Signal Name	Unit
3	BFP A Discharge Flow - Raw	Klb/hr
4	BFP B Discharge Flow - Raw	Klb/hr
25	Station Service Load	MW
34	First Stage Pressure A	psig
35	First Stage Pressure B	psig
38	Selected Drum Pressure	psig
39	FW Entering Economizer Pressure	psig
46	ID Fan A Suction Pressure	InH2O
47	ID Fan B Suction Pressure	InH2O
48	RH Furnace Press After Econ	InH2O
60	Unit CEMS RH NOx LB/MMBTU	LB/MMBTU
61	Unit CEMS RH NOx PPM	PPM
65	Unit CEMS SH NOx LB/MMBTU	LB/MMBTU
80	BFP Suction Header Temperature	DegF
88	FW Heater 1 Water In Temp	DegF
89	FW Heater 1 Water Out Temp	DegF
90	FW Heater 2 Drain Temperature	DegF
91	FW Heater 2 Water In Temp	DegF
94	FW Heater 3 Water In Temp	DegF
98	FW Heater 5 Water Out Temp	DegF
103	FW Heater 7 Drain Temperature	DegF
105	FW Heater 7 Water Out Temp	DegF
106	FW Heater 8 Drain Temperature	DegF
107	FW Heater 8 Extraction Temp A	DegF
110	FW Heater 8 Water Out Temp	DegF

Table A15 Variables in NN_10

# Sigs	Signal Name	Unit
50	SH Furnace Press After HT SH	InH2O
51	SH Furnace Press After Econ	InH2O
68	Pulv B Flow	Klb/hr
75	Air Preheater A Gas Inlet Temp	DegF
85	Cold Reheat Temp at Turbine	DegF
87	FW Heater 1 Extraction Temp	DegF
119	RH Attemp A Before Spray Temp	DegF

Table A16 Variables in NN_11

# Sigs	Signal Name	Unit
21	Superheat Spray A Flow	Klb/hr
73	Ambient Air Temperature	DegF
74	Air Preheater A Air Inlet Temp	DegF
77	Air Preheater B Air Inlet Temp	DegF
96	FW Heater 5 Drain Temperature	DegF
99	FW Heater 6 Drain Temperature	DegF
102	FW Heater 6 Water Out Temp	DegF

Table A17 Variables in NN_12

# Sigs	Signal Name	Unit
56	Turbine Exhaust Pressure	InHgAbs
81	Circ Water Inlet Temp, East	DegF
82	Circ Water Inlet Temp, West	DegF
83	Circ Water Outlet Temp, East	DegF
84	Circ Water Outlet Temp, West	DegF
109	FW Heater 8 Water In Temp	DegF

Table A18 Variables in NN_13

# Sigs	Signal Name	Unit
3	BFP A Discharge Flow - Raw	Klb/hr
10	Feedwater Flow #2 - Raw	Klb/hr
30	Excess O2 A in Reheat Furnace	%Vol
31	Excess O2 B in Reheat Furnace	%Vol
32	Excess O2 A in Superheat Furnace	%Vol
33	Excess O2 B in Superheat Furnace	%Vol
37	Deaerator Pressure	psig
57	Unit CEMS RH CO2	Pct
62	Unit CEMS SH CO2	Pct
69	Pulv C Flow	Klb/hr
72	Pulv F Flow	Klb/hr
80	BFP Suction Header Temperature	DegF
86	FW Heater 1 Drain Temperature	DegF
88	FW Heater 1 Water In Temp	DegF
89	FW Heater 1 Water Out Temp	DegF
90	FW Heater 2 Drain Temperature	DegF
91	FW Heater 2 Water In Temp	DegF
92	FW Heater 3 Drain Temperature	DegF
94	FW Heater 3 Water In Temp	DegF
98	FW Heater 5 Water Out Temp	DegF
103	FW Heater 7 Drain Temperature	DegF
105	FW Heater 7 Water Out Temp	DegF
106	FW Heater 8 Drain Temperature	DegF
110	FW Heater 8 Water Out Temp	DegF
115	Pulverizer C Outlet Temperature	DegF

Table A19 Variables in NN_14

# Sigs	Signal Name	Unit
1	Combustion Air Flow A1	Pct
2	Combustion Air Flow B1	Pct
9	Feedwater Flow #1 - Raw	Klb/hr
13	Hotwell Pumps Discharge Flow	Klb/hr
23	Measured Steam Flow - Raw	Klb/hr
25	Station Service Load	MW
34	First Stage Pressure A	psig
35	First Stage Pressure B	psig
36	Cold Reheat Pressure at Turbine	psig
40	FW Heater 1 Extraction Pressure	psig
41	FW Heater 2 Extraction Pressure	psig
42	FW Heater 3 Extraction Pressure	psig
43	FW Heater 4 Extraction Pressure	psig
45	Hot Reheat Pressure at Turbine	psig
50	SH Furnace Press After HT SH	InH2O
51	SH Furnace Press After Econ	InH2O
70	Pulv D Flow	Klb/hr
75	Air Preheater A Gas Inlet Temp	DegF
85	Cold Reheat Temp at Turbine	DegF
87	FW Heater 1 Extraction Temp	DegF
119	RH Attemp A Before Spray Temp	DegF
127	Auxiliary Air Damper AA Position	Pct
128	Auxiliary Air Damper AB Position	Pct
129	Auxiliary Air Damper BC Position	Pct
130	Auxiliary Air Damper CC Position	Pct

Table A20 Variables in NN_15

# Sigs	Signal Name	Unit
1	Combustion Air Flow A1	Pct
2	Combustion Air Flow B1	Pct
7	Condensate Flow - Raw	Klb/hr
9	Feedwater Flow #1 - Raw	Klb/hr
10	Feedwater Flow #2 - Raw	Klb/hr
13	Hotwell Pumps Discharge Flow	Klb/hr
23	Measured Steam Flow - Raw	Klb/hr
24	Unit Gross Generation	MW
25	Station Service Load	MW
34	First Stage Pressure A	psig
35	First Stage Pressure B	psig
36	Cold Reheat Pressure at Turbine	psig
40	FW Heater 1 Extraction Pressure	psig
41	FW Heater 2 Extraction Pressure	psig
42	FW Heater 3 Extraction Pressure	psig
43	FW Heater 4 Extraction Pressure	psig
44	FW Heater 5 Ext Press - Abs	psia
45	Hot Reheat Pressure at Turbine	psig
53	SH Windbox Pressure	InH2O
78	Air Preheater B Gas Inlet Temp	DegF
92	FW Heater 3 Drain Temperature	DegF
131	Auxiliary Air Damper DD Position	Pct
132	Auxiliary Air Damper DE Position	Pct
133	Auxiliary Air Damper EF Position	Pct
134	Auxiliary Air Damper FF Position	Pct

Appendix B. Network Training Performance

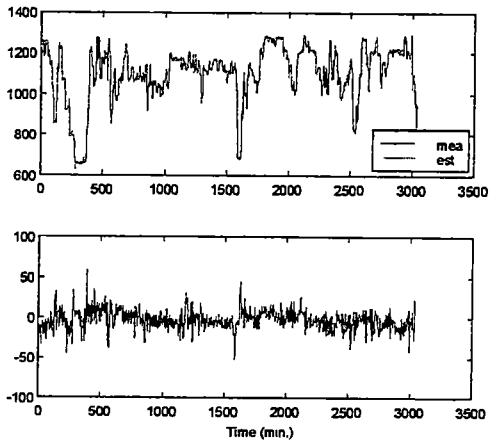


Figure B1 AANN recall on Measured Steam Flow (Klb/hr)

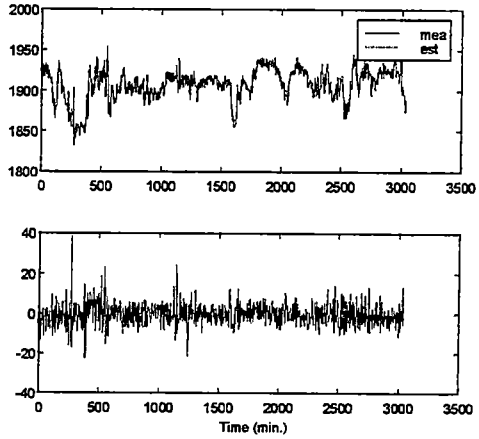


Figure B2 AANN recall on Selected Drum Pressure (psig)

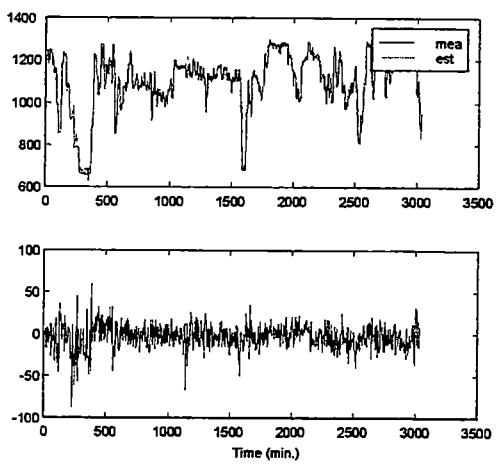


Figure B3 TDNN recall on Measured Steam Flow (Klb/hr)

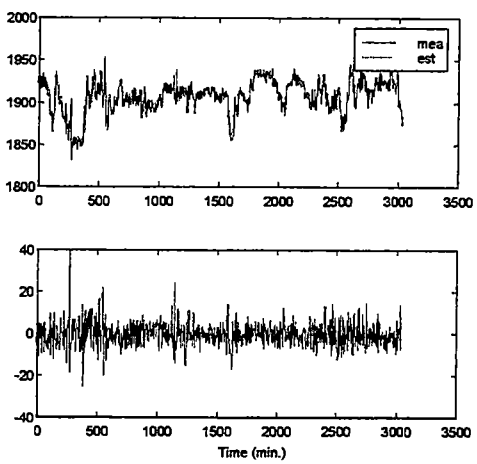


Figure B4 TDNN recall on Selected Drum Pressure (psig)

Note: The top plot of each figure is the signal measurement and model estimate, while the bottm plot is the residual between them.

Appendix C. System Detection Performance

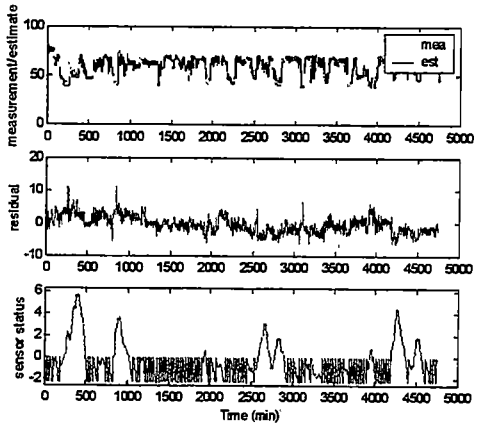


Figure C1 Recall on Combustion Air Flow (Pct)

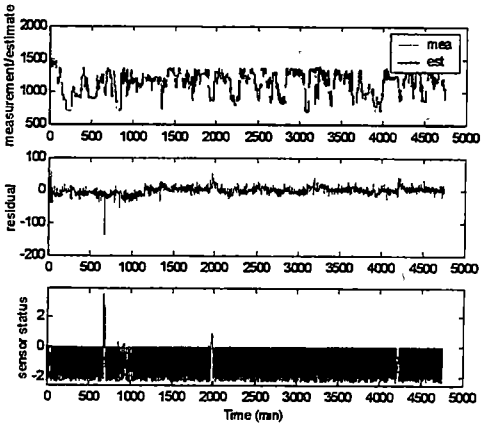


Figure C2 Recall on FW Flow (Klb/hr)

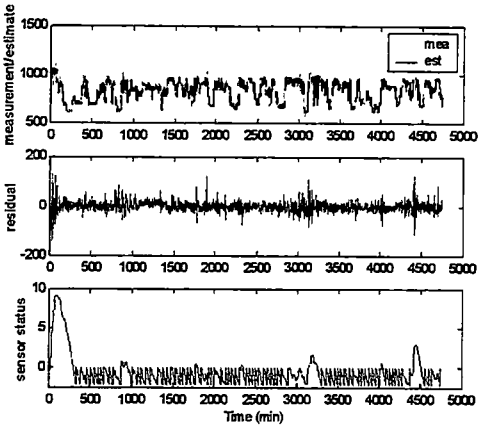


Figure C3 Recall on Hotwell Pump Discharge Flow (Klb/hr)

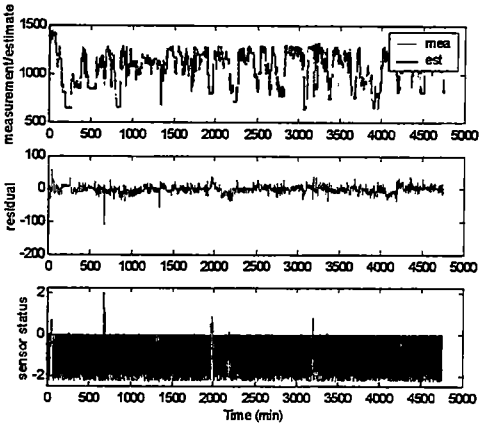


Figure C4 Recall on Measured Steam Flow (Klb/hr)

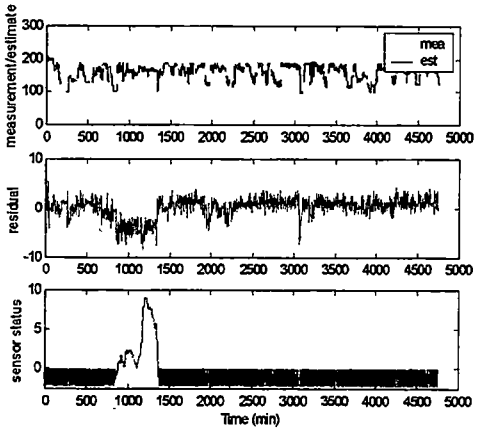


Figure C5 Recall on Unit Gross Generation (MW)

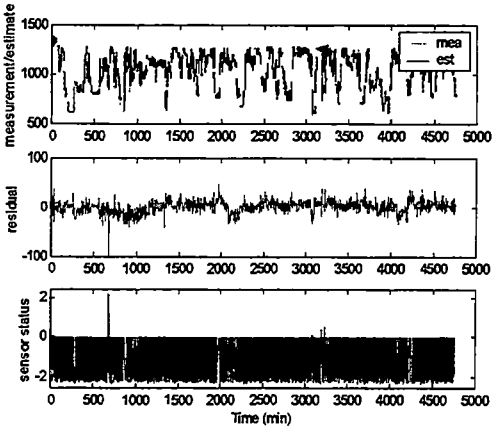


Figure C6 Recall on First Stage Pressure (psig)

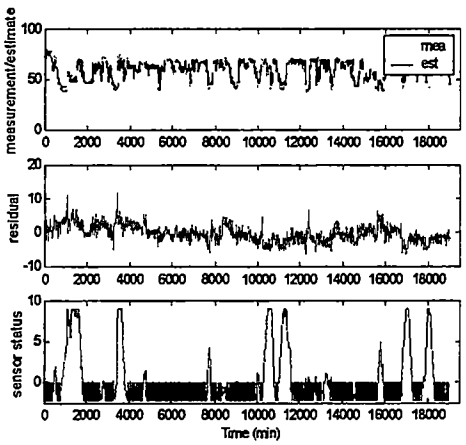


Figure C7 Prediction on Combustion Air Flow A1 (Pct)

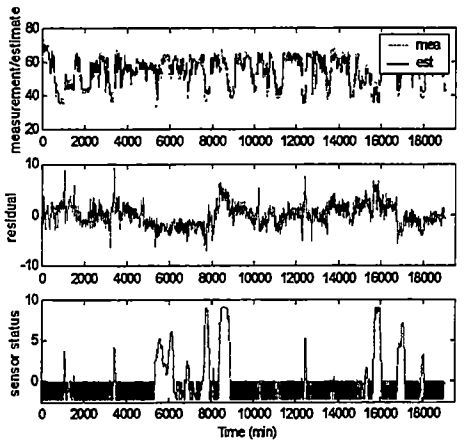


Figure C8 Prediction on Combustion Air Flow B1 (Pct)

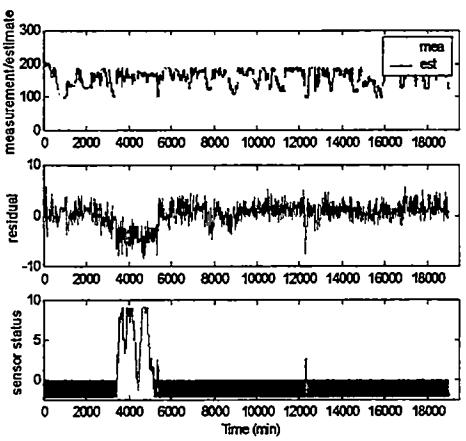


Figure C9 Prediction on Unit Gross Generation (MW)

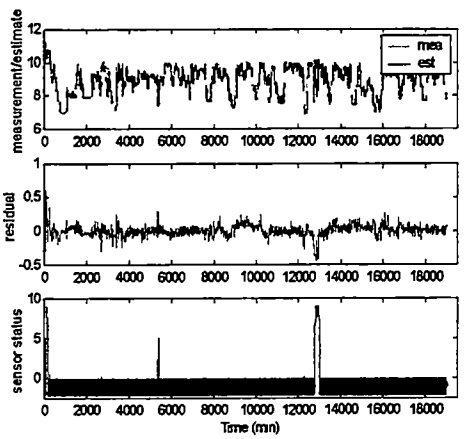


Figure C10 Prediction on Station Service Load (MW)

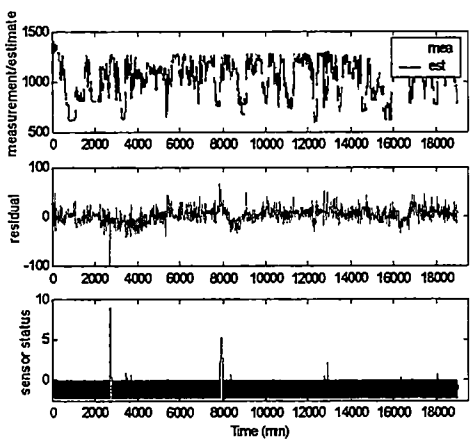


Figure C11 Prediction on First Stage Pressure A (psig)

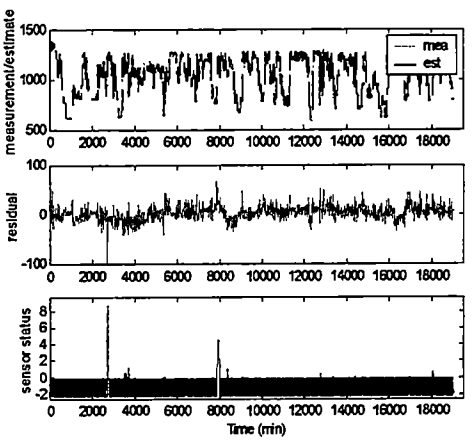


Figure C12 Prediction on First Stage Pressure B (psig)

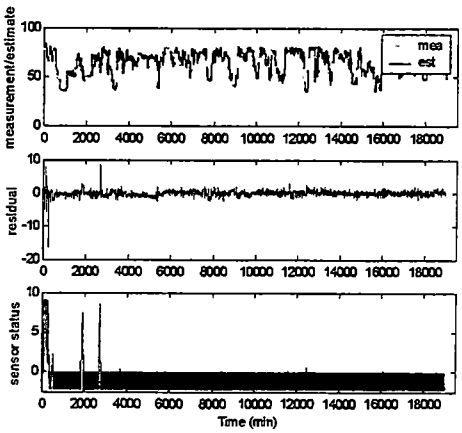


Figure C13 Prediction on Deaerator Pressure (psig)

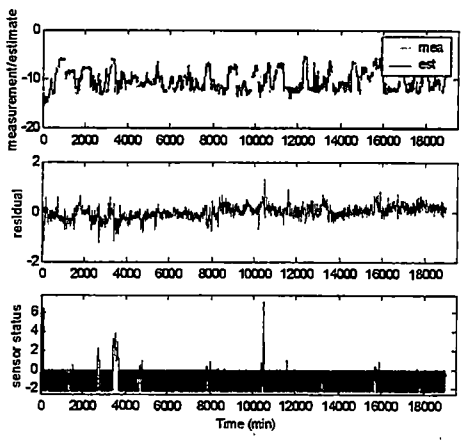


Figure C14 Prediction on ID Fan A Suction Pres. (InH2O)

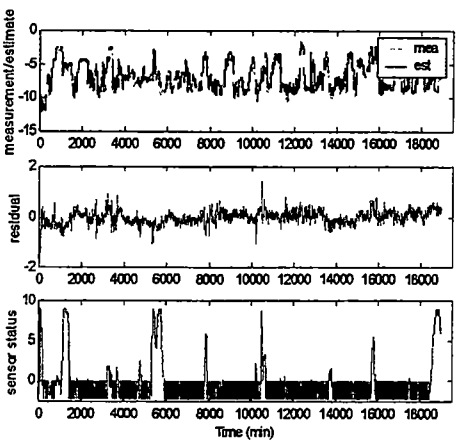


Figure C15 Prediction on ID Fan B Suction Pres. (InH2O)

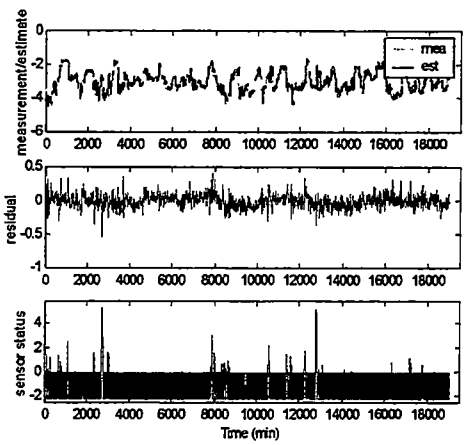


Figure C16 Prediction on RH Furnace Pres. after Econ (InH2O)

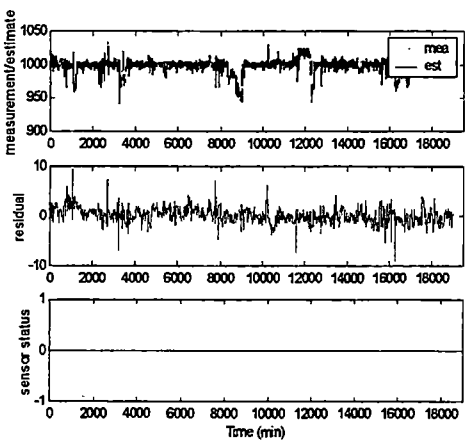


Figure C17 Prediction on SH Outlet Temp. #1 (DegF)

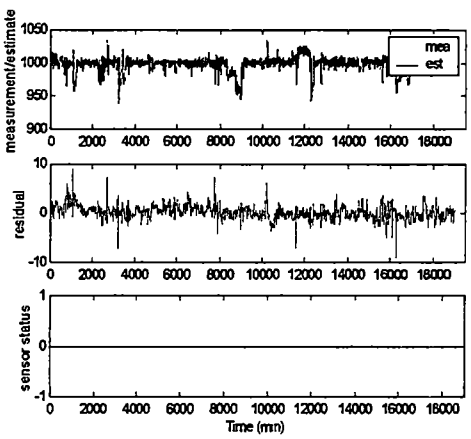


Figure C18 Prediction on SH Outlet Temp. #2 (DegF)

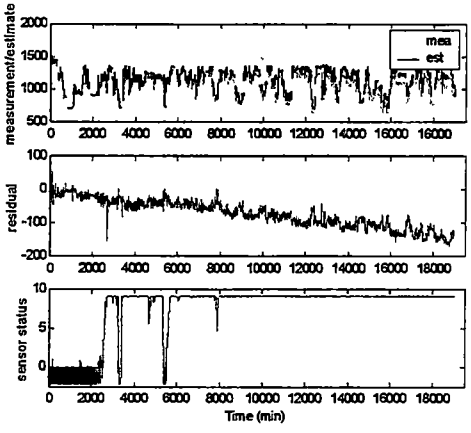


Figure C19 1%/day drift on FW Flow (Klb/hr)

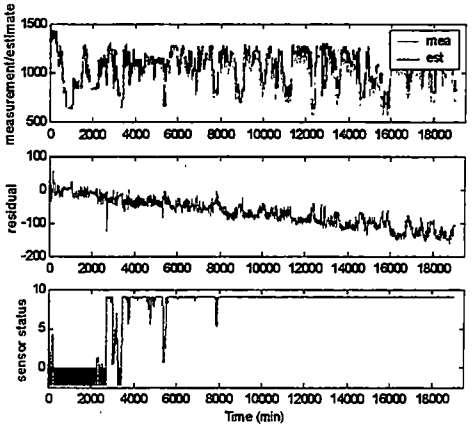


Figure C20 1% /day drift on Mea. Steam Flow (Klb/hr)

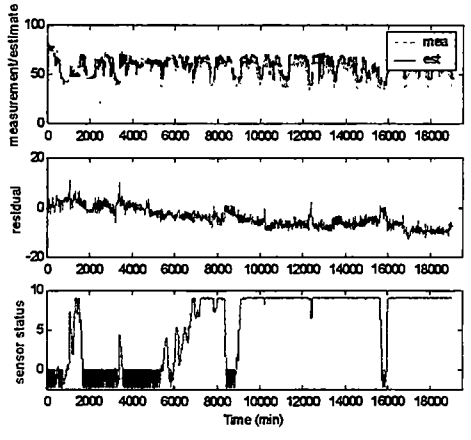


Figure C21 1% /day drift on Combustion Air Flow A1 (Klb/hr)

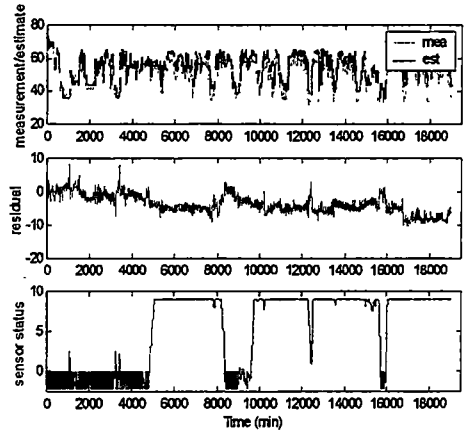


Figure C22 1% per day drift on Combustion Air Flow B1 (Klb/hr)

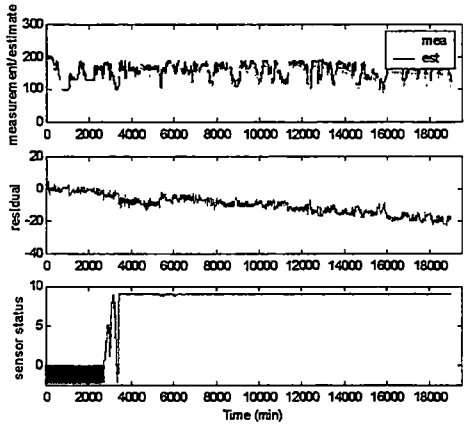


Figure C23 1%/day drift on Unit Gross Generation (MW)

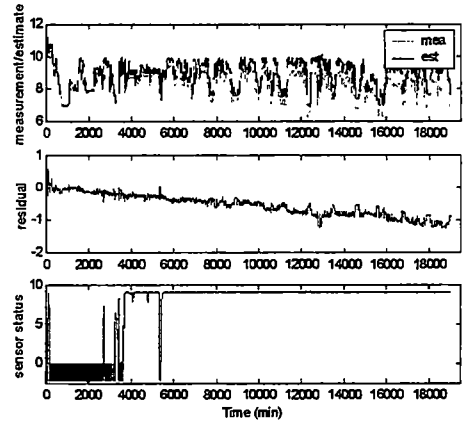


Figure C24 1%/day drift on Station Service Load (MW)

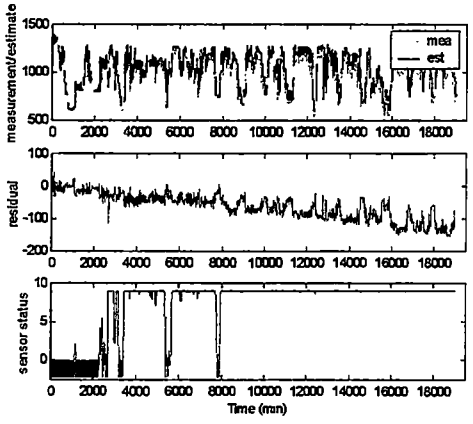


Figure C25 1%/day drift on First Stage Pressure A (psig)

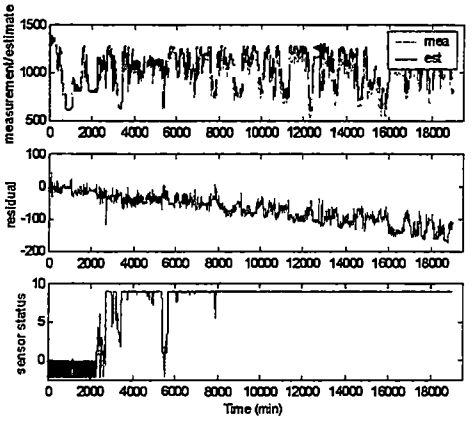


Figure C26 1%/day drift on First Stage Pressure B (psig)

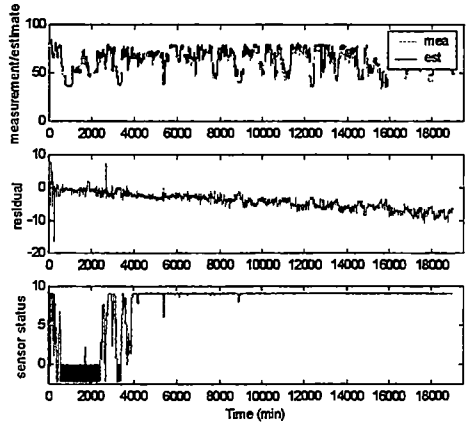


Figure C27 1%/day drift on Deaerator Pressure (psig)

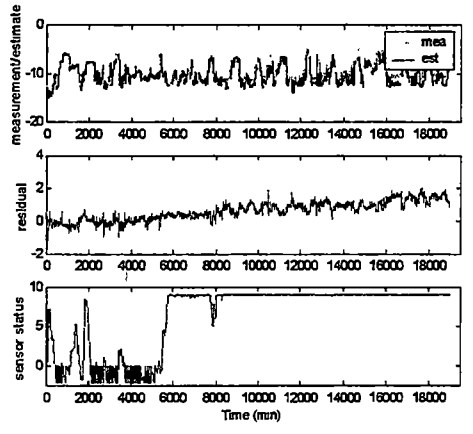


Figure C28 1%/day drift on ID Fan A Suction Pressure (InH2O)

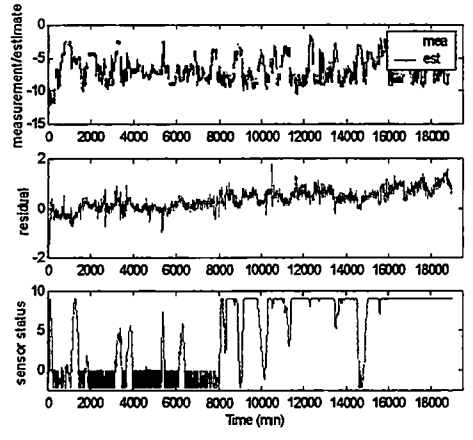


Figure C29 1%/day drift on ID Fan B Suction Pres. (InH2O)

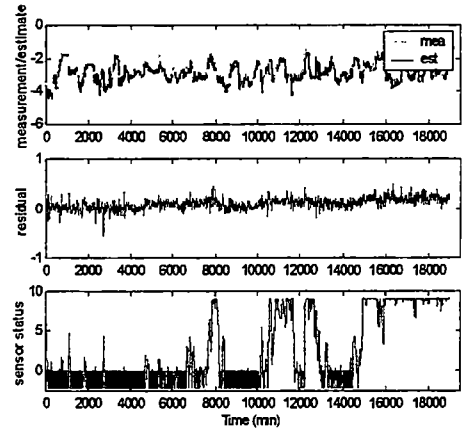


Figure C30 1% /day drift on RH Furnace Pres. after Econ (InH2O)

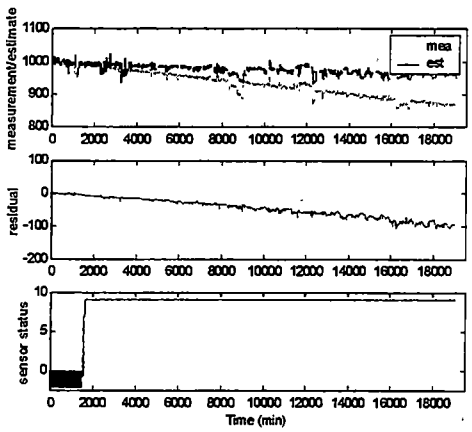


Figure C31 1%/day drift SH Outlet Temp. #1 (DegF)

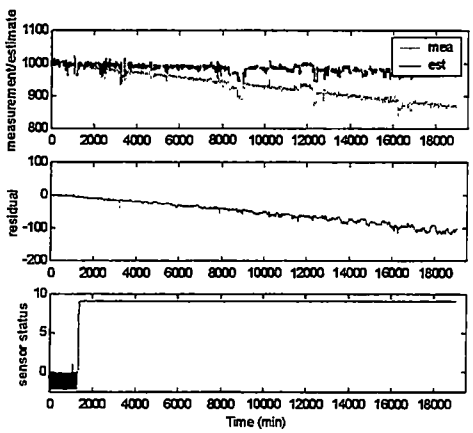


Figure C32 1%/day drift on SH Outlet Temp. #2 (DegF)

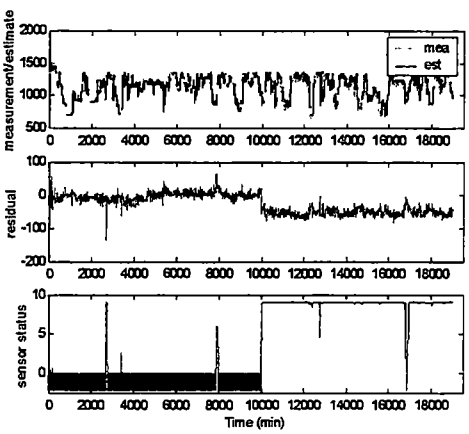


Figure C33 5% step drop on FW Flow (Klb/hr)

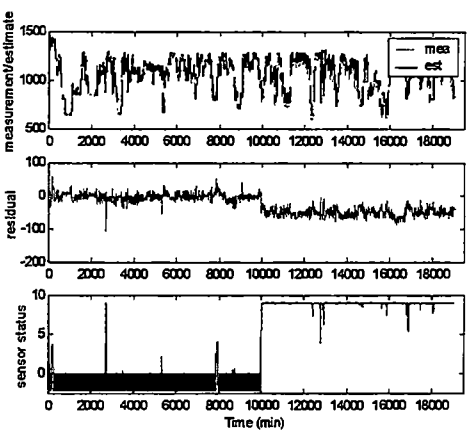


Figure C34 5% step drop on Measured Steam Flow (Klb/hr)

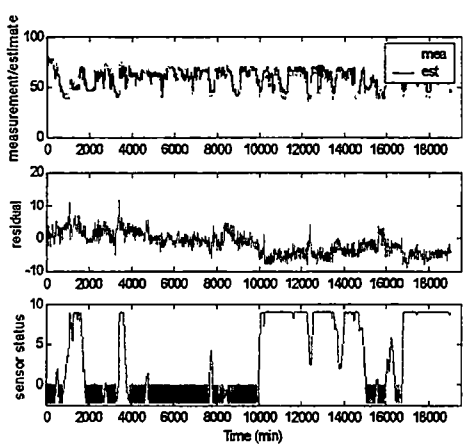


Figure C35 5% step drop on Combustion Air Flow (Pct)

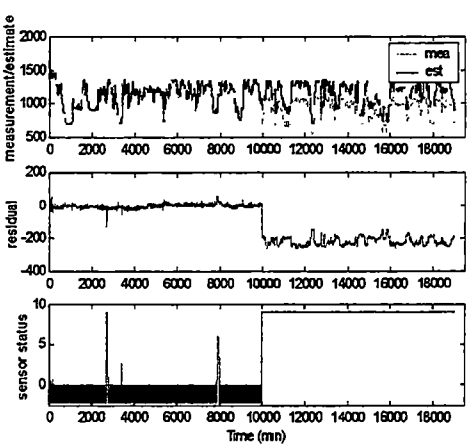
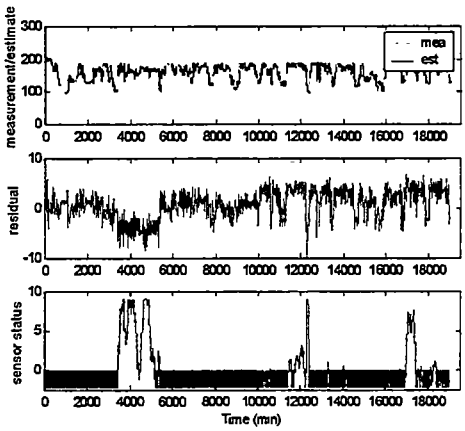


Figure C36 A 20% drop in FW Flow #1 (Klb/Hr)



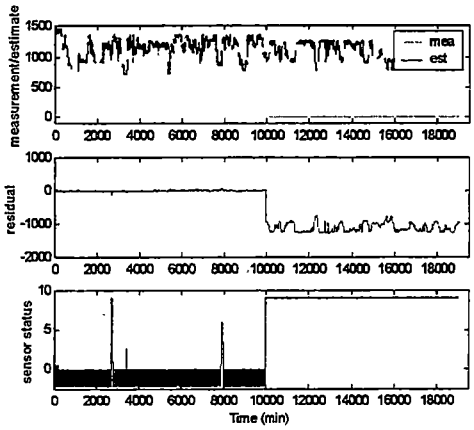


Figure C37 Predicted Unit Gross Gen. (MW) w/ bad FW Flow #1 Figure C38 Predicted lost FW Flow (Klb/hr) starting at 10000th min.

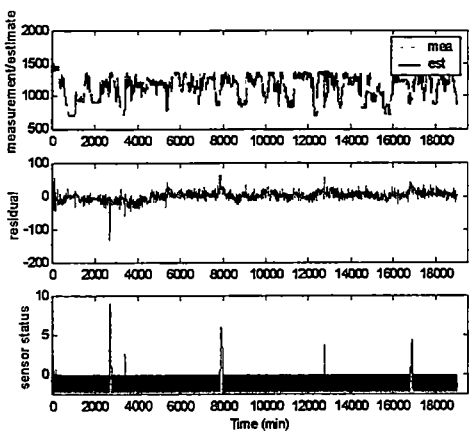


Figure C39 FW Flow (Klb/hr) after recovery

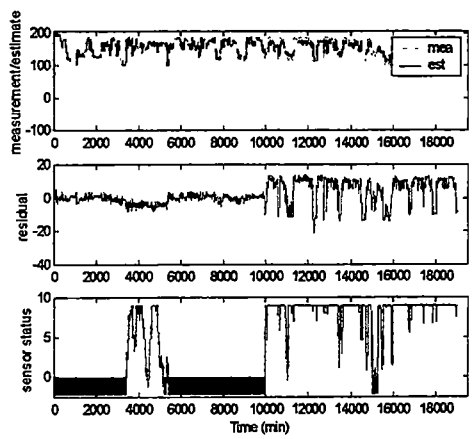


Figure C40 Predicted Unit Gross Gen. (MW) w/ faulty FW Flow #1

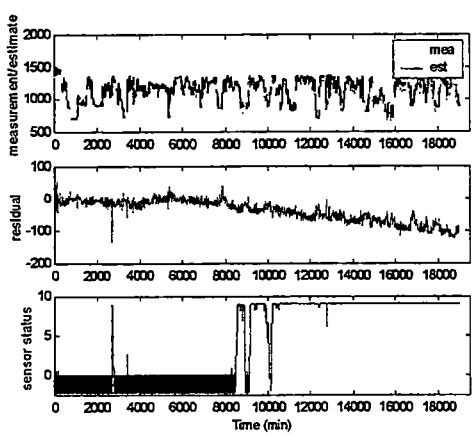


Figure C41 1%/day drift on FW Flow #1 (Klb/hr)

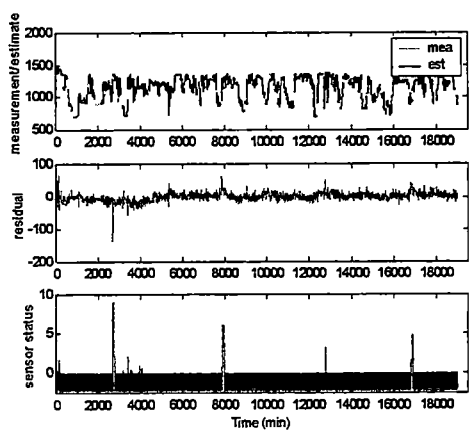


Figure C42 Corrected FW Flow #1 after replacement

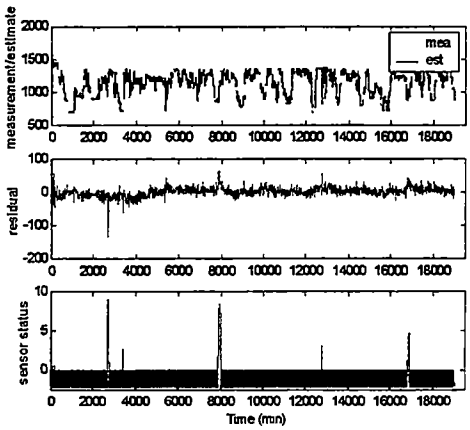


Figure C43 Corrected FW Flow #1 with 20% drop

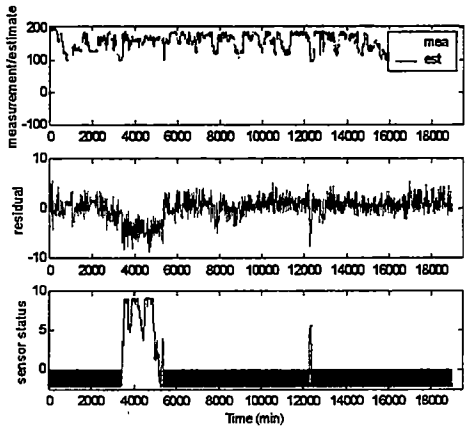


Figure C44 Predicted Unit Gross Gen. (MW) after replacement

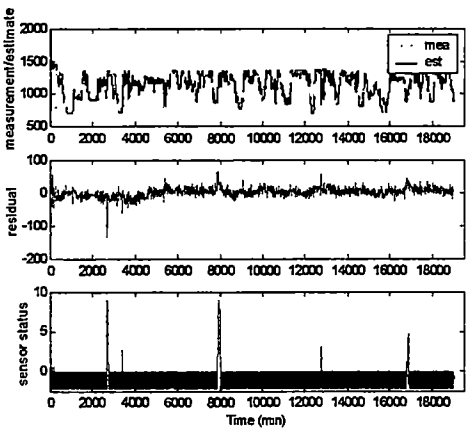


Figure C45 Recovered lost FW Flow #1

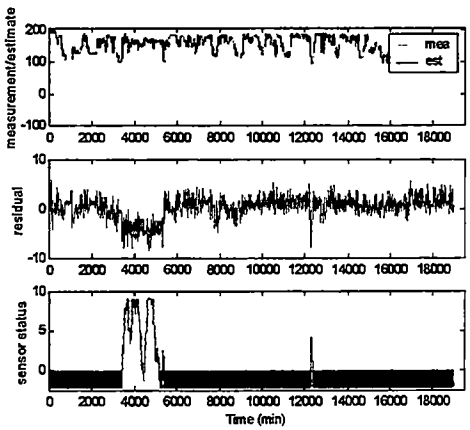


Figure C46 Predicted Unit Gross Gen. w/ recovered FW Flow #1

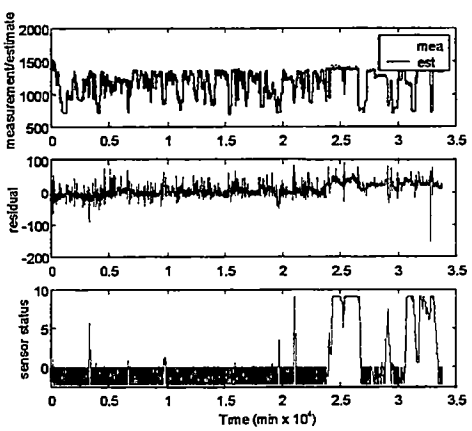


Figure C47 Predicted FW Flow (Klb/hr) before retuning

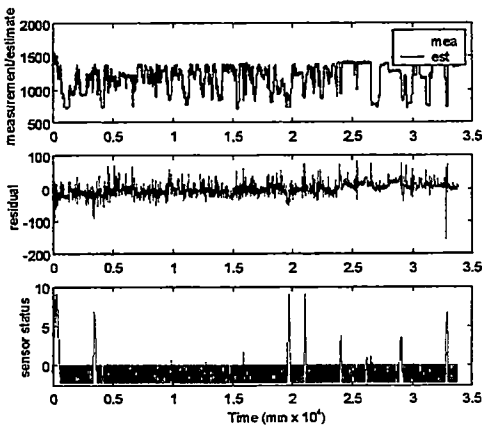


Figure C48 Predicted FW Flow (Klb/hr) after retuning

Appendix D. Noise Filtering Using SVD

The sensor measurement may contain noise due to system perturbation or environmental caused instrument channel perturbation. This noise will be associated with the neural network weight parameters and greatly affect the stability as well as the quality of the network. Therefore it should be removed to stabilize the network. A truncated SVD as a noise filter is applied for this purpose.

An example is presented here to show the mechanism of this technique. Three sine and cosine signals x, y and z are plotted in Figure D1.

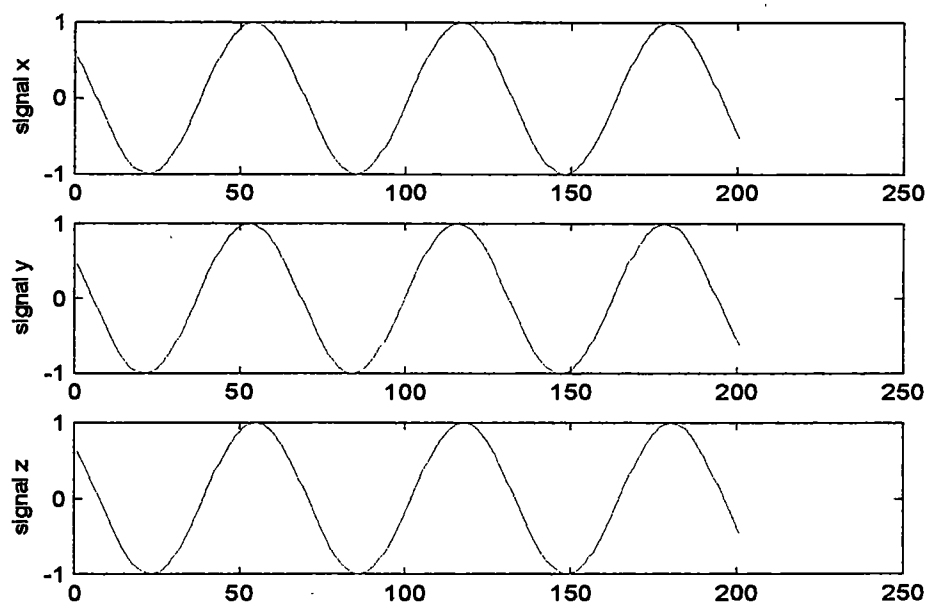


Figure D1 Exemplar perfect signals

It is noticed that signals are highly correlated shown in below.

```
1.0000    0.9945    0.9945

0.9945    1.0000    0.9782

0.9945    0.9782    1.0000
```

By adding normal distributed noise into the variables, we have the noisy signals xn, yn and zn plotted in Figure D2.

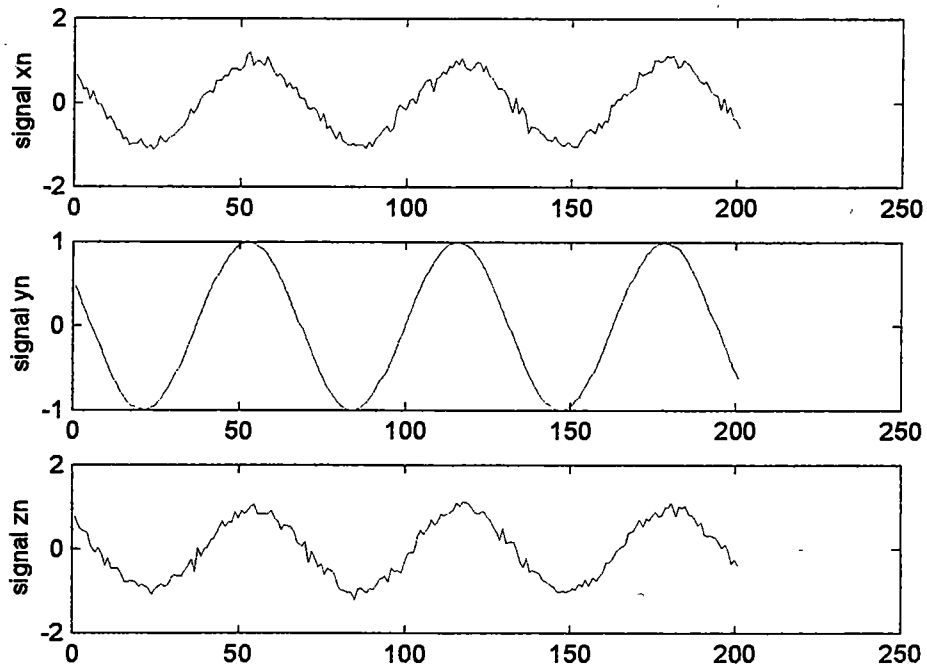


Figure D2 Exemplar noisy signals

We can see that the signals are distorted due to noise. For the convenience of data analysis, new vectors X and Xn are formed as:

```
X = [x;y;z]; Xn = [xn;yn;zn];
```

The noise levels for these signals are computed as:

```
res = Xn-X; level = std(res')

level =

    0.1016         0    0.0989
```

It is noticed that variables x and z are highly correlated, which can also be identified by the singular values *s* from SVD:

```
[u,s,v]=svd(X); S=[]; for i=1:min(size(X)), S=[S s(i,i)];end; S

S =

   16.8917    1.4485    0.0000
```

There is a big drop between 2nd and 3rd singular values, meaning that there are dependant vectors in X. The rank of X is deficient. We know that variables are highly correlated.

The important observation is that in the perfect data, the singular values drop off quickly, but in the noisy data, the singular values only gradually get smaller. Evidently, the noise in the signal prevents the singular values from dropping off. It is also noticed that the singular values increase with minor component having a singular value of 1.4187 compared with 0 for noise-free data.

```
[u,s,v]=svd(Xn); S=[]; for i=1:min(size(Xn)), S=[S s(i,i)];end; S

S =

   16.8236    1.8910    1.2695
```

An SVD-filtering process using a truncated SVD, which sets the small singular values caused by the noise to be zero, intends to remove the noise from the signals. Through the SVD-filtering process, the noise is removed, the reconstructed variables have much less noise than the noisy signals. Figures D3 shows this noise removal with 2 major singular values kept. The noise levels are decreased from 0.1016 and 0.0989 in the noisy data to

0.0429 and 0.0965 in the reconstructed data, respectively. However, the noise spreads through the SVD procedure; for example, the noise-free data was contaminated through the SVD procedure, the noise level is increased to 0.0508 in the reconstructed data. Therefore, the SVD procedure splits the noise to all signals.

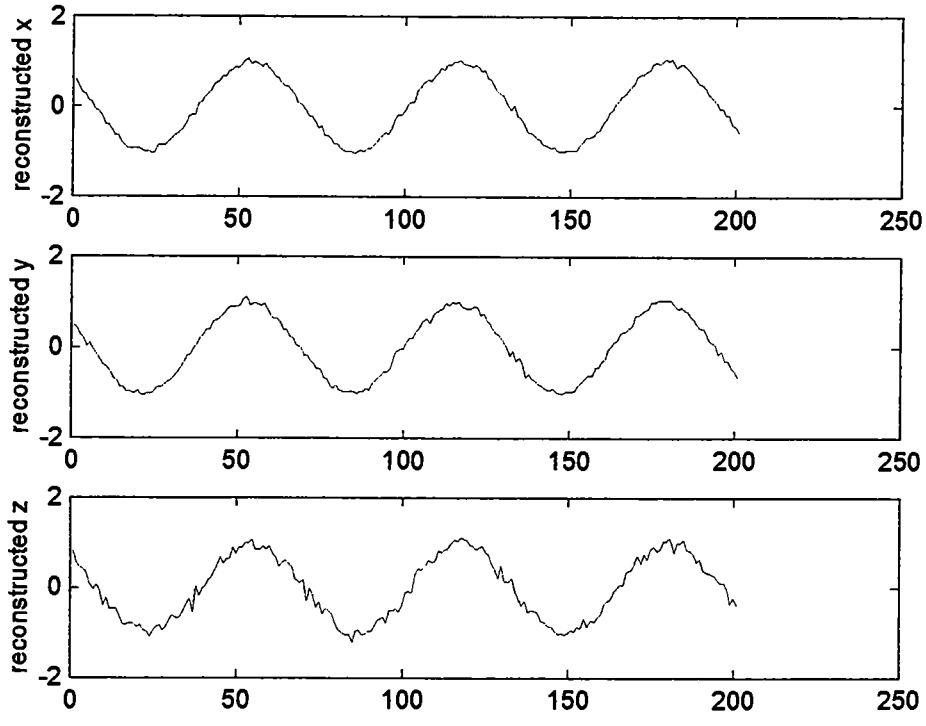


Figure D3 Reconstructed signals

```
res = Yn_new-X; level = std(res')

level =

    0.0429    0.0508    0.0965
```

Appendix E. NLPLS Algorithm

The NLPLS algorithm is constructed based on the NLPLS framework shown in Figure 2.1. The NLPLS algorithm is formulated as follows (Qin and McAvoy, 1992):

1. Scale X and Y to zero-mean and unit-variance. Let $E_0 = X, F_0 = Y$ and $h = 1$;

2. For each factor h, take u_h = some y_j;

3. PLS outer transform:

   - in matrix X:

     $w_h^T = \arg_{w_h} \min \left\| E_{h-1} - u_h w_h^T \right\|^2 = u_h^T E_{h-1} / u_h^T u_h$, normalize w_h to norm 1.

     $t_h = \arg_{t_h} \min \left\| F_{h-1} - t_h w_h^T \right\|^2 = E_{h-1} w_h$.

   - in matrix Y:

     $q_h^T = \arg_{q_h} \min \left\| F_{h-1} - u_h q_h^T \right\|^2 = u_h^T F_{h-1} / t_h^T t_h$, normalize q_h to norm 1.

     $u_h = \arg_{u_h} \min \left\| F_{h-1} - u_h q_h^T \right\|^2 = F_{h-1} q_h$.

     Iterate this step until it converges.

4. Calculate the X loadings and rescale the variables:

   $p_h^T = \arg_{p_h} \min \left\| E_{h-1} - t_h p_h^T \right\|^2 = t_h^T E_{h-1} / t_h^T t_h$

normalize p_h: $p_h \Rightarrow p_h/\|p_h\|$, $t_h \Rightarrow t_h\|p_h\|$, $w_h \Rightarrow w_h\|p_h\|$.

5. Find the inner network model: train the inner network such that the following error function is minimized.

   $J_h = \|u_h - f(t_h)\|^2$.

6. Calculate the residuals for factor h:

   for matrix X, $E_h = E_{h-1} - t_h p_h^T$,

   for matrix Y, $F_h = F_{h-1} - \hat{u}_h q_h^T$,

   where $\hat{u}_h = f(t_h)$.

7. Increment h by one, repeat Steps 2 through 6 until all principal factors are calculated.

Appendix F. Matlab Codes for System Design

F1 Neural Network Training with Cross Validation (Main Routine)

```
%
% autotrn.m
%
% A 4 layer NN training using cross-validation
% Use TSVD solving for bottleneck and output layer weights
% Network data group is generated by AUTOGROUP.m
%
% Xiao Xu,  Nov 11, 1999

clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  load training/testing data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('the data format should be Pattern*Sig_channels\n');
load work_data_2;
load Final_Group;
stdx = std(dat);
ind = find(stdx <= 0.02);
if ~isempty(ind)
   dat(:,ind) = [];
end
clear stdx ind;
dat = medfilt1(dat,3);

num_groups = size(final_group,1);
temp_string = sprintf('train which network?[1-%d]: ',num_groups);
nn_no = input(temp_string);
group = [];
for i = 1:length(final_group)
   if final_group(nn_no,i) ~= 0
      group = [group final_group(nn_no,i)];
   end
end

dat = dat(:,group);
dat_trn = dat(1:round(size(dat,1)*0.7),:);   % training data
dat_tst = dat(round(size(dat,1)*0.7)+1:size(dat,1),:);      % test data
clear dat;

fprintf(' There are %0.f patterns\n',size(dat_trn,1));
fprintf(' There are %0.f signals\n',size(dat_trn,2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  construct training/testing data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x = dat_trn;
x_p = perturb(x);
clear nr

i = 1:0.5*size(x_p,1);
x_pert = x_p(i,:);
x_free = row_subset(x_p,x_pert,i);
y_free = x_free;
x_tst = dat_tst;

%%%%%%%%%%%%%%%%%%%%
%  scale the data
%%%%%%%%%%%%%%%%%%%%

[yn_free,xm,xs] = prestd(y_free');
yn_free = yn_free';
xn_pert = trastd(x_pert',xm,xs)';
```

```
xn_free = trastd(x_free',xm,xs)';

clear dat_trn dat_tst;

percent=input('how many data do you want for training(%): ');
i = 1:round(percent*0.01*size(yn_free,1));
p = [xn_pert(i,:);xn_free(i,:)];
t = [yn_free(i,:);yn_free(i,:)];

[v.P_free,ind] = row_subset(xn_free,xn_free(i,:),i);
[v.P_pert,ind] = row_subset(xn_pert,xn_pert(i,:),i);
[v.T,ind] = row_subset(yn_free,yn_free(i,:),i);

v.P = v.P_free';
v.T = v.T';

[nov1,numin] = size(p);                          % p are input vectors
[nov2,numout] = size(t);                 % t are target vectors
if nov1 ~= nov2
   error('The rows of input and target vectors must be same.')
end
p = p';
t = t';

%------------------------------
%  calculate # of hidden nodes
%------------------------------

M_numhid1 = [];
for j = 1:numin
   M_numhid1 = [M_numhid1 numhid_cal(xn_free(i,:),yn_free(i,j))];
end
numhid1 = max(M_numhid1);
clear M_numhid1;
numhid2 = bottle(xn_free(i,:));
numhid3 = numhid1;

clear x x_free y_free x_pert x_p xn_free yn_free xn_pert;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Printout network parameters.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\nThis network has:\n\n');
fprintf(' %0.f input neurons\n',numin);
fprintf(' %0.f neurons in the 1st hidden layer\n',numhid1);
fprintf(' %0.f neurons in the 2nd hidden layer\n',numhid2);
fprintf(' %0.f neurons in the 3rd hidden layer\n',numhid3);
fprintf(' %0.f output neurons\n\n',numout);
time_start = cputime;

f1 = 'tansig';          % activation functions in each layer
f2 = 'purelin';
f3 = f1;
f4 = f2;

trial = 0;
trained = 'false';
trn_fns = input('select training function (1)traingdx (2)traincgf: ');
if trn_fns == 1
   trn_fn = 'traingdx';
else
   trn_fn = 'traincgf';
end
obj_fn = 'msereg';

while trial == 0
  fprintf('There are %0.f input/output pairs in this training set.\n\n',nov1);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%
%  Initialize network. %
%%%%%%%%%%%%%%%%%%%%%%%%%

net = newff(minmax(p),[numhid1 numhid2 numhid3 numout],...
       {f1 f2 f3 f4},trn_fn,'',obj_fn);
if strcmp(trained, 'false')
    net.performParam.ratio = input('enter performance ratio: ');
end

if strcmp(trained, 'false')                % set small initial weights
   net.IW{1,1} = 0.1*randn(size(net.IW{1,1}));
   net.LW{2,1} = 0.1*randn(size(net.LW{2,1}));
   net.LW{3,2} = 0.1*randn(size(net.LW{3,2}));
   net.LW{4,3} = 0.1*randn(size(net.LW{4,3}));
else
   net.IW = IW; net.LW = LW; net.b = b;
   clear IW LW b;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Set up training parameters.   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

net.trainParam.epochs = 9;  % Maximum number of training epochs = 9
net.trainParam.show = 1;    % Errors plotted in progress every 1 epoch
net.trainParam.goal = .05;  % Training error goal = 0.05

if strcmp(trn_fn,'traingdx')
   net.trainParam.lr = 0.01;        % Learning rate = .01
   net.trainParam.mc = 0.95;        % Momentum = 0.95
   net.trainParam.lr_inc = 1.05;    % Learning rate increase = 1.05
   net.trainParam.lr_dec = 0.7;     % Learning rate decrease = 0.70
   net.trainParam.max_perf_inc = 1.04;     % Error Ratio = 1.04
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     train the network using cross-validation and SVD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clf
for cycle = 1:100
   [net,tr] = cvtrain(net,p,t);
   nntwarn off
   a1 = feval(f1,net.IW{1,1}*p,net.b{1});  % demapping layer output
   nntwarn off
   a2 = feval(f2,net.LW{2,1}*a1,net.b{2}); % bottleneck layer output
   [w2,b2] = tsvd(a1,a2);
   net.LW{2,1} = w2; net.b{2} = b2; % retune w/b using SVD
   nntwarn off
   a2 = feval(f2,net.LW{2,1}*a1,net.b{2}); % bottleneck layer output
   nntwarn off
        a3 = feval(f3,net.LW{3,2}*a2,net.b{3});    % mapping layer output
        [w4,b4] = tsvd(a3,t);              % call SVD algorithm
   net.LW{4,3} = w4; net.b{4} = b4;
   if (w4-net.LW{4,3}) < 1e-5*ones(size(w4)) | ...
         tr.perf(size(tr.perf,2)) <= net.trainParam.goal
      fprintf('best weights found.\n')
      break
   end
end
clear a1 a2 a3 w2 b2 w4 b4;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  preserve the trained weights/biases
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

IW = net.IW; LW = net.LW; b = net.b; trained = 'true';
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   the percent error goal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pct_goal = 0.01;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   test validation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[V_hat] = sim(net,v.P);
V = poststd(v.T,xm,xs);
V_hat = poststd(V_hat,xm,xs);
err_pct = (V-V_hat)./V;
sum_err = sum(abs(err_pct));
fprintf('The test error goal is %f\n', pct_goal*size(V,1));
fprintf('The max test error is %f\n', max(sum_err));
if max(sum_err) > pct_goal*size(V,1)
   fprintf('Add data to training set\n');
else
   trial = -1;
end
clear V V_hat err_pct;

if max(sum_err) > pct_goal*numout
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %  find the number of points with large error
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   index = find(sum_err > pct_goal*numout);
   if isempty(index)
      trial = -1;
      break;
   else
       number = size(index,2);
   end
   p = p';
   t = t';
   adding = 'y';
   point = 0;        % start adding data points
   while strcmp(adding,'y')
      index = max(find(sum_err > pct_goal*numout));
      if ~isempty(index)
         v.P = v.P';
         v.T = v.T';
         p = [p;v.P_pert(index,:);v.P_free(index,:)];
         t = [t;v.P_free(index,:);v.P_free(index,:)];
         [v.P_free,ind] = row_subset(v.P_free,v.P_free(index,:),index);
                  [v.P_pert,ind] = row_subset(v.P_pert,v.P_pert(index,:),index);
         [v.T,ind] = row_subset(v.T,v.T(index,:),index);
         v.P = v.P_free';
         v.T = v.T';
         sum_err = sum_err(:,ind);

         point = point+1;
         if (number > 10 & point <= round(0.1*number)) | (number < 10)
            adding = 'y';
         else
            adding = 'n';
         end
      else
         adding = 'n';
      end
   end
   [nov1,numin] = size(p);
   [nov2,numout] = size(t);
   p = p';
   t = t';
```

```
  end
end

if nn_no == 1
   save autowb_G1 net xm xs
elseif nn_no == 3
   save autowb_G3 net xm xs
end

fprintf('weights have been saved\n')';

%%%%%%%%%%%%%%%%%%
%  test network
%%%%%%%%%%%%%%%%%%

tst = input('do network test?: ','s');
if tst == 'y'
   [y,res,err_pct,fmean,vr] = nn_test(net,x_tst,xm,xs);
end

%%%%%%%%%%%%%%%%%%%%%%%%
%  sensitivity analysis
%%%%%%%%%%%%%%%%%%%%%%%%
sa = input('do network sensitivity analysis?: ','s');
if sa == 'y'
   p_free = p(:,0.5*size(p,2)+1:size(p,2));
   x_trn = poststd(p_free,xm,xs)';
   sen_test(net,x_trn,xm,xs);
end
```

F2 Variable Grouping Finalization

```
%  Finalize the groups initially obtained
%  from AUTOGROUP.M by eliminating variables
%  having poor correlations with others
%  pairwise-wide.
%

%---------------------------------
%  remove little changed variables
%---------------------------------

clear all
load work_data_2;
stdx = std(dat);
ind = find(stdx <= 0.02);
if ~isempty(ind)
   dat(:,ind) = [];
end
clear stdx ind;
dat = medfilt1(dat,3);
%  calculate correlation matrix
cc = corrcoef(dat);
init_group = 1:length(cc);
clear dat;

cases = input('1)remove 2)not remove grouped vars: ');
max_num = input('max # of vars per group: ');

%----------------------------
%  input interested variables
%----------------------------

Inst_var = [1 2 23 24 34 35 37 46 47 48 124 125];
```

```
%-----------------------------------------------
%  regrouping to remove unmonitorable variables
%-----------------------------------------------

rem_var = [];
continue  = 'yes';
while strcmp(continue, 'yes')
   final_group = autogroup(init_group,cc,cases,max_num,Inst_var);
   if cases == 2
      loop_rem_var = 0;
      for row = 1:size(final_group,1)
         count = 0;
         for col = 1:size(final_group,2)
            count = count+any(final_group(row,col));
         end
         if count <= 5 & count ~= 0
            rem_var = [rem_var final_group(row,1)];
            loop_rem_var = final_group(row,1);
       end
       end
      if ~isempty(rem_var) & loop_rem_var ~= 0
         init_group = remover(init_group,rem_var);
      else
         fprintf('Grouping finally done\n');
         break;
      end
   end
end

%------------------------
%  finalize final_group
%------------------------

i = 1;
max_num_row = size(final_group,1);
while i <= max_num_row
   if final_group(i,1) == 0
      final_group(i,:)=[];
      max_num_row = size(final_group,1);
   else
       i = i+1;
   end
end

%-------------------
%  sort the results
%-------------------

temp_final_group = zeros(size(final_group));

for row = 1:size(final_group,1)
   temp_final_group(row,:) = sort(final_group(row,:));
end

rem_var = sort(rem_var);

final_group = temp_final_group;
clear row temp_final_group

save Final_Group final_group;
save Rem_Var rem_var;
```

F3 Automatic Variable Grouping Algorithm

```
function Group = autogroup(init_group,cc,cases,max_numbers,Inst_var)
```

```
%
% Use correlation analysis to intially group the variables
%
%-----------------
%  initial setups
%-----------------

master_group = init_group;
ungrouped = init_group;
grouped = [];
Group = zeros(40,max_numbers);

%-----------------
%  start grouping
%-----------------

grouping = 0;
gg_process= 'y';
while strcmp(gg_process, 'y')

   if isempty(Inst_var)
      if ~isempty(ungrouped)
         Inst_var = ungrouped;
      else
        fprintf('process finish.\n');
         break;
      end
   end

   grouping = grouping+1;
   fprintf('%d group(s) formed\n', grouping);
   id = find(abs(cc(Inst_var(1),init_group)) >= 0.5);
   group = init_group(id);
   clear id;

   %  remove low correlated variables

   temp_group = group;
   for i = 2:length(group)
       count = 0;
      for j = 1:length(group)
         if abs(cc(group(i),group(j))) < 0.5
               count = count+1;
         end
               if count >round(0.1*length(group))
               fprintf('signal %d is excluded\n',group(i));
            temp_group(i) = 0;
            break;
               end
       end
   end
   group = sort(temp_group);
   if isempty(group)
      gg_process= 'n';
      break;
   end

   ind = find(group == 0);
   group(ind) = [];
   clear temp_group;

   %--------------------------------------
   %  find the optimal combination groups
   %--------------------------------------

   [sort_cc,id] = sort(cc(Inst_var(1),group));
   id = id(length(id):-1:1);  % sort cc from high to low
   group = group(id);
```

```
    clear sort_cc;

    if length(group) > max_numbers     %  group signals
       Group(grouping,1:max_numbers) = group(1:max_numbers);
       start_pos = find(id == Inst_var(1));
       if start_pos > max_numbers
          Group(grouping,1:max_numbers) = group([start_pos 1:(max_numbers-1)]);
       end
    else
        Group(grouping,1:max_numbers) = [group zeros(1,max_numbers-length(group))];
    end

    inc_var_id = [];
    for i = 1:length(Inst_var)
       inc_var_id = [inc_var_id find(Group(grouping,:) == Inst_var(i))];
    end

    %  ------------------------------------
    %  remove grouped interested variables
    %  ------------------------------------

    Inst_var = remover(Inst_var,Group(grouping,inc_var_id));
    if cases == 1
       init_group = remover(init_group,Group(grouping,:));
    end

    %  ------------------------
    %  remove common variables
    %  ------------------------

    grouped = [grouped Group(grouping,:)];

    grouped = sort(grouped);
    for i = 1:length(grouped)-1
       if grouped(i+1) == grouped(i)
          grouped(i) = 0;
       end
    end

    redundancy_id = find(grouped == 0);
    grouped(redundancy_id) = [];

    %  ----------------------------
    %  regroup remaining variables
    %  ----------------------------

    ungrouped = remover(master_group,grouped);
    ungrouped = sort(ungrouped);

end
```

F4 Variable Removal Algorithm

```
function x = remover(x,sub_x)

%  Remove variables sub_x from variable pool x

sort_sub_x = sort(sub_x);

for i = 1:length(sort_sub_x)
    id = find(x == sort_sub_x(i));
    x(id)=[];
end
```

F5 Sample Selection Algorithm

```
function [v,ind] = row_subset(x,xi,index)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% row_subset.m:
%
%  Generate a subset of data from the
%  original data set.
%
% [V,IND]=ROW_SUBSET(X,XI,INDEX)
%
% X -- original data set;
% XI -- selected data set;
% V -- generated sub data set;
%
% Xiao Xu,  10/28/96

[nr,nc]=size(x);
[xr, xc]=size(xi);
if nc ~= xc, error('matrix dimension must agree'); end
y=[];
if index(1) ~= 1
   y=[y;zeros(index(1)-1,1)];
end
for i=1:xr-1
    delta=index(i+1)-index(i);
    if delta > 1
       y=[y; xi(i,1); zeros(delta-1,1)];
    else
       y=[y; xi(i,1)];
    end
end
y=[y; xi(xr,1)];
[yr,yc]=size(y);
if yr < nr
    delta=nr-yr;
    y=[y; zeros(delta,1)];
end
ind=find(y~=x(:,1));
v=x(ind,:);
```

F6 Robust Training Algorithm

```
function  x_pert = perturb(x)

% PERTURB
% Trains an autoassociative network using faults in the input set
% and error free data in the output set. Random fault values (~10%)
% of the operation level are placed on the diagonals of the input
% matrix.
%
% Xiao Xu                    University of TN - Knoxville

[nr,nc] = size(x);          % Determine size of input data set.

min_max_value = minmax(x');
range = min_max_value(:,2)-min_max_value(:,1);
pert = range';
X = x;

j = 0;                      % Set column counter to zero.
for i = 1:nr                % For row i, column j, add random fault.
```

```
   j = j+1;
   x(i,j) = x(i,j)+0.2*(rand(1)-0.5)*pert(:,j); % Random fault of +-1%.
        if j == nc              % If at end of column reset to first column.
        j = 0;
        end
end
x_pert = [x;X];
```

F7 Calculation of Number of Mapping/Demapping Layer Nodes

```
function numhid = numhid_cal(x,y)

%  estimate the required number of network hidden
%  nodes based on least squares technique


if length(x) > 1000
   int = round(length(x)/1000);
   if size(x,1) < size(x,2)
      x = x(:,1:int:length(x));
      y = y(:,1:int:length(y));
      x = x';
      y = y';
   else
      x = x(1:int:length(x),:);
      y = y(1:int:length(y),:);
   end
end

[u,s,v] = svd(x,0);


%% find the truncated point in matrix s
sum_s = 0;

for i = 1:size(s,2)
   sum_s = sum_s+s(i,i);
end
for j = 1:size(s,2)
   nor_s(j)=s(j,j)./sum_s;            % normalized singular values
end
info = [];
for k = 1:length(nor_s)
   info = [info;sum(nor_s(1:k))];
end
cut_off = .9;
m = max(find(info <= cut_off));
if m < length(nor_s)
        if abs(info(m)-cut_off) > abs(info(m+1)-cut_off)
        m = m+1;
        end
end
s = s(1:m,1:m);
u = u(:,1:m);
v = v(:,1:m);

w = v*inv(s)*u'*y;                    % estimate solution

y_hat = x*w;

RSE = sqrt(sse(y-y_hat));


%-----------------------------
%  calculate error criterion
```

```
%----------------------------

c = sqrt(sum((y*.01).^2)/length(y));

%------------------------------
%  calculate # of hidden nodes
%------------------------------

numhid = round(5*log(RSE/c));
```

F8 Calculation of Number of Bottleneck Layer Nodes

```
function m = bottle(x)

%  estimate the number of bottleneck layer nodes


if length(x) > 1000
   int = round(length(x)/1000);
   if size(x,1) < size(x,2)
      x = x(:,1:int:length(x));
      x = x';
   else
      x = x(1:int:length(x),:);
   end
end

[u,s,v] = svd(x,0);

%%% find the truncated point in matrix s
sum_s = 0;

for i = 1:size(s,2)
   sum_s = sum_s+s(i,i);
end
for j = 1:size(s,2)
   nor_s(j)=s(j,j)./sum_s;            % normalized singular values
end
info = [];
for k = 1:length(nor_s)
   info = [info;sum(nor_s(1:k))];
end
cut_off = .95;
m = max(find(info <= cut_off));
if m < length(nor_s)
        if abs(info(m)-cut_off) > abs(info(m+1)-cut_off)
        m = m+1;
        end
end
```

F9 Training with Cross Validation Algorithm

```
function [net,tr] = cvtrain(net,x,y)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[NET, TR] = CVTRAIN(Net,X,Y);

%       Train a neural network using cross validation stopping
%  criteria. This is designed specifically for robust training.
%

%       X: R*Q input matrix, rows are patterns and columns are inputs
```

```
%       y = target outputs, rows are patterns and columns are outputs

%  Xiao Y., Oct 15, 1999

x = x';
y = y';
[nov1,numin] = size(x);
[nov2,numout] = size(y);

if nov1 ~= nov2
   error('The number of input and target vectors has to be the same.')
else
   nov = nov1;
end
clear nov1 nov2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pick up the noisy & noise-free points in pairs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i = 1:2:nov/2;                  % indices of noisy points
j = nov/2+i;                    % indices of noise-free points
id = [i j];

p = x(id,:);                                    % input data
t = y(id,:);                                    % target data

[v.P,ind] = row_subset(x,p,id);                 % validation data
[v.T,ind] = row_subset(y,t,id);

p = p';
t = t';
v.P = v.P';
v.T = v.T';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Train the Network
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[net,tr] = train(net,p,t,[],[],v);
```

F10 Neural Network Performance Testing Algorithm

```
function [y,res,err_pct,fmean,vr] = nn_test(net,dat,xm,xs);

% nn_test.m
%
% Network test program for TVA project
% If operation condition changes, GVD used for
% network retuning.
%
% Xiao Yu
% Jan 19,1999

clf

drift = input('Simulate drift?[y,n]: ','s');
if drift == 'y'
   undrift = dat;
   disp('1. ramp drift;')
   disp('2. step drop;')
   disp('3. momentarily step drop.')
   change_type = input('your choice: ');
   sig = 0;
   while sig >= 0
```

```
        str = sprintf('drifting which signal?[1-%d,-1 for no more drift sigs]:
',size(dat,2));
      sig = input(str);
      if sig < 0
         break;
      end
      pct = input('enter the percentage of drift: ');
      start = input('start at ');
       if change_type == 3
       finish = input('finish at ');
       end
       if change_type == 1
       for i = start:size(dat,1)
              dat(i,sig) = dat(i,sig)-0.01*pct/(24*60)*(i-start)*dat(i,sig);
       end
       elseif change_type == 2
       dat(start:size(dat,1),sig) = dat(start:size(dat,1),sig)*(1-pct*0.01);
       else
       dat(start:finish,sig) = dat(start:finish,sig)*(1-pct*0.01);
      end
   end
end

nr = size(dat,1);
xn = trastd(dat',xm,xs);

f1='tansig'; f2='purelin';f3=f1;f4=f2;

adapt = input('system condition change?: ','s');
if strcmp(adapt,'y')           % SVD retunes the network
   if size(xn,2) > 3000              % reduce matrix size for SVD calculation
      int = round(size(xn,2)/3000);
      red_xn = xn(:,1:int:size(xn,2));
   else
      red_xn = xn;
   end
   nntwarn off
   a1 = feval(f1,net.IW{1,1}*red_xn,net.b{1});
   nntwarn off
   a2 = feval(f2,net.LW{2,1}*a1,net.b{2});
   nntwarn off
      a3 = feval(f3,net.LW{3,2}*a2,net.b{3});
      [w4,b4] = tsvd(a3,red_xn);                  % call SVD algorithm
   net.LW{4,3} = w4; net.b{4} = b4;
end
X = sim(net,xn);
y = poststd(X,xm,xs)'; % unscaled output

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Determine residual variance and faulted means
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
scale = 1;
if strcmp(drift,'n')
      [data_mean,sigma] = sprt_par(net,dat,xm,xs);
else
      [data_mean,sigma] = sprt_par(net,undrift,xm,xs);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  calculate mean error percentage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

res = dat-y;
err_pct = mean(abs(res./y))*100;

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  set up SPRT model
%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
drawCmd = 'y';
while drawCmd == 'y'
   str = sprintf('which signal you want to plot[1-%d][0: exit]? ',size(dat,2));
   signal = input(str);
   if signal > size(dat,2) | signal < 0
      disp('input out of range !!!')
      str = sprintf('which signal you want to plot[1-%d][0: exit]? ',size(dat,2));
       signal = input(str)
   end
   if signal == 0
      break;
   end

   [status_mod,status_unmod,upline] = sprt(res,data_mean,sigma);

        %%%%%%%%%%%%
        % plotting %
        %%%%%%%%%%%%
   clf
        subplot(311)
   plot(dat(:,signal),'color',[0.5 0.5 0.5]);hold on;plot(y(:,signal),'k--');hold off
   legend('mea','est');
        ylabel('measurement/estimate')
        str = sprintf('Average Error Level = %f %%',err_pct(signal));
   title(str)
   xlim=get(gca,'XLim');
   if strcmp(drift,'y') & pct == 100
      axis([0 inf -100 inf]);
   end
        set(gca,'XLim',xlim);
        subplot(312)
        plot(dat(:,signal)-y(:,signal))
        ylabel('residual')
   subplot(313)
   plot(status_mod(:,signal),'color',[.5 0 0]);hold on;...
   plot(status_unmod(:,signal),'g--');hold off
   legend('modified SPRT','original SPRT');
   ylabel('sensor status')
   axis([0 inf min(status_mod(:,signal))*1.1 max(status_mod(:,signal))*1.1]);
   set(gca,'XLim',xlim);
   xlabel('data sampled in every minute')
end
```

F11 Faulty Sensor Replacement Algorithm

```
% replacement.m
%
% Replacing faulty signals when fault occurs
%
% Xiao Xu
% Jan 2,2000

clf

undrift = dat;
disp('1. ramp drift;')
disp('2. step drop;')
disp('3. momentarily step drop.')
change_type = input('your choice: ');
sig = 0;
while sig >= 0
   str = sprintf('drifting which signal?[1-%d,-1 for no more drift sigs]: ',size(dat,2));
   sig = input(str);
   if sig < 0
```

```
      break;
   else
      signal = sig;
   end
   pct = input('enter the percentage of drift: ');
   start = input('start at ');
   if change_type == 3
      finish = input('finish at ');
   end
   if change_type == 1
      for i = start:size(dat,1)
         dat(i,signal) = dat(i,signal1)-0.01*pct/(24*60)*(i-start)*dat(i,signal);
      end
   elseif change_type == 2
      dat(start:size(dat,1),signal) = dat(start:size(dat,1),signal)*(1-pct*0.01);
   else
      dat(start:finish,signal) = dat(start:finish,signal)*(1-pct*0.01);
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Determine residual variance and faulted means
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[data_mean,sigma] = sprt_par(net,undrift,xm,xs);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% faulty signal replacement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('start replacement')
pause

alpha = 0.01e-2;
beta = 10e-2;
lowline = log(beta/(1-alpha));
upline = log((1-beta)/alpha);
det_level = input('detection level (%) = ');
m = det_level/100*data_mean;

init_detect_time = inf;
status = zeros(1,size(dat,2));
for i = 1:length(dat)
   if rem(i,100) == 0
      fprintf('time elapsed = %d min.\n',i);
   end

   xn = trastd(dat(i,:)',xm,xs);
   X = sim(net,xn);
   y = poststd(X,xm,xs)';      % unscaled output
   res = dat(i,:)-y;

   status = status+(((m./sigma).^2)/27).*(abs(res)./m-0.5*ones(1,size(res,2)));
   if(status(signal) <= lowline)
      status(signal) = 0;
   end
   if(status(signal) > upline)
      status(signal) = upline;
      dat(i,signal) = y(:,signal);
      if i <= init_detect_time & i >= start
         init_detect_time = i;
      end
      xn = trastd(dat(i,:)',xm,xs);
      X = sim(net,xn);
      y = poststd(X,xm,xs)';  % unscaled output
      dat(i,signal) = y(:,signal);
   end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% rerun network prediction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xn = trastd(dat',xm,xs);
X = sim(net,xn);
y = poststd(X,xm,xs)'; % unscaled output

draw_plot(undrift,y,data_mean,sigma);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% signal recovered percentage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

recover_part = mean(abs(y(init_detect_time:length(dat),signal)./...
   undrift(init_detect_time:length(undrift),signal)))*100;
```

F12 Truncated Singular Value Decomposition Algorithm

```
function [w,b]= tsvd(x,y)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FUNCTION: [w,b] = tsvd(x,y)
%
%       Performs a least means square estimate using truncated singular
%  value decomposition (SVD). Function takes two arguments:
%
%  x - Q*P independent var. matrix (Q: # of input variables)
%  y - R*P dependent var. matrix (R: # of output variables)
%
%  and estimates:
%
%  w - weight matrix
%       b - bias vector
%
%       The input matrix is padded with a bias matrix.
%       Noise is eliminated in the input set by the parameter e.

%  Xiao Xu  Feb. 12, 1999

if size(x,1) > size(x,2)
   error('input/output should be row vectors.')
end
if size(x,2) ~= size(y,2)
   error('input/output should have same patterns.')
end
if size(x,2) > 1000
   int = round(size(x,2)/1000);
   x = x(:,1:int:size(x,2));
   y = y(:,1:int:size(y,2));
end
fprintf('start SVD \n');
if nargout <= 1
   w = y/x;
else
   x = x';
       y = y';
       [r,c] = size(x);
       x = [x ones(r,1)];                    % padding unity bias vector
       [u,s,v] = svd(x);                     % svd of input vectors

       %%% find the truncated point in matrix s
    sum_s = 0;

   for i = 1:size(s,2)
```

```
        sum_s = sum_s+s(i,i);
        end
        for j = 1:size(s,2)
        nor_s(j)=s(j,j)./sum_s;                  % normalized singular values
        end
        info = [];
        for k = 1:length(nor_s)
        info = [info;sum(nor_s(1:k))];
        end
        cut_off = .98;
        m = max(find(info <= cut_off));
        if m < length(nor_s)
                if abs(info(m)-cut_off) > abs(info(m+1)-cut_off)
                m = m+1;
                end
        end
        s = s(1:m,1:m);
        u = u(:,1:m);
        v = v(:,1:m);
        beta = v*inv(s)*u'*y;                    % estimate solution
        w = beta(1:c,:)';                        % extract weight matrix
        b = beta(c+1,:)';                        % extract bias vector
end
```

F13 Sequential Probability Ratio Test (SPRT) Parameter Setting

```
function [data_mean,sigma] = sprt_par(net,dat,xm,xs);

% sprt_par.m
%
% Calculate the parameters in SPRT algorithm
% mean - unfaulted signal mean
% vr - standard deviation of residuals
%
% Xiao Xu
% Apr 10,1999

xn = trastd(dat',xm,xs);
X = sim(net,xn);
y = poststd(X,xm,xs)'; % unscaled output

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Determine residual variance and faulted means
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

res = dat-y;
sigma = std(res);
data_mean = abs(mean(dat));
```

F14 SPRT Procedure Processing

```
function [status_mod, status_unmod, upline] = sprt(res,data_mean,sigma);

%  Process SPRT procedure

status_mod = zeros(size(res));
status_unmod = zeros(size(res));

det_level = input('detection level (%) = ');
m = det_level/100*data_mean;


alpha = 0.01e-2;
```

```
beta = 10e-2;
lowline = log(beta/(1-alpha));
upline = log((1-beta)/alpha);

%clear old_dat;

for t = 2:size(res,1)

   status_mod(t,:) = status_mod(t-1,:)+(((m./sigma).^2)/27).*(abs(res(t,:))./m-
0.5*ones(1,size(res,2)));
   status_unmod(t,:) = status_unmod(t-1,:)+((m./sigma).^2).*(abs(res(t,:))./m-
0.5*ones(1,size(res,2)));

   for sig = 1:size(res,2)
      if(status_mod(t,sig) <= lowline)
         status_mod(t,sig) = 0;
       end
      if(status_mod(t,sig) > upline)
         status_mod(t,sig) = upline;
       end


      if(status_unmod(t,sig) <= lowline)
         status_unmod(t,sig) = 0;
       end
      if(status_unmod(t,sig) > upline)
         status_unmod(t,sig) = upline;
       end
       end
end
```

F15 Autocorrelation Matrix Calculation

```
function CC = autocorr(dat);

%  This code is to calculate autocorrelation function
%  x -- data vector (# of variables by # of patterns)
%
%  Syntax:
%
%  CC = autocorr(dat);
%  It takes the signals matrix dat and returns a normalized
%  auto-correlation matrix CC.

%   Mike Xu, Nov 2, 1998

clear all

load work_data_2;
load Final_Group;
stdx = std(dat);
ind = find(stdx <= 0.02);
if ~isempty(ind)
   dat(:,ind) = [];
end

dat = medfilt1(dat,3);
x = dat';
x = x(:,1:10:size(x,2));
clear stdx ind final_group dat;

[m,N] = size(x);                        % N - # of sample points

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% mean centered data
```

```
[illegible]

x = x-mean(x')'*ones(1,size(x,2));

[illegible]
%  estimate autocorrelation matrix R_xx
[illegible]
Max_Time_Lag = input('enter max time lag (return for max. time lag): ');
if isempty(Max_Time_Lag)
   Max_Time_Lag = N-1;
end
R_xx = [];
k = 0;
while k <= Max_Time_Lag
   cov = zeros(m,m);
       for i = 1:N-k
       cov = cov+x(:,i+k)*x(:,i)';
   end
   temp = 1/N*cov;
   if k == 0           % zero time lag cov.
      c0 = temp;
   end

   for ii = 1:m                % Normalize R_xx
      for jj = 1:m
         temp(ii,jj) = temp(ii,jj)/sqrt(c0(ii,ii)*c0(jj,jj));
      end
   end
   R_xx = [R_xx;temp];
   k = k+1;
end
```

F16 Sensitivity Analysis Algorithm

```
function sen_test(net,dat,xm,xs)

%       sen_test.M:
%
%       Sensitivity analysis on a five layer NN. Each input vector
%       was perturbed  by 5%. Plots change in output due to each input.
%
%       Miao Xu, Sep 7, 1992
%

clf

x = dat;

[patterns] = size(x,1);
sprintf('construct input data set.\n')
[patterns,numin] = size(x);
numout = numin;
y = x(:,1:numout);

min_max_value = minmax(y');
range = min_max_value(:,2)-min_max_value(:,1);

[illegible]
%  Perform sensitivity analysis
[illegible]

m = zeros(numout,numin);

xn = trastd(x',xm,xs);
out = sim(net,xn);
```

```
y = poststd(out,xm,xs)';      % unscale output

pert = 0.05*range';                  % Perturb input by 5%
for j = 1:numout                                  % Perturb one input at a time
   X = x;
   for i = 1:patterns
        X(i,j) = X(i,j)-pert(:,j);
      end
   end
        Xn = trastd(X',xm,xs);
   Out = sim(net,Xn);

        Y = poststd(Out,xm,xs)';      % convert scaled output back to original
   m(:,j) = (sum(abs((y-Y)./y))./length(y))'*100;    % Error between input and output
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Normalize Matrix for Plotting
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

s = sum(m);
sm = ones(numout,1)*s;
if sm ~= 0
   nm=m./sm;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Plot Sensitivities
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

subplot(2,1,1)
bar(m','stacked')
title('Sensitivity Analysis of Neural Network')
ylabel('Sum of % Output Change')

subplot(2,1,2)
bar(nm','stacked')
xlabel('Input Changed by 5%')
ylabel('Normalized Output Change')
```

VITA

Xiao Xu was born in Chengdu, China on November 11, 1966. He attended Xi'an Jiaotong University where he received the Bachelor of Science in Nuclear Engineering in 1987. After working for several years at Nuclear Power Institute of China, he entered the Master's program in Nuclear Engineering at the University of Arizona in 1992 and received his Master of Science in 1994. In 1995, he began his Ph.D. program in Nuclear Engineering at the University of Tennessee. He spent five years of research in artificial intelligence, especially in neural network modeling for complex systems. He earned his Doctor of Philosophy in Nuclear Engineering in 2000.