View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Texas A&M Repository

USE OF AUTOASSOCIATIVE NEURAL NETWORKS FOR SENSOR DIAGNOSTICS

A Thesis

by

MASSIEH NAJAFI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2003

Major Subject: Mechanical Engineering

USE OF AUTOASSOCIATIVE NEURAL NETWORKS FOR SENSOR DIAGNOSTICS

A Thesis

by

MASSIEH NAJAFI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Reza Langari (Co-Chair of Committee)	Charles H. Culp III (Co-Chair of Committee)
Jeff S. Haberl (Member)	Dennis L. O'Neal (Head of Department)

December 2003
Major Subject: Mechanical Engineering

ABSTRACT

Use of Autoassociative Neural Networks for Sensor Diagnostics. (December 2003)

Massieh Najafi, B.Sc., University of Tehran

Committee Co-Chairs: Dr. Reza Langari
Dr. Charles H. Culp III

The new approach for sensor diagnostics is presented. The approach, Enhanced Autoassociative Neural Networks (E-AANN), adds enhancement to Autoassociative Neural Networks (AANN) developed by Kramer in 1992. This enhancement allows AANN to identify faulty sensors. E-AANN uses a secondary optimization process to identify and reconstruct sensor faults. Two common types of sensor faults are investigated, drift error and shift or offset error. In the case of drift error, the sensor error occurs gradually while in the case of shift error, the sensor error occurs abruptly. E-AANN catches these error types. A chiller model provided synthetic data to test the diagnostic approach under various noise level conditions. The results show that sensor faults can be detected and corrected in noisy situations with the E-AANN method described. In high noisy situations (10% to 20% noise level), E-AANN performance degrades. E-AANN performance in simple dynamic systems was also investigated. The results show that in simple dynamic situations, E-AANN identifies faulty sensors.

DEDICATION

I dedicate this work to my parents. No words will ever express how grateful I am to them.

ACKNOWLEDGMENTS

This work could not have been completed without the guidance and supervision of my two professors, Dr. Reza Langari and Dr. Charles Culp, to whom I owe most of this accomplishment. I also acknowledge the very helpful assistance and review from Dr. Jeff Haberl. The continual support and inspiration I received from them made all efforts worthwhile.

I gratefully acknowledge the support received from the California Energy Commission in supporting this work. This work was supported under contract number 66503346 with the Ernest Orlando Lawrence Berkeley National Laboratory. This work has been performed as part of the High Efficiency Commercial Building Project under the California Energy Commission Public Interest Energy Research Program.

TABLE OF CONTENTS

Page

ABSTRACT iii

DEDICATION iv

ACKNOWLEDGMENTS v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES xiv

1. INTRODUCTION 1

1.1 Autoassociative Neural Network and Its Application in Sensor Diagnostics 2
1.2 Proposed Solution, Enhanced Autoassociative Neural Network (E-AANN) 4

2. LITERATURE SURVEY 8

2.1 System Fault Diagnostics 8
2.1.1 Reference Computation Section 9
2.1.2 Analysis Section 12
2.2 Sensor Diagnostics 12
2.2.1 Reference Computation Section 13
2.2.2 Analysis Section 14
2.3 Chiller Fault Diagnostics 15

3. INTRODUCTION TO NEURAL NETWORKS 18

3.1 Introduction 18
3.2 Architecture of the Neural Network 19
3.3 Neural Network Training 20
3.3.1 Gradient Descent 20
3.3.2 Backpropagation Technique 22

4. AUTOASSOCIATIVE NEURAL NETWORK 28

4.1 Architecture of the Autoassociative Neural Network 28
4.2 Autoassociative Neural Networks and Identity Mapping 29

Page

4.3 Autoassociative Neural Network Training 30
4.3.1 AANN Training in Noisy Situations 31

5. CHILLER MODEL AND DATA GENERATION 36

5.1 Model Inputs/Outputs 36
5.2 Data Generation 38
5.3 Normalization 40
5.4 Inducing Noise 41

6. ENHANCED AUTOASSOCIATIVE NEURAL NETWORK (E-AANN) 44

6.1 Hines Approach 45
6.2 Enhanced Autoassociative Neural Network 46
6.3 Examples 48

7. E-AANN SPECIFICATIONS 53

7.1 E-AANN Accuracy 53
7.2 E-AANN Performance in Noisy Situations 54
7.3 Maximum Level of Noise Tolerated by E-AANN 71

8. APPLICATION OF E-AANN IN SENSOR DIAGNOSTICS IN DYNAMIC SYSTEMS 80

8.1 Introduction 80
8.2 E-AANN Response to Dynamic Systems in Non-faulty Situations 81
8.3 E-AANN Application in Sensor Diagnostics in Dynamic Situations 94

9. CONCLUSION 97

REFERENCES 98

VITA 102

LIST OF FIGURES

FIGURE Page

Figure 1-1 Model based approach 1

Figure 1-2 Autoassociative Neural Network 3

Figure 1-3 Hines' approach to using AANN for sensor diagnostics 4

Figure 1-4 Drift and shift errors 6

Figure 1-5 Drift error with noise 6

Figure 1-6 Shift error with noise 7

Figure 2-1 A generic fault detection mechanism 9

Figure 2-2 The structure of analyzer 9

Figure 2-3 AANN architecture 14

Figure 3-1 Node with several inputs and one output 18

Figure 3-2 Architecture of a three layer feed forward neural network 20

Figure 3-3 Local minima and global minima 22

Figure 3-4 A general architecture of neural network 24

Figure 3-5 Neural network architecture 27

Figure 4-1 AANN general architecture 28

Figure 4-2 Identity mapping 30

Figure 4-3 Data set with one input and one output 32

Figure 4-4 A good solution found by neural network 32

Figure 4-5 A bad solution found by neural network 32

Figure 4-6 Identity mapping with 8 inputs/outputs 34

FIGURE Page

Figure 4-7 Training set 35
Figure 4-8 The relation between sum squared error and 300 epochs 35
Figure 5-1 System boundary of the chiller model, from [3] 36
Figure 5-2 Chiller model inputs/outputs 37
Figure 5-3 Whole data with 1000 samples 38
Figure 5-4 Training set (sorted) 39
Figure 5-5 Test set (sorted) 39
Figure 5-6 Normalized training set (sorted) 40
Figure 5-7 Normalized test set (sorted) 41
Figure 5-8 Training set with 2% noise 42
Figure 5-9 Test set with 2% noise 42
Figure 5-10 Training set with 5% noise 43
Figure 5-11 Test set with 5% noise 43
Figure 6-1 AANN structure 44
Figure 6-2 Approach recommended by Hines 46
Figure 6-3 Test data with 1% noise 49
Figure 6-4 Test data with 1% noise, sensor #3 has drift error 49
Figure 6-5 E-AANN output 50
Figure 6-6 Sensor #3 output before contamination, after inducing drift error, and the data reconstructed by E-AANN 50
Figure 6-7 Cost function 52
Figure 6-8 Contour plot of cost function 52
Figure 7-1 Test set with 2% noise 55

FIGURE Page

Figure 7-2 Test set with 2% noise, sensor 3 has drift error 56

Figure 7-3 Drift error and shift error 56

Figure 7-4 Sensor 3 output before and after inducing drift error, noise level is 2%... 57

Figure 7-5 E-AANN output; the input data had 2% noise and drift-error 58

Figure 7-6 Sensor #3 output before contamination, after inducing drift error, and the data reconstructed by E-AANN 58

Figure 7-7 The difference between E-AANN input and output. The input data had 2% noise and drift error 59

Figure 7-8 Test set with 2% noise, sensor 3 has drift error 60

Figure 7-9 Sensor 3 output before contamination and after inducing shift error, noise level 2% 60

Figure 7-10 The induced 2% noise and shift error to sensor #3 ([sensor #3 output + noise + drift error]-[sensor #3 output]) 61

Figure 7-11 E-AANN output; the input data had 2% noise and shift-error 62

Figure 7-12 Sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN 62

Figure 7-13 The difference between E-AANN input and output. The input data had 2% noise and shift error 63

Figure 7-14 Test data with 5% noise 64

Figure 7-15 Test set with 5% noise, sensor 3 has drift error 64

Figure 7-16 Sensor 3 output before contamination and after inducing drift error, noise level is 5% 65

Figure 7-17 The induced 5% noise and drift error to sensor #3 65

Figure 7-18 E-AANN output; the input data had 5% noise and drift-error 66

Figure 7-19 Sensor #3 before contamination, after inducing drift error, and the data reconstructed by E-AANN, noise level is 5% 66

FIGURE Page

Figure 7-20 The difference between E-AANN input and output. The input data had 5% noise and drift error 67

Figure 7-21 Test set with 5% noise, sensor 3 has shift error 68

Figure 7-22 Sensor 3 outputs before contamination and after inducing shift error, noise level is 5% 68

Figure 7-23 The induced 5% noise and shift error to sensor #3 69

Figure 7-24 E-AANN output; the input data had 5% noise and shift-error 69

Figure 7-25 Sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN, noise level is 5% 70

Figure 7-26 The difference between the E-AANN input and output. The input data had 5% noise and shift error 70

Figure 7-27 Test data with 10% noise 72

Figure 7-28 Test set with 10% noise, sensor 3 has drift error 73

Figure 7-29 Sensor 3 output before contamination and after inducing drift error, noise level 10% 73

Figure 7-30 The induced 10% noise and drift error to sensor #3 74

Figure 7-31 E-AANN output; the input data had 10% noise and drift-error 74

Figure 7-32 Sensor #3 outputs before contamination, after inducing drift error, and the data reconstructed by E-AANN, noise level is 10% 75

Figure 7-33 The difference between E-AANN input and output. The input data had 10% noise and drift error 75

Figure 7-34 Test set with 10% noise, sensor 3 has shift error 76

Figure 7-35 Sensor 3 output before contamination and after inducing shift error, noise level is 10% 77

Figure 7-36 The induced 10% noise and shift error to sensor #3 77

Figure 7-37 E-AANN output; the input data had 10% noise and shift-error 78

FIGURE Page

Figure 7-38 Sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN, noise level is 10% 78

Figure 7-39 The difference between E-AANN input and output. The input data had 10% noise and shift error 79

Figure 8-1 Chiller model inputs/outputs 81

Figure 8-2 Static model with delay on one sensor to mimic dynamic system, Z^{-4} means a delay of 4 samples 82

Figure 8-3 E-AANN inputs and outputs 83

Figure 8-4 Training set 83

Figure 8-5 Test set, step size is 10 84

Figure 8-6 E-AANN output 85

Figure 8-7 The difference between the E-AANN input and output (delay = 4 steps) 85

Figure 8-8 The difference between the E-AANN input and output zoomed in (delay = 4 steps) 86

Figure 8-9 Static model with delay on one sensor to mimic dynamic system, Z^{-12} means a delay of 12 samples 86

Figure 8-10 E-AANN output 87

Figure 8-11 The difference between the E-AANN input and output (delay =12 steps) 87

Figure 8-12 The difference between the E-AANN input and output zoomed in (delay = 12 steps) 88

Figure 8-13 Fourth graph of Figure 8-8 zoomed in 89

Figure 8-14 Static model with delay on one sensor to mimic dynamic system, Z^{-20} means a delay of 20 samples 90

Figure 8-15 The difference between the E-AANN input and output (delay = 20 steps) 91

FIGURE Page

Figure 8-16 Static model with delay on multiple sensors to mimic dynamic system... 92

Figure 8-17 Test data 92

Figure 8-18 E-AANN output 93

Figure 8-19 The difference between the E-AANN input and output 93

Figure 8-20 Test Data, the sixth sensor has drift error 95

Figure 8-21 E-AANN output 95

Figure 8-22 The difference between the E-AANN input and output 96

LIST OF TABLES

Table Page

Table 6-1 Original data, contaminated data and reconstructed data 51

1. INTRODUCTION

When sensors malfunction, control systems become unreliable. Even with the most sophisticated instruments and control algorithms, a control decision based on faulty data could lead to disaster. Sensor Fault Detection is usually considered as a subset of Fault Detection. One of the well known approaches in Fault Detection is the model based approach in which a computational model is designed to predict the real system output while receiving the same input. Figure 1-1 shows the generic diagram of the model-based technique.

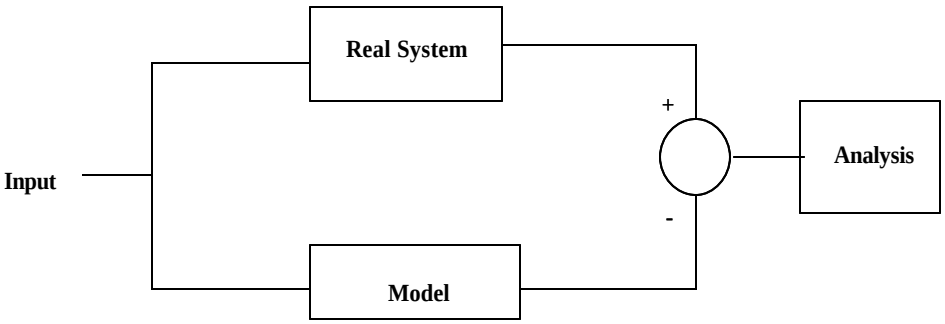

Figure 1-1, Model based approach

In spite of the popularity of the model-based approach in Fault Detection, this method is not appropriate for Sensor Diagnostics. This is because the model-based approach relies on the correct input data (which might be measured using sensors). This assumes that the input to the real system and the input to the model are correct (fault free). When there is a notable difference between the output of the real system and the output of the model, a problem exists in the real system. In Sensor Diagnostics the focus is to find dysfunctional sensors.

Autoassociative Neural Networks (AANN) are an alternative solution for Sensor Diagnostics. The AANN concept, which was developed by Kramer ([1] and [2]) can

---

This thesis follows the style and format of IEEE Transactions on Power Systems.

capture the interrelationship between plant variables that have some degree of relation with each other. The correlation between plant variables is embedded in the AANN.

Autoassociative Neural Networks for Sensor Diagnostics is the focus of this research. We are looking for a generic approach to be applicable in different systems. A chiller model developed previously [3] will be used as a system to test our approach.

1.1 Autoassociative Neural Network and Its Application in Sensor Diagnostics

"An Autoassociative Neural Network (AANN) is a network in which the outputs are trained to emulate the inputs over an appropriate dynamic range. Plant variables that have some degree of coherence with each other constitute the input. During training, the interrelationships between the variables are embedded in the Neural Network connection weights"(Hines 1998). Autoassociative networks are composed of an input layer, a number of hidden layers and an output layer. Theoretically, it is sufficient for the AANN to contain three hidden layers [1]. However; in practice more hidden layers might be used for improved performance [4]. The architecture of a three hidden layer AANN is shown in Figure 1-2. The output layer of AANN produces a transformed version of the inputs and is equal in dimension to the input.

In Sensor Diagnostics, all data measured from the real system (a mix of input variables and output variables) constitute the input to the AANN. The AANN is trained in a way that its outputs match the inputs as closely as possible, in a least squares sense, over the training set. When data having no errors (no faulty sensors) is fed to the trained AANN, the difference between the input and output of the AANN is ideally zero. If the data is contaminated (one or more sensors being faulty), the difference between the input and output of the AANN will be non-zero.

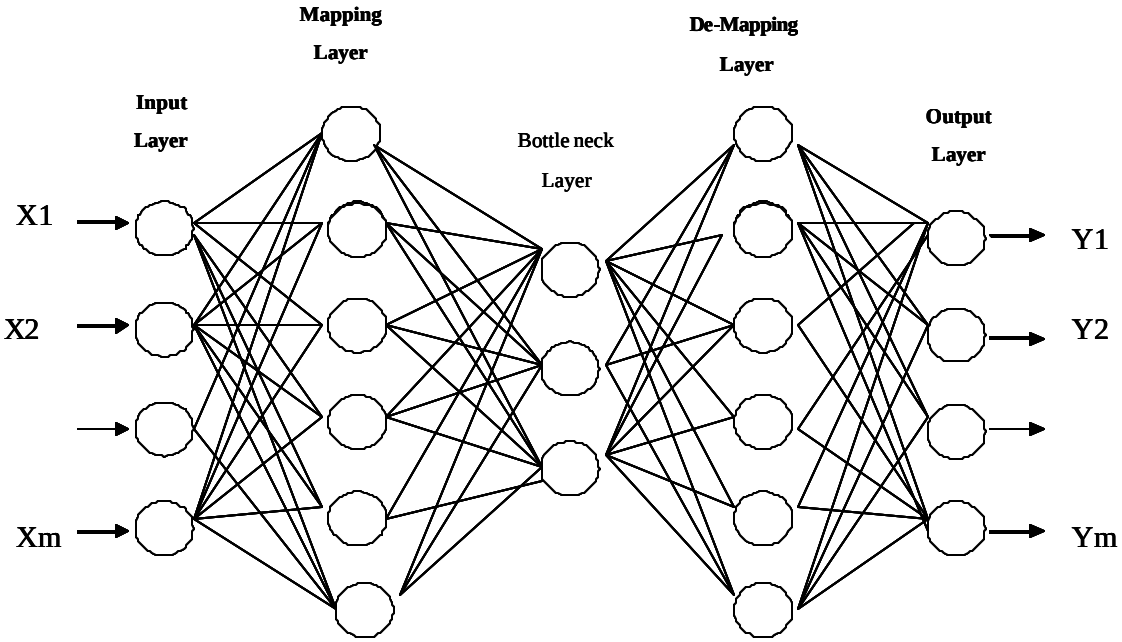

Figure 1-2, Autoassociative Neural Network

The AANN approach can be used to determine whether there is a sensor problem but the problem of locating the faulty sensors still remains. At the University of Tennessee, Hines and his colleagues have proposed a method to locate faulty sensors using AANN ([5] and [6]). As noted by Hines [5] "As during the training of the AANN, the interrelationships between the variables are embedded in the Neural Network connection weights. 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 means that the difference between each AANN input and its corresponding output contains enough information to help determine the faulty sensors. If the difference is zero, the corresponding sensor is healthy otherwise the sensor has a problem (Fig. 1-3). However we were unable to get the same results as they did, perhaps due to an error or

unreported assumptions in the original work. The proposed work seeks to develop an enhanced method which is referred to as E-AANN.

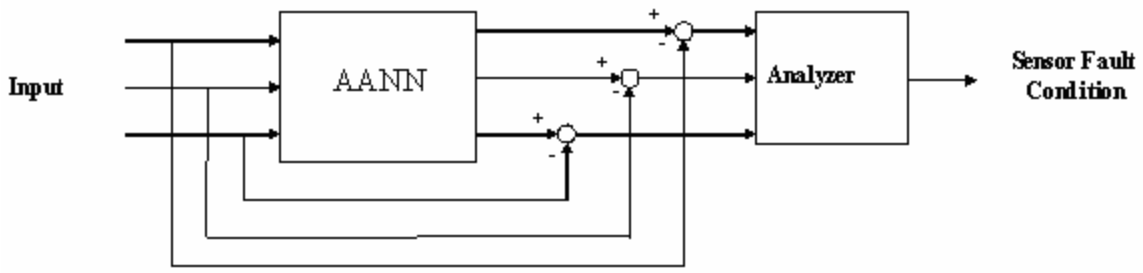

Figure 1-3, Hines' approach to using AANN for sensor diagnostics

1.2 Proposed Solution, Enhanced Autoassociative Neural Network (E-AANN)

Due to the inherent non-orthogonality of the AANN, when one of the AANN inputs is degraded or contaminated, it affects all AANN outputs. So any difference between the input and output of an AANN reflects sensor problems but is not sufficient to localize the faulty sensors. The new approach, E-AANN, is based on the fact that whenever the input to the AANN is fault free, the AANN output will be the same as input. In this situation, the Sum Squared Error (SSE) between the inputs ($X_i, i = 1,2,... n$) and outputs ($Y_i, i = 1,2,... n$) should ideally be zero (Eq. 1).

$$J = (X - Y)^T (X - Y) = \sum_{i=1}^{N} (Y_i - X_i)^2 = 0 \qquad (1)$$

(*Eq.*1)

In order to locate the faulty sensor, each input is varied over its defined range while holding all other inputs fixed to determine when SSE is zero or close to zero (or in principle minimized). Once SSE equals to zero, the corresponding input is identified as the contaminated input (faulty sensor). E-AANN is explained in section 6 in details.

The next step in this research is to characterize the E-AANN performance. First it is investigated how precisely the E-AANN can reconstruct the correct value of a faulty sensor output. The aim is to find parameters that affect on the precision of E-AANN. Secondly the effect of noise on the E-AANN performance is investigated. Real sensors are noisy. It is necessary to see if E-AANN is capable of determining faulty sensors in noisy conditions.

Two common types of sensor faults are investigated in this study. One is drift error and the other is shift or offset error. In the case of drift error, the sensor error occurs gradually. Initially, the error is very small but it grows slowly with time (Fig. 1-4). In the case of the shift error, the sensor error occurs abruptly. The E-AANN capability to catch shift errors and drift errors in noisy conditions is studied. Initially the induced noise is low and then it will be increased step by step. At each noise level, the data will be contaminated by shift and drift errors (Figures 1-5 and 1-6 show drift and shift errors with noise). E-AANN will be evaluated in terms of its ability to catch these errors. If necessary, E-AANN will be revised to improve its performance. The maximum level of noise handled by E-AANN will be determined. The functionality of the E-AANN in high noisy situations will be evaluated.

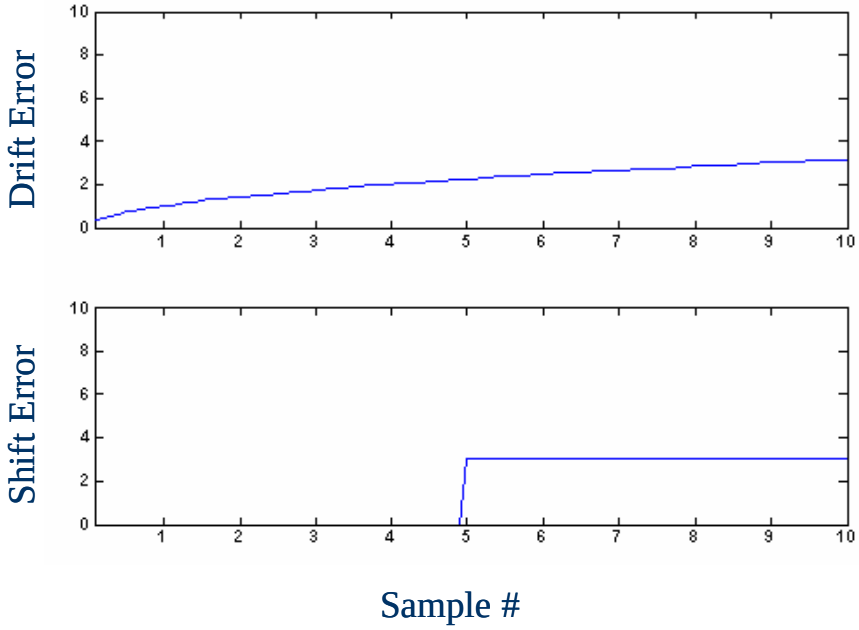

Figure 1-4, Drift and shift errors

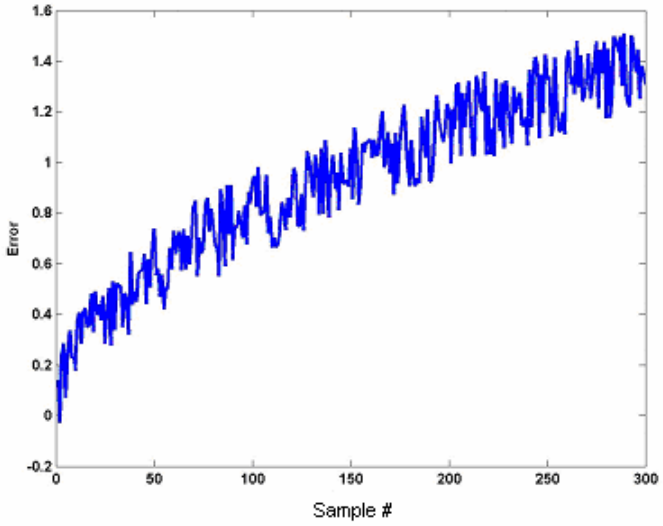

Figure 1-5, Drift error with noise

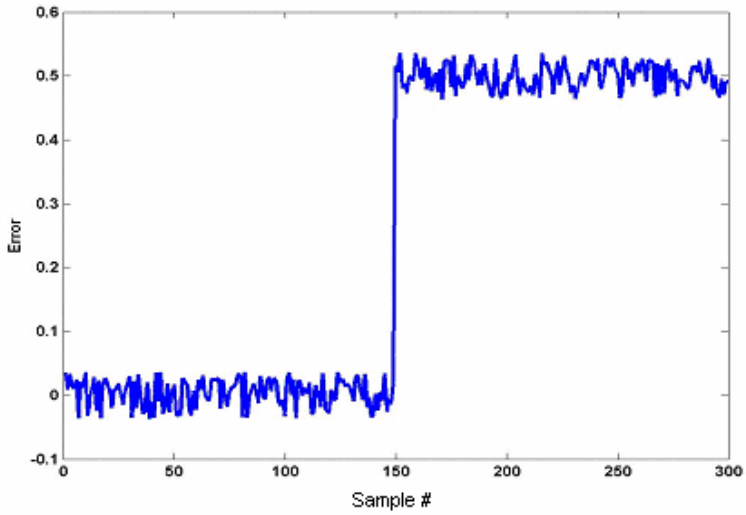

Figure 1-6, Shift error with noise

2. LITERATURE SURVEY

The topic of Fault[1] Diagnostics can be divided into two main categories:

System Fault Diagnostics

Sensor Fault Diagnostics

In a system, Fault Diagnostics deals with faults originating inside the system. It checks system components to see whether they work well. Sensor Diagnostics deals with faults coming from the sensors that measure system variables. It checks whether sensors show the correct values.

2.1 System Fault Diagnostics

Generally, the process of fault detection is divided into two steps: 1- Fault Determination and 2- Fault Isolation. In fault determination, it is checked whether system performance is within specifications or not. In fault isolation, the source of fault is located.

There are several generic approaches to fault determination but the topic of fault isolation is still in its early stages. This is due to the fact that the process of Fault Isolation depends on the system specifications. Therefore it is difficult to derive a generic approach to fault Isolation.

A generic fault detection mechanism must contain two sections: 1- Reference Calculation Section and 2-Analysis Section (Figure 2-1). The task of the Reference

[1] A comprehensive literature survey has been performed on the issue of fault diagnostics and sensor diagnostics. Different approaches used in different systems have been investigated and their advantaged and disadvantages have been discussed. We have also performed a literature survey on the issue of fault diagnostics in chiller. This is true that in this project we look at diagnostic problems generically, but we thought it would be useful to also have a literature survey on a specific system (like chiller) to see how the results could be expanded.

Calculation is to prepare the tools for evaluation of the system performance. It calculates the reference values to be compared with system outputs. The analysis Section handles two tasks. It compares the system outputs with the calculated references to determine if there is any problem in the system (Fault Determination). After determination of faults, it tries to isolate the fault (Fault Isolation). Therefore the analysis section is divided into two sections (Figure 2-2):

I) Fault Determination Section

II) Fault Isolation Section

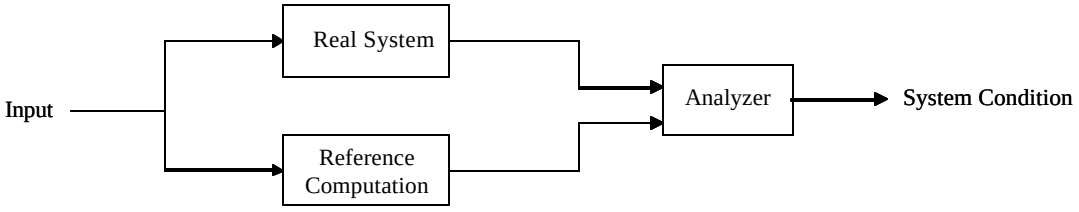

Figure 2-1, A generic fault detection mechanism

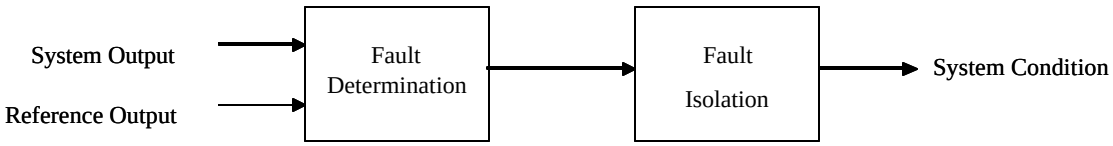

Figure 2-2, The structure of analyzer

2.1.1 Reference Computation Section

As it was previously stated, the Reference Calculation prepares the tools for evaluation of the system performance. The most popular approach for reference calculation is the Model Based approach. In this approach, a model is designed based on the real system specifications. It predicts the output of the real system with a defined input. The common methods to design a model are:

Physical or Parametric methods

Data Driven methods

Qualitative methods

Physical/Parametric Models

These models are designed based on the equations and parameters of the real system. In this approach the key issue is to resolve the system equations and find the parameters. As an example, we can mention to the work done by Lee and his colleagues [7] in which they have present a scheme for detecting faults in air-handling units.

In less complicated systems, the physical model can be readily found. In some research efforts an estimation algorithm has been developed to update the model parameters based on the measured signals. In [8] Keyhani has explained how the measurement signals from an electric motor can be used for parameter identification to estimate the relevant information regarding the motor working condition. Kornand and his colleagues have persuaded the same approach for fault detection in milling [9]. They have used a precise force model and an estimation algorithm to update model parameters based on the measured force.

However, when the system is too complicated or there is not enough information of the physical specifications of the system, the approach of physical/parametric models does not work. Also this approach can not lead to a generic fault detection mechanism as it mainly depends on system specifications which change from case to case.

Data Driven Models

This is a generic approach for system modeling. In this approach the model is designed based on the system performance. The data constructed based on the system records is used to tune the model parameters. Neural Networks and Fuzzy logic systems are two well known tools for this approach. Chow and his colleagues have carried out comprehensive investigation on various neural network based (model based) fault detection scheme [10-12]. In [13], Chow proposed a typical back-propagation neural network structure for incipient motor faults diagnosis. In [14], Gao and Ovaska presented several typical fault diagnosis schemes based on neural networks, fuzzy logic, neural-fuzzy, and genetic algorithms. They compared the advantages and disadvantages of these methods. Nejjari and Benbouzid [15] applied fuzzy logic to the diagnosis of induction motor stator and phase conditions.

Data Driven Models can be deigned quickly. However, these models are only as good as data used to generate them. Their accuracy falls off rapidly in the region of operating space for which there is no training data.

Qualitative Models

Qualitative Models are designed based on the qualitative knowledge of the system. They predict the system behavior qualitatively. If some quantitative knowledge is included, the model is called semi-qualitative. Semi-qualitative models can determine a range for each system variable. For example Yumoto and his colleagues [16 and 17] have worked on fault diagnostics in HVAC systems using qualitative reasoning.

The main advantage of qualitative models is their ability to predict the system performance even when system parameters change slightly. In situations that it is impossible to find the system variables exactly, qualitative approaches are a better solution. Qualitative models can not be used to develop generic approaches in diagnostic

area. This is due to the dependency of qualitative models on system specifications which change from case to case.

2.1.2 Analysis Section

The role of the Analyzer is to compare the system outputs with references to determine and isolate faults (Fault Determination and Fault Isolation). In model based approaches, the Fault Determination section compares the difference between the system output and model output (residual) with a constant or changeable threshold. The value of threshold depends on the system specifications, system noise level, model accuracy, and sensor resolutions. For example Ton and Huo [18] have shown how the threshold should be evaluated in neurofuzzy based fault detection mechanism for induction motors.

When the residual passes the threshold, the Fault Determination section finds that there is an error. After that the Fault Isolation section comes in to isolate the fault. It looks for some predefined signatures in the faulty response. The signatures are defined based on the system specifications. They might be used to design a fuzzy system, train a neural network. Benhouzid and Nejjari [19] have shown how signatures can be expressed in a simple fuzzy logic approach to monitor the stator of induction motors.

2.2 Sensor Diagnostics

Like Fault Diagnostics, generally the process of sensor diagnosis is divided into two steps: 1- Sensor Fault Determination and 2- Sensor Fault Isolation. The first step determines if there is any sensor problem in the system while the second step locates the faulty sensors.

A generic Sensor Fault Detection mechanism is divided into two parts: 1- Reference Calculation Section and 2- Analysis Section. Although this classification is similar to that of Fault Diagnostics, the internal structures are different.

2.2.1 Reference Computation Section

As it was mentioned earlier, the task of the Reference Computation is to prepare some references for the evaluation of the system performance. On the issue of Fault Diagnostic, the well known approach for reference calculation was model based approach. Feeding the same input to the real system and the model, the difference between their outputs is the best source for fault detection purposes.

In spite of the popularity of the model-based approach in Fault Detection, this method is not appropriate for Sensor Diagnostics. The model-based approach relies on the correct input data (which might be measured using sensors). This assumes that the input to the real system and the input to the model are correct (fault free). When there is a notable difference between the output of the real system and the output of the model, a problem likely exists in the real system. In Sensor Diagnostics, the focus is to find dysfunctional sensors.

In a Sensor Diagnostics, as both input and output data are measured by sensors, none of them is trustable (each one might be faulty). We can not assume that the input data is correct and then relate any problems in the output data to the output sensors. Therefore the conventional model based approaches (the model that predicts the output of the system having the same data at the input) are not suitable for the Sensor Diagnostics.

Autoassociative Neural Network (AANN)

Autoassociative Neural Networks (AANN) are an alternative solution for Sensor Diagnostics. Autoassociative Neural Networks do not discriminate between the input and output. The concept of Autoassociative Neural Network was first developed by Kramer [1] in 1992. Figure 2-3 shows the general architecture of the AANNs. Basically AANNs are identity mappings. In sensor diagnostics, the set of inputs and outputs of the real system constraint the input to the network [1, 5, and 6]. The AANN is trained to recreate the input as its output. When the data input to the AANN is non-faulty, the

AANN output is the same as input. When the input data is contaminated (there is a sensor problem), the AANN output will not be the same as its input. Autoassociative Neural Networks and their specifications are discussed in details in section 4.

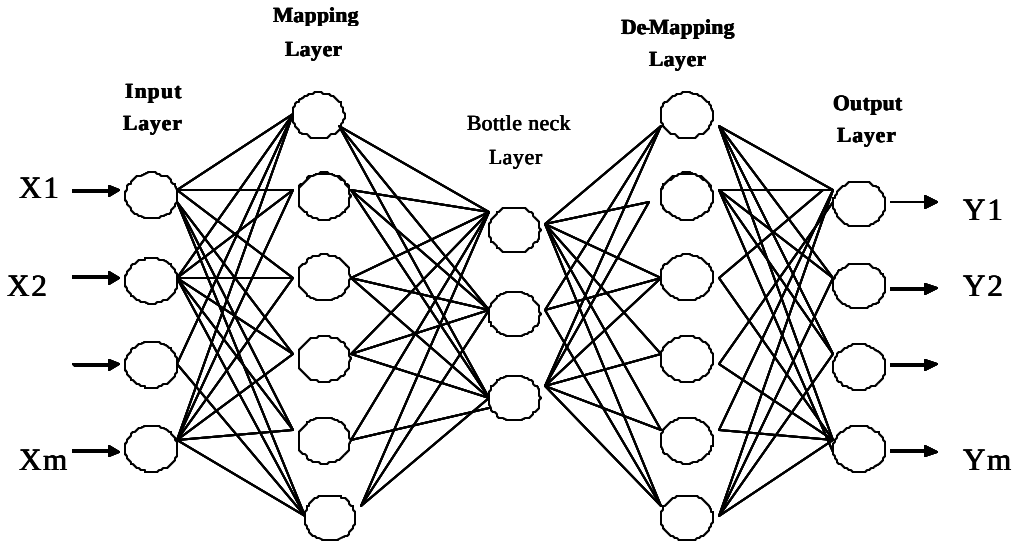

Figure 2-3, AANN architecture

2.2.2 Analysis Section

The analyzer compares the calculated references by the system variables to determine if there is any problem (Task 1). After determination of a sensor problem, the analyzer tries to identify the faulty sensors (Task 2). The Analyzer needs to distinguish between the perturbations coming from the system noise and those coming from the sensor fault. These tasks are different than fault diagnostics. In fault diagnostics the analyzer should locate faulty parts based on the signatures defined based on the system specifications while in sensor diagnostics, the analyzer located faulty sensors without including system specifications. With sensor diagnostics, it is possible to come up with a generic approach as a sensor fault detection mechanism may not necessarily depend explicitly on system specifications.

2.3 Chiller Fault Diagnostics

The majority of fault detection efforts in HVAC systems have focused on fault detection[2] in chiller. This is due to the fact that chiller uses the largest portion of energy in buildings. Several efforts have been done to develop a chiller model for diagnostic purposes. A comprehensive chiller model [20] has been developed by ASHRAE (American Society for Heating, Refrigerating, and Air Conditioning engineers), which is called HVAC 1 primary Toolkit. This model has been written with FORTRAN 77 and used a comprehensive set of subroutines (38 subroutines) to predict the performance of different HVAC components.

Gordon and Ng [21] have also developed their universal thermodynamic model. They used the fundamental relation between chiller COP and the cooling capacity (Heat absorption at the evaporator). Based on this way they were able to predict the chiller performance by identifying certain chillers parameters. The robustness of the Gordon and Ng model has been investigated in [22]. The model has been verified with experimental results with three proposed built system configuration. It has been shown that the model is both flexible and accurate with respect to changes in the system layout.

Another model has been developed with the cooperation of United States Government (USG) and Lawrence Berkeley National Laboratory [23]. DOE-2 according to [24] is the most complex and comprehensive building energy simulation program

[2] In the area of fault detection there are several names like “Fault Detection”, “Fault Diagnosis”, “Fault Determination”, and “Fault Isolation”. The meaning of these names might be slightly different from paper to paper. In this thesis, we have the following definitions:
Fault Detection = Fault Diagnosis : The whole process of fault isolation and fault determination
Fault Determination: The process of checking whether system performance is normal or not
Fault Isolation : The process of isolating and finding the source of the fault

In some papers, specially those in the area of chiller fault detection, these names have slightly different meanings:
Fault Detection = Fault Determination
Fault Diagnosis = Fault Isolation
In those papers, the Fault Detection is know as Fault Detection and Diagnosis (FDD)

available. DOE-2, writhen in FORTRAN, is primary used for building energy analysis and predicts energy uses and costs.

Sreedharam and Haves [25] have evaluated the above three models for FDD purpose (the Gordan and Ng model, the ASHRAE model, and DOE2 model). They have shown that all three models display similar level of accuracy. The Gordan-Ng model has the advantage of being linear in parameters. The ASHREA model may have advantages when refrigerant temperature measurements are also available. They have stated that the DOE2 model can be expected to have advantages when very limited data are available to calibrate the model.

Fault isolation is inherently more complicated than fault detection though the last few decades have seen an absolute proliferation in this field. Stylianou and Nikanpour [26] have used a rule based approach using steady state and transient data to perform fault detection and diagnosis in chiller. Braun and Rossi [27] have proposed the use of a two-step approach towards chiller fault diagnostics. They first used statistical pattern recognition to test for the presence of a fault. If a fault was detected, they followed that up by using a matrix of symptoms to diagnose the fault. Grimmelius [28] has developed a comprehensive symptom matrix to aid the diagnosis process. A nonlinear statistical chiller model output was used as the base case value and was compared to values of selected variables whose deviation or residues were used to build the symptom matrix. Recently Wang and Wang [29] reported a first principle model for fault detection, diagnosis and evaluation (FDD&E) of the temperature sensors and flow meters in a central chilling plant and presented dynamic simulation results. The law-based sensor FDD&E is based on the fundamental mass (steady state) energy conservation relationships.

The approach of Wang and Wang [29] monitors only a part of the sensors in the chiller (temperature sensors and flow rate sensors). They have assumed that data

measured by other sensors are correct. If there is any sensor problem, it should be from temperature or flow rate sensors. This approach cannot be made into a generic sensor diagnostic mechanism. In general we cannot assume that a part of the sensors are correct and try to monitor the others based on that.

On the other hand, the other approaches explained above are based on model-based approach. As it was explained in section 2.2.1, the model-based approach is not an appropriate approach for a generic sensor diagnostic mechanism. The model-based approach relies on the correct input data (which might be measured using sensors). This assumes that the input to the real system and the input to the model are correct (fault free). In Sensor diagnostics, as both input and output data are measured by sensors, none of them is trustable. Model-based approach might be used to monitor a part of the system sensors but not all sensors together. There are also some qualitative approaches for chiller diagnostic but as we explained in section 2.1.1, the qualitative approaches mainly depend on system specifications. Therefore for any system, a separate diagnostic mechanism must be designed. In this project, we are looking for a generic sensor diagnostic mechanism to be applicable in different systems.

3. INTRODUCTION TO NEURAL NETWORKS

3.1 Introduction

Basically a neural network constructs of a set of nodes connected by links. Each node might have several inputs with one output. The output is a function of the inputs (linear or nonlinear). The node output is transmitted to other nodes through the links. Thus each node output is the input for other nodes. This structure is similar to the structure of neural systems in human brains in which nodes are corresponding to neurons and links are corresponding to synapses transmitting signals between neurons.

A weight factor is assigned to each link. When a signal transmits through the link, it is multiplied by the corresponding weight before it reaches the receiving node. Therefore the nodes can reinforce or inhibit signals between two nodes. For convenience, we name the weight associated with a link from node n_i to node n_j as w_{ij} (Figure 3-1). The signal sent from n_i is denoted x_i.

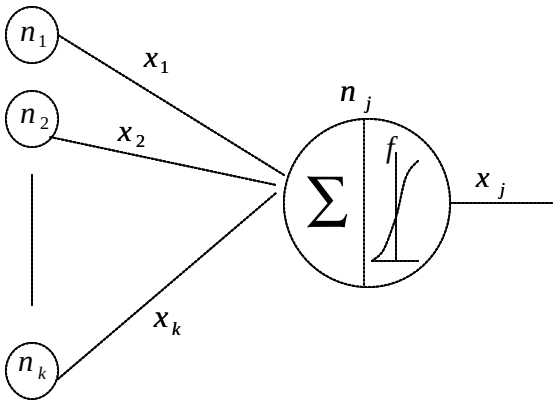

Figure 3-1, Node with several inputs and one output

In feed forward neural networks, each node performs a simple two step operation. First it calculates a weighted sum of all input signals. For example node n_j in Fig 3-1 calculates the following weighted sum in the first step.

$$s_j = \sum_{i=1}^{k_c} w_{ij} \times x_i \qquad (Eq.3-1)$$

Then the calculated sum is fed to a predefined function, f (typically a sigmoid function), to produce the output signal of the node. These two steps are illustrated in Fig. 3-1. Combining these two steps, we have

$$x_j = f(s_j) = f\left[\sum_{i=1}^{k_c} w_{ij} \times x_i\right] \qquad (Eq.3-2)$$

3.2 Architecture of the Neural Network

There are different types of neural networks. Feed Forward neural networks are the most widely used networks. The architecture of a three layer feed forward neural network[3]is shown in Figure 3-2. It organizes its nodes into three layers: the input layer, the hidden layer, and output layer. Each node in the input layer receives external input signal and transmit it to all nodes in the hidden layer through links. Similarly, hidden layer nodes receive input signals from the input layer and send the processed signal to each node in the output layer. Nodes in the output layer generate network outputs by processing transmitted signals from the hidden layer. This Neural Network architecture is called “feed forward” because the signals flow from the input layer to the output layer in a forward direction.

[3] One of the main applications of feed forward neural networks is nonlinear mapping. It has been mathematically proven that a 3-layer feed forward neural network can handle any nonlinear mapping problem with sufficient number of nodes in each layer [35]. However, more layers help the network to have a better performance and learn the relation between the input and output data better.

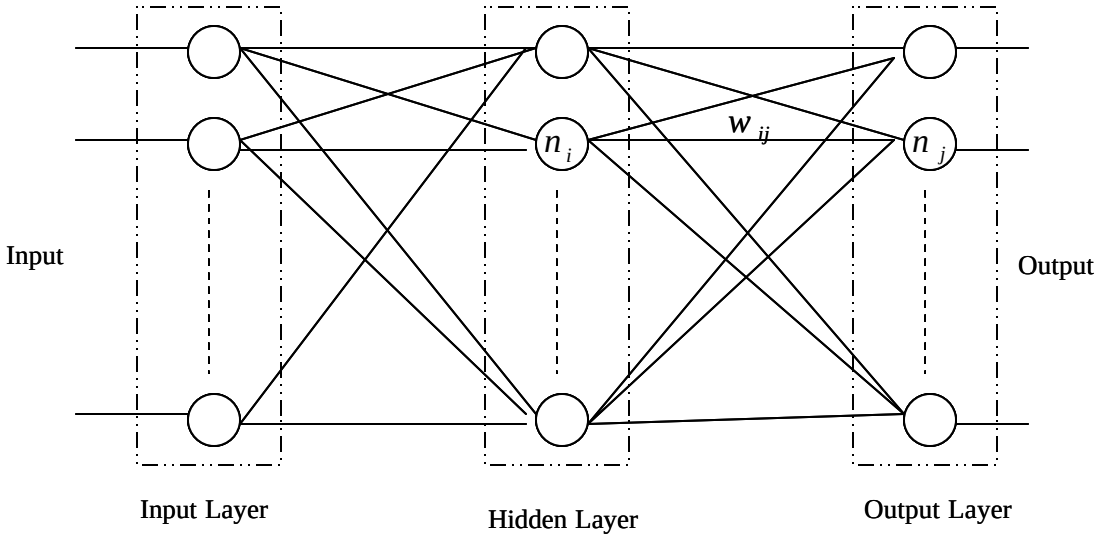

Figure 3-2, Architecture of a three layer feed forward neural network

3.3 Neural Network Training

Backpropagation technique is the most popular approach in training feed forward neural networks. A set of training data is used to train the network. During training, the difference between the neural network output and the training data target gradually reduces. Backpropagation is based on the gradient descent concept.

3.3.1 Gradient Descent

Gradient Descent is a general method for function minimization. As you know the minimum of a function J(x) is defined as the zeros of its gradient.

$$X^* = \arg\min\left[J(X)\right] \Rightarrow \nabla_X J(X) = 0 \qquad (Eq.3-3)$$

However, only in very special cases this minimization function has closed form solution. In other cases, finding a closed form solution is almost impossible or impractical.

Gradient descent finds the minimum of the function in an interactive fashion by moving in the direction of steepest descent. Assume that $J(X)$ ($X = [x_1, x_2, ...x_n]^T$) is the cost function. We want to find the value of X that minimizes J. The symbol ∇J denotes the gradient of J. The gradient is calculated as:

$$\nabla J = [\frac{\partial J}{\partial x_1}(X), ..., \frac{\partial J}{\partial x_2}(X)]$$

$$(Eq.3-4)$$

Therefore ∇J is a vector function of X. The gradient has the property that when it is evaluated as X, it points in the direction of travel from X that will maximally increase J. Therefore to decrease the function J, the value of X should be slightly changed in the opposite direction (i.e. $-\nabla J$). This can be done through the following algorithm.

1- Start with an arbitrary initial condition $X(0)$

2- Compute the gradient $\nabla_X J(X(k))$

3- Move in the direction of steepest descent:

$X(k+1) = X(k) - h\nabla_X J(X(k))$ (h is the learning rate)

4- Go to 2 (until converges)

Practical Problems

The implementation of gradient descent has several problems.

I) Local Minima

Gradient Descent algorithm does not guarantee to find the global minimum even if it converges. It might trap in local minima. There is no way to discriminate between the global minimum and local minimum. For instance, consider Figure 3-3 which shows the cost function, J, as a function of X. The function has two minimums; a global minimum and a local minimum. Gradient descent might converge to either one. It depends on the starting point. There is no way to prevent the solution to converge to a local minimum.

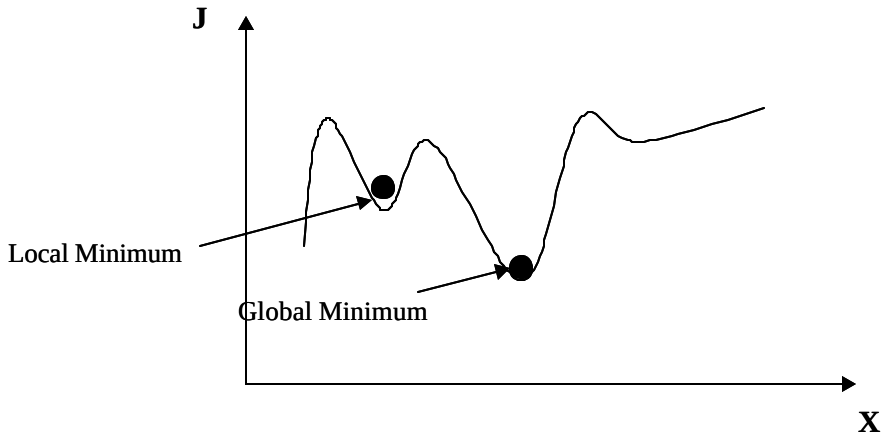

Figure 3-3, Local minima and global minima

Some types of problems are proven to have no local minima. Therefore convergence to a global minimum is assured (provided the step size h is not too large). For other problems we might not know whether local minima exist or whether a minimum found by gradient descent is a local or global minimum.

II) Step Size

In the updating equation of the gradient descent algorithm, there is a constant ($h > 0$) which defines how big the step is made at each iteration. Unfortunately the choice of h is problem-specific and can greatly affect the working of the algorithm. Too small an h will drastically slow down the algorithm while a large value can cause the algorithm to oscillate and become unstable.

3.3.2 Backpropagation Technique

In a feed forward neural network, the main problem is to find the weights, W, such that the network captures the relation between the input and output. Backpropagation is

the well known approach for this purpose. The approach is based on the Gradient Descent. The objective function is defined as the sum-square-error at the outputs.

$$J(W) = \sum_{n=1}^{N_{ex}} \sum_{k=1}^{N_o} \frac{1}{2} (t_k^n - y_k^{<2>n})^2 \qquad (Eq.3-5)$$

Where t_k^n is the desired target of the k^{th} output neuron for the n^{th} example and $y_k^{<2>n}$ is the output of the k^{th} output neuron for the n^{th} example[4].

[4] Note that here we considered a three layer neural network (one hidden layer, input layer, and output layer). The same approach can be implemented to other networks with more number of layers. An example of a hidden layer network is shown in Figure 3-4. The above notations are from [35].

based on the figure 3-4 we have the following notations:

- x_i is the i^{th} input to the network
- $w_{ij}^{<1>}$ is the weight connecting the i^{th} input to the j^{th} hidden layer.
- $net_j^{<1>}$ is the dot product at the j^{th} hidden layer.
- $y_j^{<1>}$ is the output of the j^{th} hidden neuron
- $w_{jk}^{<2>}$ is the weight connecting the k^{th} hidden neuron to the j^{th} output
- $net_k^{<2>}$ is the dot product at the k^{th} output neuron
- $y_k^{<2>}$ is the output of the k^{th} output neuron
- t_k is the target (desired) output at the k^{th} output neuron
- For convenience we will treat biases as regular weights with an input 1

Now based on the above notation the network equations are:

$$net_j^{<1>} = \sum_{i=1}^{N_j} x_i w_{ij}^{<1>}$$

$$y_j^{<1>} = f(net_j^{<1>}) = \frac{1}{1+\exp(-net_j^{<1>})}$$

$$net_j^{<2>} = \sum_{j=1}^{N_{Hj}} y_j^{<1>} w_{jk}^{<2>}$$

$$y_k^{<2>} = f(net_k^{<2>}) = \frac{1}{1+\exp(-net_k^{<2>})}$$

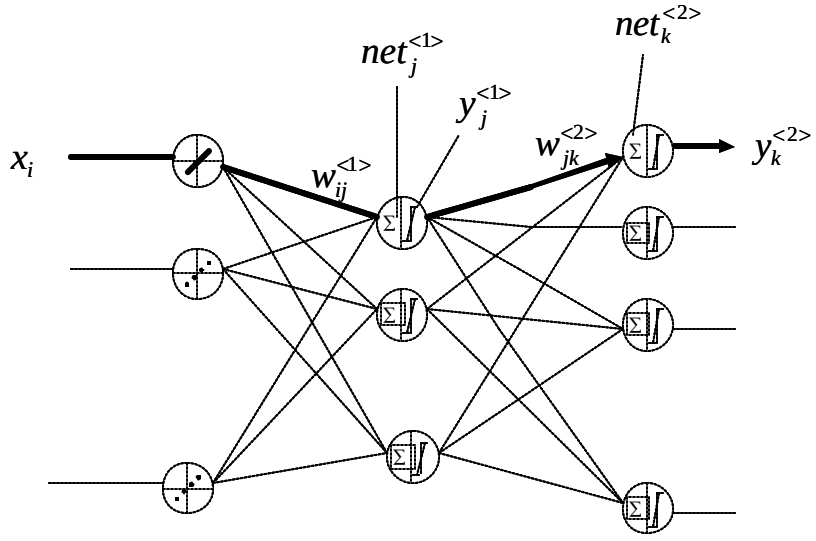

Figure 3-4, A general architecture of neural network

Based on the gradient descent algorithm, the updating equation is:

$$W = W + \Delta W = W - h\frac{\partial J(W)}{\partial W}$$

$$(Eq.3-6)$$

Now the problem is calculation of $\frac{\partial J(W)}{\partial W}$ (for each weight) in terms of what we know: the inputs X_j, the network outputs $y_k^{<2>}$, and the desired targets t_k. For simplicity we find the derivation for one example. This allows us to drop the outer summation.

$$J(W) = \sum_{k=1}^{N_c}\frac{1}{2}(t_k - y_k^{<2>})^2$$

$$(Eq.3-7)$$

Calculation of $\frac{\partial J(W)}{\partial W}$ for Hidden to Output Weights

In a three layer feed forward neural network, there are two sets of weights: hidden to output weights, input to hidden weights. In this section we find the derivation of cost function for hidden to output weights. By using the chain rule, the derivative of J(W) with respect to a hidden-to-output weight is (equations are based on Figure 3-4):

$$\frac{\partial J(W)}{\partial W_{jk}^{<2>}} = \frac{\partial J(W)}{\partial y_k^{<2>}} \frac{\partial y_k^{<2>}}{\partial net_k^{<2>}} \frac{\partial net_k^{<2>}}{\partial W_{jk}^{<2>}}$$

$$(Eq.3-8)$$

Now we calculate each of these terms separately:

$$\frac{\partial J(W)}{\partial y_k^{<2>}} = \frac{\partial}{\partial y_k^{<2>}} \left[\sum_{k=1}^{N_c} \frac{1}{2} (t_k - y_k^{<2>})^2 \right] = \left(t_k - y_k^{<2>} \right)$$

$$(Eq.3-9)$$

$$\frac{\partial y_k^{<2>}}{\partial net_k^{<2>}} = \frac{\partial}{\partial net_k^{<2>}} \left[\frac{1}{1+\exp(-net_k^{<2>})} \right] = \frac{\exp(-net_k^{<2>})}{(1+\exp(-net_k^{<2>}))^2} =$$
$$= \left[\frac{1+\exp(-net_k^{<2>})-1}{(1+\exp(-net_k^{<2>}))^2} \right] \left[\frac{1}{(1+\exp(-net_k^{<2>}))^2} \right] = (1-y_k^{<2>}) y_k^{<2>}$$

$$(Eq.3-10)$$

$$\frac{\partial net_k^{<2>}}{\partial W_{jk}^{<2>}} = \frac{\partial}{\partial W_{jk}^{<2>}} \left[\sum W_{nk}^{<2>} y_n^{<1>} \right] = y_j^{<1>}$$

$$(Eq.3-11)$$

Merging all these derivations yields

$$\frac{\partial (J(W))}{\partial W_{jk}^{<2>}} = (y_k^{<2>} - t_k)(1-y_k^{<2>}) y_k^{<2>} y_j^{<t>}$$

$$(Eq.3-12)$$

For the bias weights, use $y_j^{<t>} = 1$ in the expression above.

Calculation of $\frac{\partial J(W)}{\partial W}$ for Input to Hidden Weights

Using the chain rule, the derivative of J(W) with respect to a IH weight is

$$\frac{\partial J(W)}{\partial W_{ij}^{<1>}} = \frac{\partial J(W)}{\partial y_j^{<1>}} \frac{\partial y_j^{<1>}}{\partial net_j^{<1>}} \frac{\partial net_j^{<1>}}{\partial W_{ij}^{<1>}}$$

$$(Eq.3-13)$$

The second and third terms are easy to calculate from our previous results:

$$\frac{\partial y_j^{<1>}}{\partial net_j^{<1>}} = (1 - y_j^{<1>}) y_j^{<1>}$$

$$(Eq.3-14)$$

$$\frac{\partial net_j^{<1>}}{\partial W_{ij}^{<1>}} = x_j$$

$$(Eq.3-15)$$

Calculation of the first term (i.e. $\frac{\partial J(W)}{\partial y_j^{<1>}}$), however, is not straight-forward since we do not know what the output of the hidden neurons ought to be. This is known as the credit assignment problem, which puzzled connectionist for two decades. The trick to solve this problem is to realize that hidden neurons only contribute to the errors of the output nodes. The derivative of the error with respect to a hidden node's output is therefore the sum of that hidden node's contribution to the errors of all the output neurons.

$$\frac{\partial J(W)}{\partial y_j^{<1>}} = \sum_{n=1}^{N_2} \frac{\partial J(W)}{\partial y_n^{<2>}} \frac{\partial y_n^{<2>}}{\partial net_n^{<2>}} \frac{\partial net_n^{<2>}}{\partial y_j^{<1>}}$$

$$(Eq.3-17)$$

The first two terms in the summation are known from our earlier derivation:

$$\frac{\partial J(W)}{\partial y_n^{<2>}} \frac{\partial y_n^{<2>}}{\partial net_n^{<2>}} = (y_n^{<2>} - t_n)(1 - y_n^{<2>}) y_n^{<2>} = P_n$$

$$(Eq.3-18)$$

The last term in the summation is:

$$\frac{\partial net_n^{<2>}}{\partial y_j^{<1>}} = W_{jn}^{<2>}$$

$$(Eq.3-19)$$

Merging these derivations yields:

$$\frac{\partial J(W)}{\partial y_j^{<1>}} = \sum_{n=1}^{N_c} \frac{\partial J(W)}{\partial y_n^{<2>}} \frac{\partial y_n^{<2>}}{\partial net_n^{<2>}} \frac{\partial net_n^{<2>}}{\partial y_j^{<1>}} = \sum_{n=1}^{N_c} \underbrace{(y_n^{<2>} - t_n)(1 - y_n^{<2>}) y_n^{<2>}}_{P_n} W_{ij}^{<2>}$$

$$(Eq.3-20)$$

Note that what have actually done is propagation of the error term P_n backwards trough the hidden-to-output weights (hence the term backprop, Figure 3-5). The final expression of $\frac{\partial J(W)}{\partial W}$ for input to hidden weights is:

$$\frac{\partial J(W)}{\partial w_{ij}^{<1>}} = \left[\sum_{n=1}^{N_c} (y_n^{<2>} - t_n)(1 - y_n^{<2>}) y_n^{<2>} w_{ij}^{<2>} \right] (1 - y_i^{<1>}) y_j^{<1>} x_i$$

$$(Eq.3-21)$$

For bias weights, use $x_j=1$ in the expression.

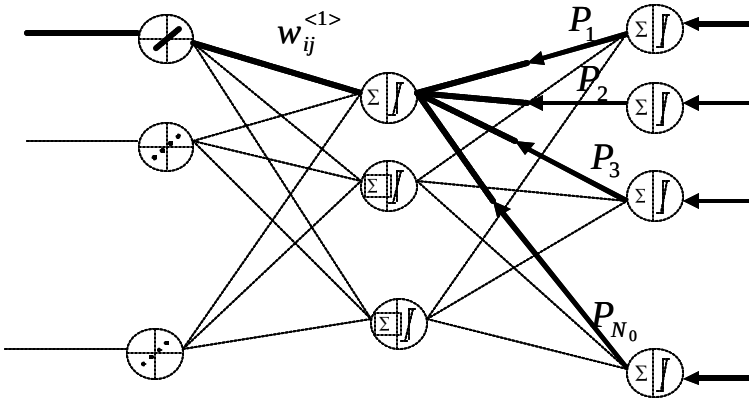

Figure 3-5, Neural network architecture

4. AUTOASSOCIATIVE NEURAL NETWORK

"An Autoassociative Neural Network (AANN) is a network in which the outputs are trained to emulate the inputs over an appropriate dynamic range. Plant variables that have some degree of coherence with each other constitute the input. During training, the interrelationships between the variables are embedded in the Neural Network connection weights" [2]

4.1 Architecture of the Autoassociative Neural Network

Autoassociative Neural Networks are essentially feed forward Neural Networks. Figure 4-1 shows the general architecture of an AANN. AANN architecture contains an input layer, a number of hidden layers and an output layer.

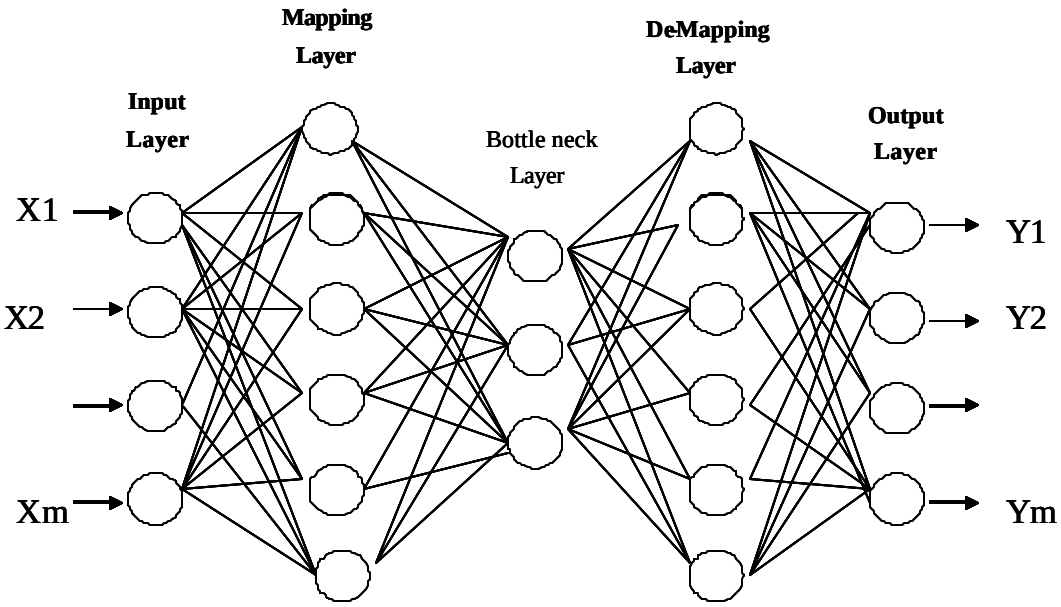

Figure 4-1, AANN general architecture

It is theoretically sufficient for an AANN to have three hidden layers [1]. However, in practice, it has been shown that more hidden layers help AANN to have improved

performance. Additional hidden layers help the AANN to more effectively map the interrelationship among variables [4].

The first hidden layer is called “mapping layer”. The transfer function of the nodes in the mapping layer can be sigmoid or other similar nonlinear functions. The second hidden layer is called the bottleneck layer. The dimensionality of the bottleneck layer is the smallest one in the network (its transfer function can be linear or nonlinear). The third or last hidden layer is called the de-mapping layer, whose nodal transfer functions are nonlinear (usually sigmoid).

4.2 Autoassociative Neural Networks and Identity Mapping

Autoassociative Neural Networks provide an identity mapping. The topic of identity mapping is considered as a subset of general mapping. In identity mapping the input variables are mapped to themselves[5] without making a simple one-to-one mapping. Each output should be a function of all inputs. For example in Figure 4-2 the identity mapping must makes each $X_i, i = 1,2, \text{and } 3$ as a function of X_1, X_2, X_3;

$$X_1 = f(X_1, X_2, X_3)$$

$$X_2 = f(X_1, X_2, X_3)$$

$$X_3 = f(X_1, X_2, X_3)$$

[5] The output is equal to input

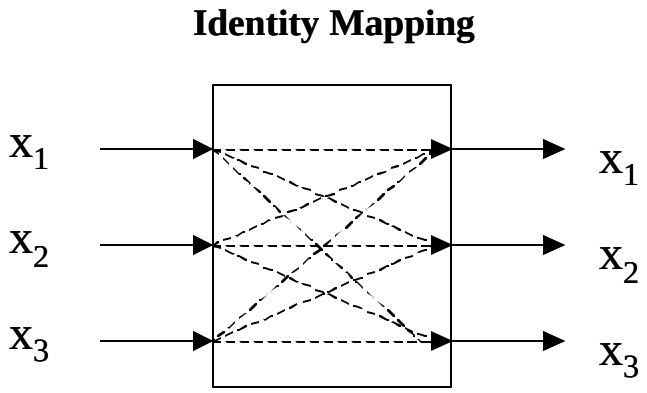

Figure 4-2, Identity mapping

In spite of the fact that there are many techniques for general mapping, most of them are not appropriate for identity mapping. When a general mapping technique is used for an identity-mapping problem, it chooses the easiest solution, which is developing a simple one to one mapping. Unless a mapping technique has an internal force to prevent one-to-one mapping, it is not appropriate for identity mapping.

Autoassociative Neural Networks are appropriate for identity mapping as they have an internal force to prevent one-to-one mapping. The bottleneck layer (the second layer) plays this role. During training, the bottleneck layer forces the AANN to encode the inputs (compress the inputs) then decode them (decompression) to produce the network outputs. The training process selects the network weights such that the re-created measurement vector at the output layer matches the input as closely as possible, in least square sense, over the set of training examples.

4.3 Autoassociative Neural Network Training

Since AANNs are feed forward neural networks, they are trained based on back propagation technique[6]. During training, the network learns the interrelationship among

[6] Back Propagation has been explained in section 3.

the variables. The set of variables fed to the AANN should be correlated. This is checked by analyzing the covariance matrix of the data[7]. If the variables are not correlated, AANN will not find any coloration. In this situation, the network will memorize the data.

4.3.1 AANN Training in Noisy Situations

Training is critical with neural networks. Good training leads to a good generalization capability in which the trained network can generalize samples out of the training set with a good approximation. During training, the network must learn the interrelation between the input and output data to a specified accuracy. Too high accuracy has opposite effect. If the network learns the data too perfectly, its generalization capability falls down. It cannot generalize samples out of the training set well.

For example consider Figure 4-3, which shows a set of data (data with one input and one output).

---

[7] Given n sets of variables denoted $\{X_1\},.....,\{X_n\}$, the covariance $s = \mathrm{cov}(x_i, x_j)$ of X_i and X_j is defined by:

$$\mathrm{cov}(x_i, x_j) \equiv \left\langle (x_i - m_i)(x_j - m_j) \right\rangle = \left\langle x_i x_j \right\rangle - \left\langle x_i \right\rangle \left\langle x_j \right\rangle$$

Where $\langle\ \rangle$ is the expectation value, $m_i = \langle x_i \rangle$, and $m_j = \langle x_j \rangle$ respectively. The matrix (V_{ij}) of quantities $V_{ij} = \mathrm{cov}(x_i, x_j)$ is called the covariance matrix.

The covariance of two variables x and x provides a measure of how strongly these variables are correlated.

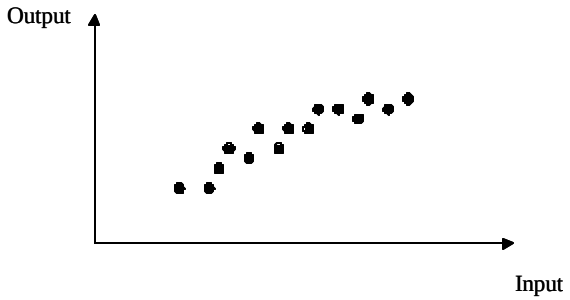

Figure 4-3, Data set with one input and one output

We want to train a neural network to find the relation between the input and output. Two possible solutions are shown in Figures 4-4 and 2-5.

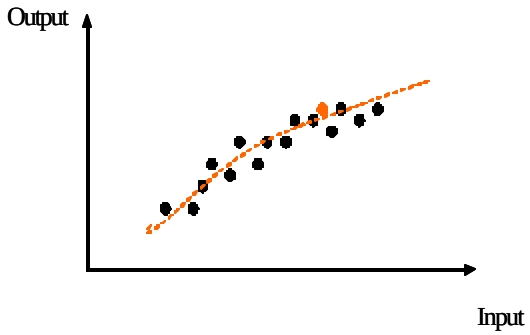

Figure 4-4, A good solution found by neural network

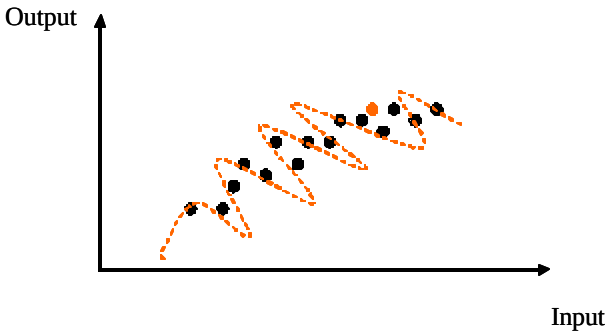

Figure 4-5, A bad solution found by neural network

In Figure 4-4 the network has found a curve that is not so accurate. It maps the samples with a good approximation but not so perfectly. In Figure 4-5 the curve is perfect. It maps the samples exactly.

Now let see what would be the network response to a sample out of the training set (red point). In the case of Figure 4-4, the network still maps the point with a good approximation while in the case of Figure 4-5 the network generalization is poor. This is due to the fact that in the case of Figure 4-5 the network has memorized the data. It has learned how to map each training sample exactly but is has not learned the relation between input/output. Figure 4-4 is an example of good training while Figure 4-5 is a bad one. Situations like Figure 4-5 usually happen when the network is forced to learn the data too perfectly. The network sees that the only way to reach the defined accuracy is to memorize the samples one by one.

When a data is noisy, it has a level of uncertainty. The uncertainty level depends on the noise level. Higher noise causes higher uncertainty in individual measurements. If a neural network dealing with noisy data is forced to learn the data with accuracy beyond the uncertainty level, it will memorize the data (like what we had in Figure 4-5). At first the network tries to learn the data. It tries to find a relation between the input and output. When it sees no relation can satisfy the desired accuracy, it starts to memorize the samples one by one. In this situation the learning process takes a lot of time (symptom). As AANNs are feed forward neural networks, we might expect the same performance in noisy situations. Lets show it through an example.

Consider the identity mapping problem[8] shown in Figure 4-6 (an 8-sensor system problem). The training data is shown in Figure 4-7.

[8] This identity mapping problem is an 8-sensor system (a chiller model system which has 3 inputs and 5 outputs). The data used in this example is from the performance of that model. The process of data generation has been explained in section 5. Here we just use the data got from the model performance to train the AANN.

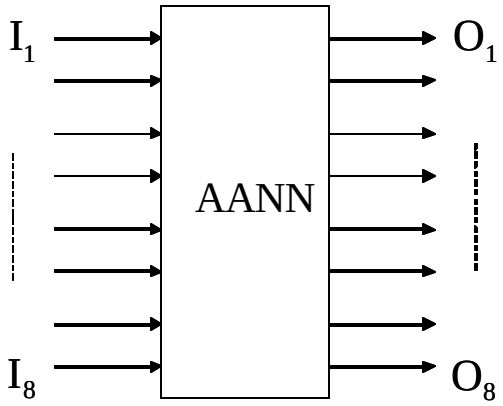

Figure 4-6, Identity mapping with 8 inputs/outputs

The AANN performance is compared in noise free and noisy conditions.

I) Noise free condition: The AANN was trained based on the data shown in Figure 4-7. The selected AANN was an 8-11-5-11-8 neural network[9]. The nodal functions were sigmoid. The training accuracy was set at 0.00001[10]. The AANN reached the desired accuracy in less than 20 epochs[11].

II) Noisy condition: In this situation the data was contaminated by 1% noise[12]. Again the AANN was a 8-11-5-11-8 neural network and the nodal function was sigmoid. At first the training accuracy was set at 0.00001. After 300 epochs, AANN still did not reach the desired accuracy. Figure 4-8 shows the relation between the epochs and the sum square error. As you see, AANN does not seem to reach the desired accuracy even after 400 or 500 epochs. This means that the 0.0001 training accuracy is beyond the

[9] 8-11-5-11-8 means a network with 8 nodes in the first layer, 11 nodes in the second layer, … , and 8 nodes in the last later.

[10] Training accuracy is a standard term in training neural networks. It determines how precisely the network must be trained. When the training accuracy is, say 0.00001, this means that after training, the sum square error between the inputs and outputs of the AANN should be less than 0.00001.

[11] Epoch is another standard term training neural network area. There are two approaches to train neural networks: batch and ... In batch at first the whole data is fed to the network. Then the weights are updated based on the error of the whole data. After updating the weights, the data is again fed to the network and weights are updated again based on the new error. This process continues until the network reaches the desired accuracy. The number of times that data is fed to the network is named epoch.

[12] In section 5, the details about the process of inducing noise is explained.

uncertainty level caused by noise. In the second attempt, the training accuracy was set at 0.001. This time the AANN reached the desired accuracy in less than 20 epochs.

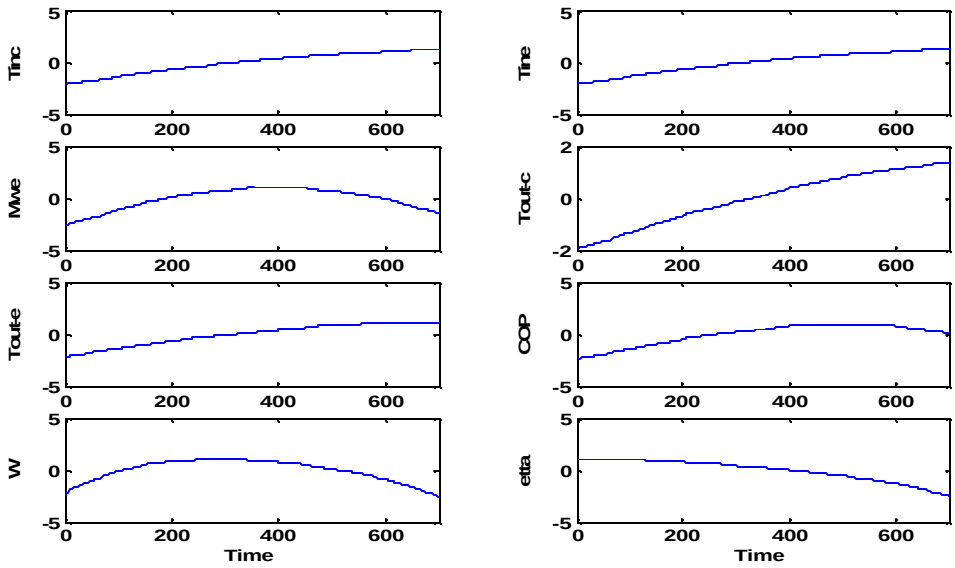

Figure 4-7, Training set

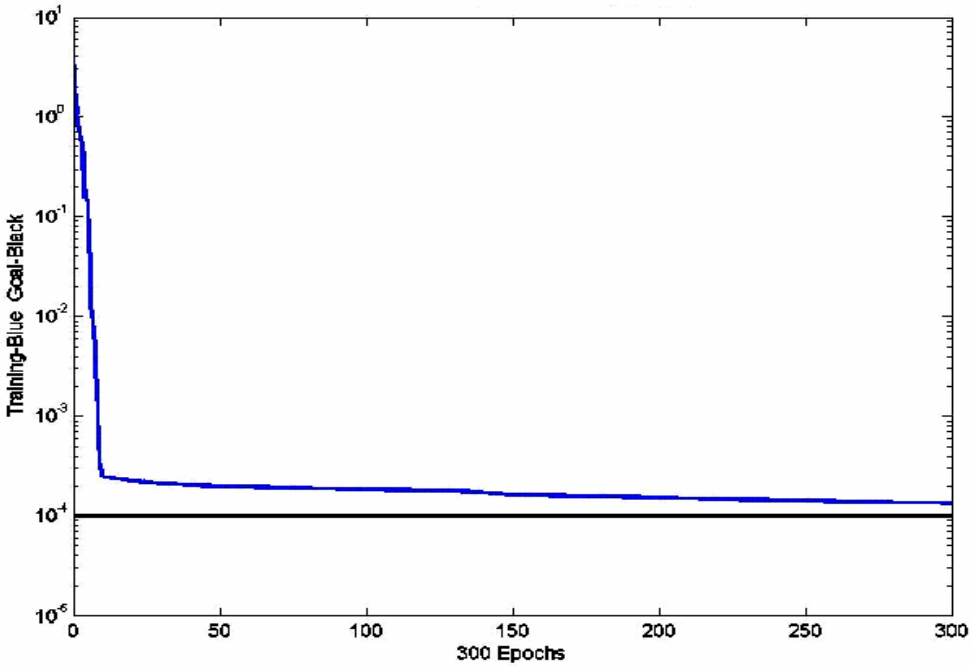

Figure 4-8, The relation between sum squared error and 300 epochs

5. CHILLER MODEL AND DATA GENERATION

A chiller model was used as a system to test the developed diagnostic approach using synthetic data. The model was previously developed as a part of a master thesis. The detailed specifications of the model can be found in [3]. In this section, we explain the model's inputs/outputs, the process of data generation, and the method of inducing noise to the system.

5.1 Model Inputs/Outputs

The model simulates the performance of a reciprocating, vapor compression cycle chiller (Figure 5-1).

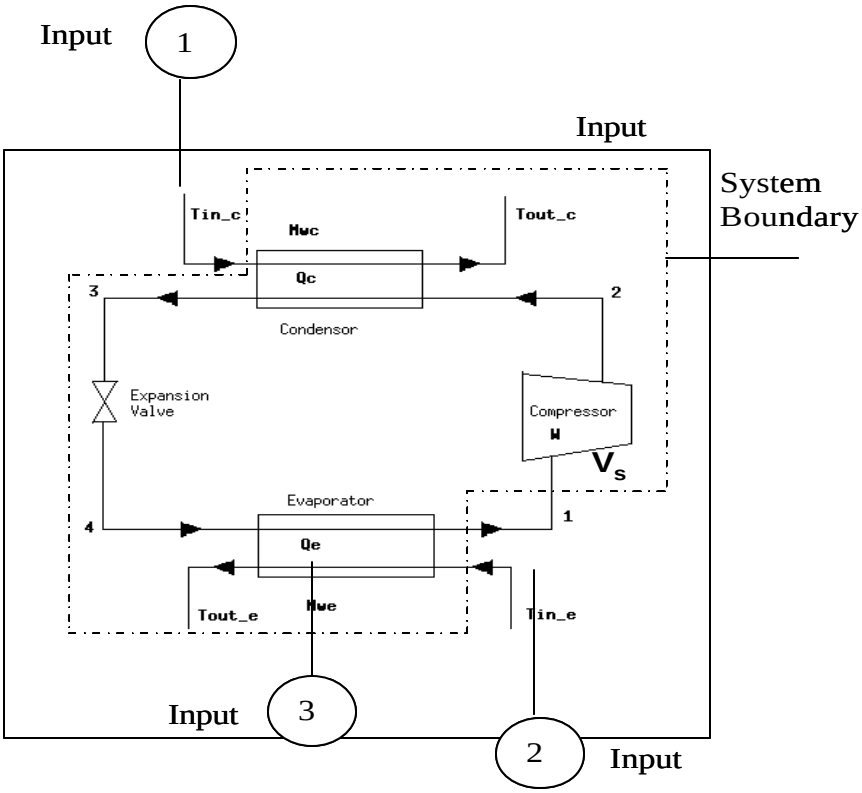

Figure 5-1, System boundary of the chiller model, from [3]

The inputs to the chiller model are (Figure 5-2):

1- Temperature of water into condenser: Tw_{in_c}

2- Temperature of water into evaporator: Tw_{in_e}

3- Mass flow rate of water in evaporator: Mw_{evap}

The outputs to the model are (Figure 5-2):

1- Temperature of water existing condenser: Tw_{out_c}

2- Temperature of water existing evaporator: Tw_{out_e}

3- Coefficient of the chiller performance: COP

4- Power consumed by compressor: W

5- Efficiency of Compressor: h_{comp}

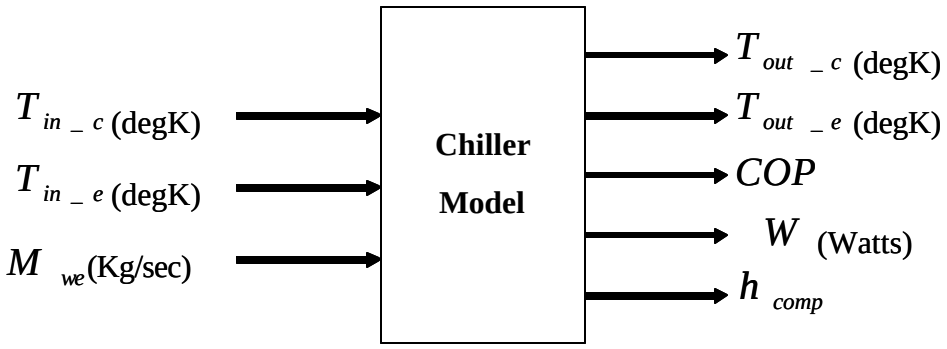

Figure 5-2, Chiller model inputs/outputs

The input ranges are[13]:

$T_{in_c} \approx 289 \rightarrow 309 \quad \deg K$,

$T_{in_e} \approx 275 \rightarrow 288 \quad \deg K$

$M_{we} \approx 0.22 \rightarrow 0.64 \quad Kg/\sec$

[13] These ranges have been defined by designers [3]

5.2 Data Generation

From 1000 samples generated by using the model, 300 samples were randomly chosen as the test set and the remainder (700 samples) were used as the training set. The training set is used to train the network and the test set is used to test the trained network. Figures 5-3 shows the complete dataset[14]. Figure 5-4 shows the training set. The data has been sorted in Figure 5-4. Figure 5-5 shows the test set (sorted) .In another test, we prepared a data with 4000 samples (3000 samples as training set and 1000 samples as test set). Since the same results occurred with both datasets, the smaller data set was used.

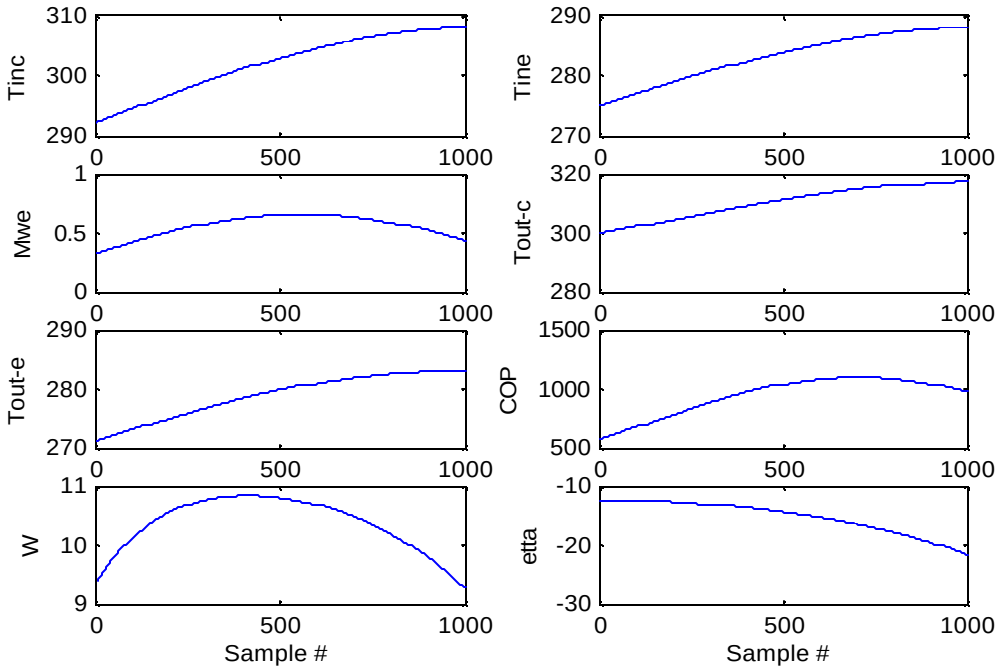

Figure 5-3, Whole data with 1000 samples

[14] As it was explained, this data has been got from the chiller model developed by Rahul [3]. However, the data might be unreal comparing with a real chiller (for example it seems that the COP generated by model is unreal and the chiller efficiency is negative). In this project this data has only been used to verify the diagnostic approach. The unreality of the data does not have any effect on the results.

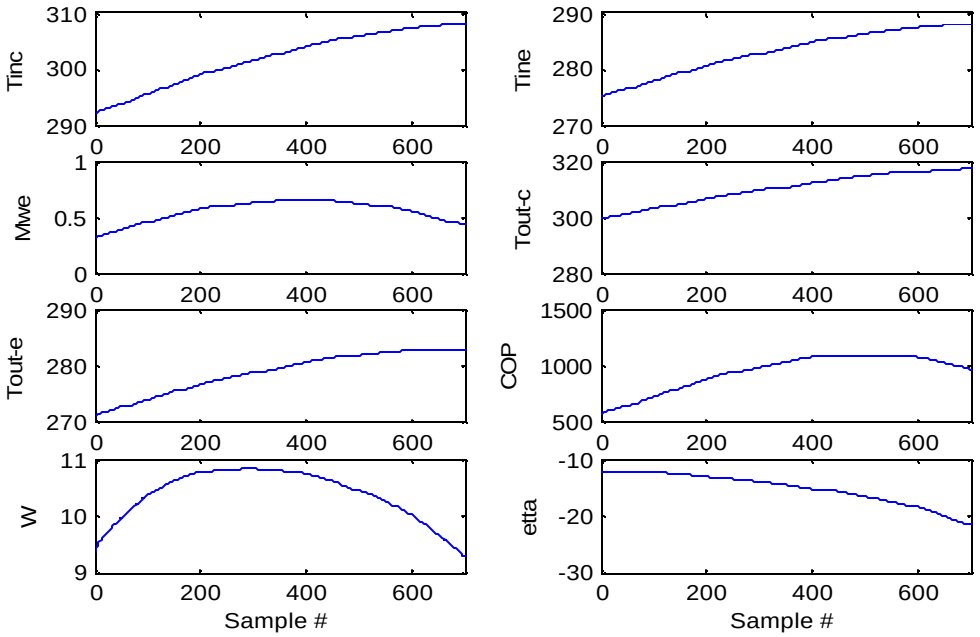

Figure 5-4, Training set (sorted)

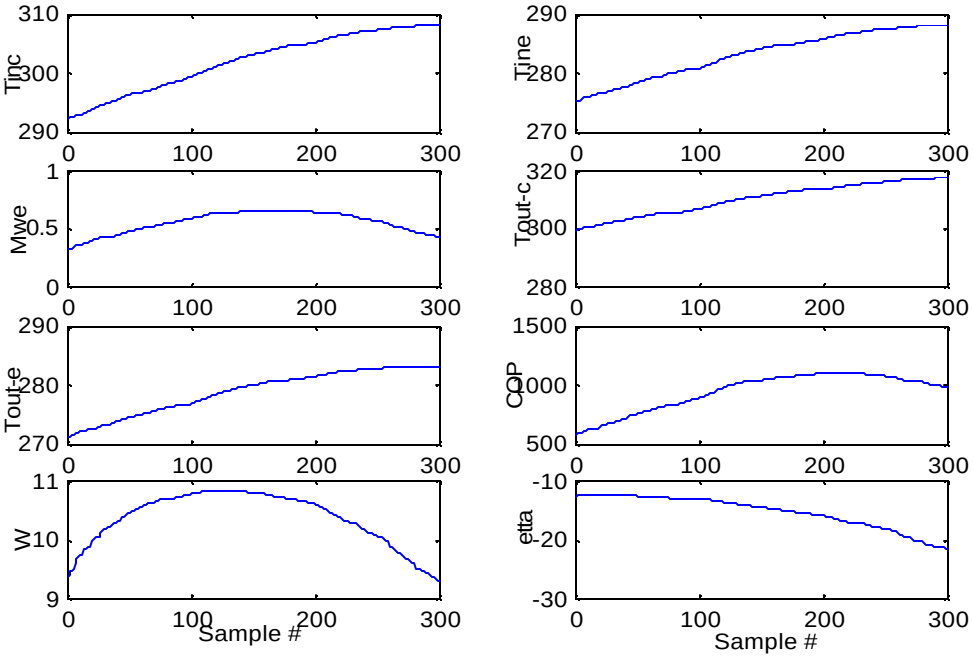

Figure 5-5, Test set (sorted)

5.3 Normalization

The data was normalized such that the mean value and variance of each variable became zero and one respectively[15]. This normalization keeps any single sensor from biasing the results due its large numerical values. Figure 5-6 shows the normalized training set and Figure 5-7 shows the normalized test set.

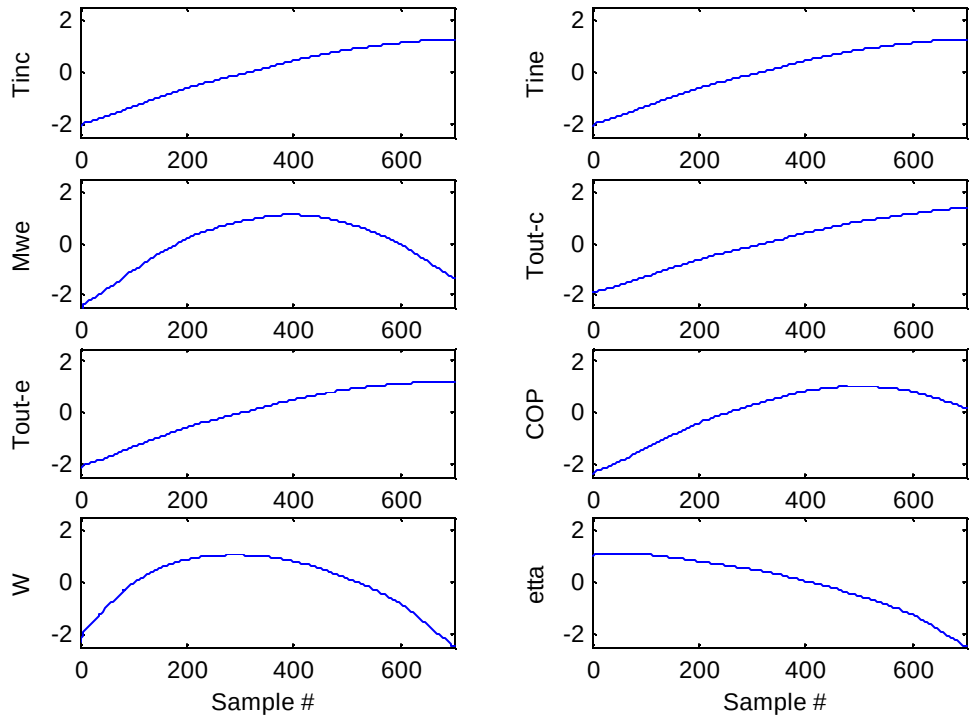

Figure 5-6, Normalized training set (sorted)

[15] If we have a variable X with the mean value of X_{mean} and the variance of X_{var}, we can normalize the X to have the mean value and variance of zero and one respectively based on the following equation:

$$X_{normalized} = \frac{X - X_{mean}}{\sqrt{X_{var}}} \qquad (Eq.(5-1)$$

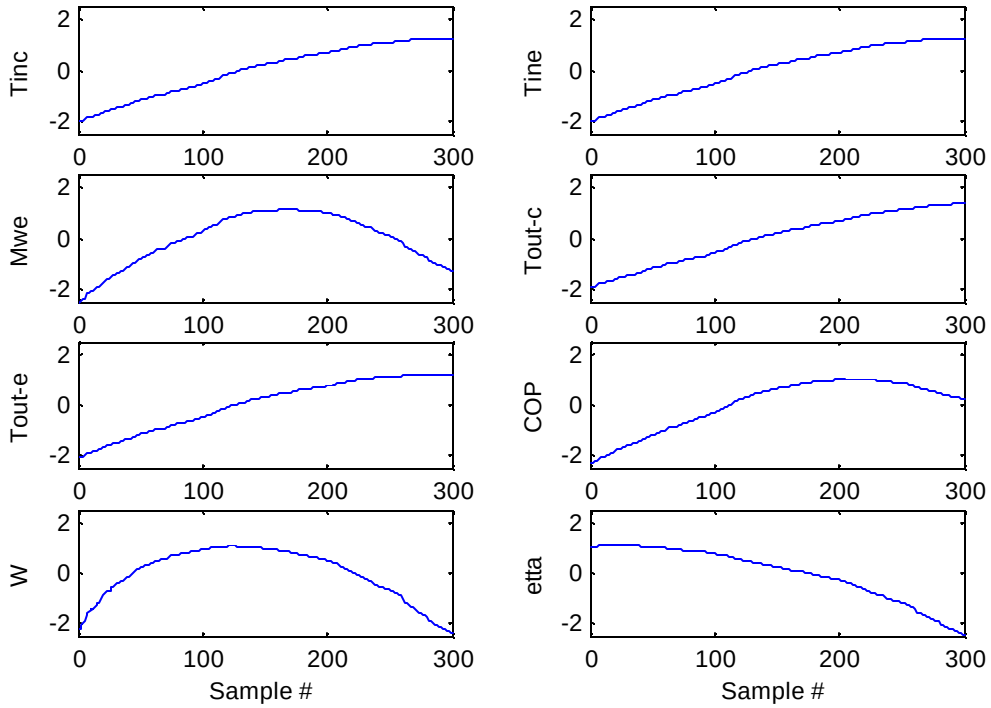

Figure 5-7, Normalized test set (sorted)

5.4 Inducing Noise

As a part of this project, the effect of noise on sensor diagnostics is studied. Therefore it is necessary to generate noisy data in different situations. As you know the existing noise in a system does not depend on the current value of the system variables. The noise level is defined based on the variable changes. In order to induce A% noise to a variable X, we have:

$$X_{noisy} = X + \frac{\max(X) - \min(X)}{100} \times A \qquad (Eq.5-2)$$

In order to induced noise to the data got from the chiller model, at first we need to find the range of variables. The noise is induced before normalization. Figures 5-8 and 5-9 show the training data and the test data with 2% noise. Figures 5-10 and 5-11 show the same data with 5% noise.

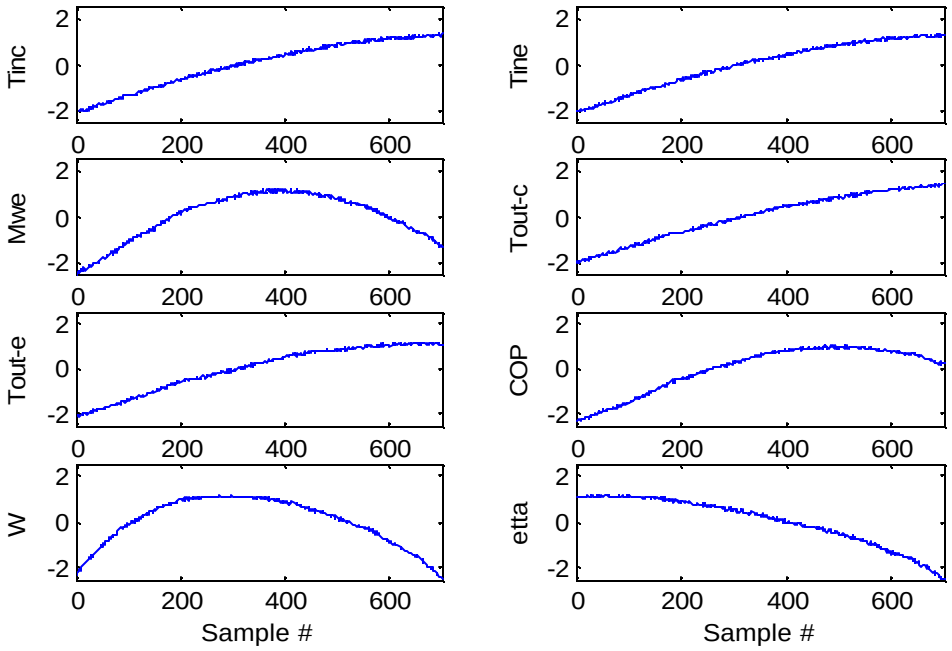

Figure 5-8, Training set with 2% noise

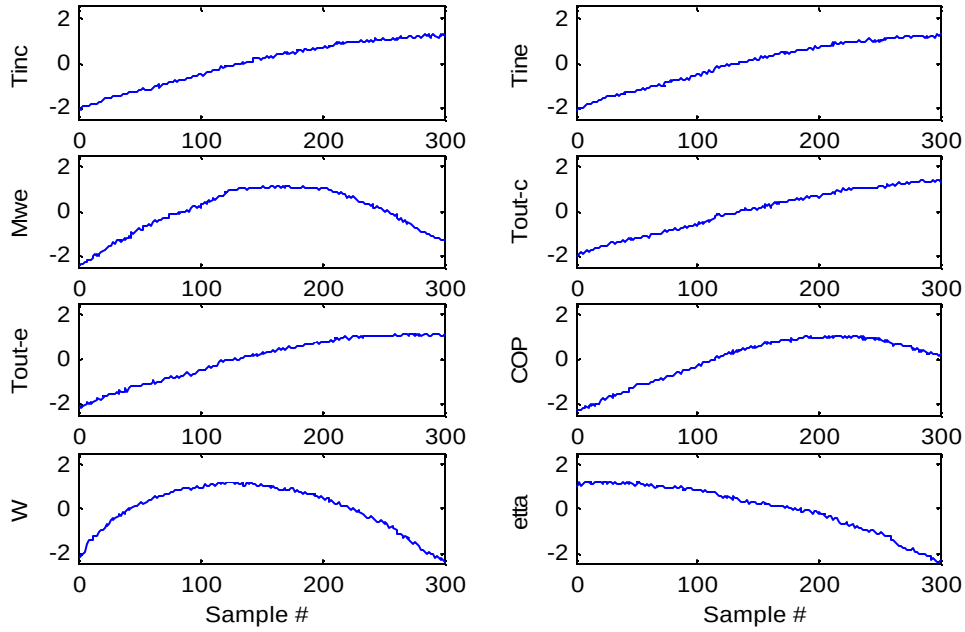

Figure 5-9, Test set with 2% noise

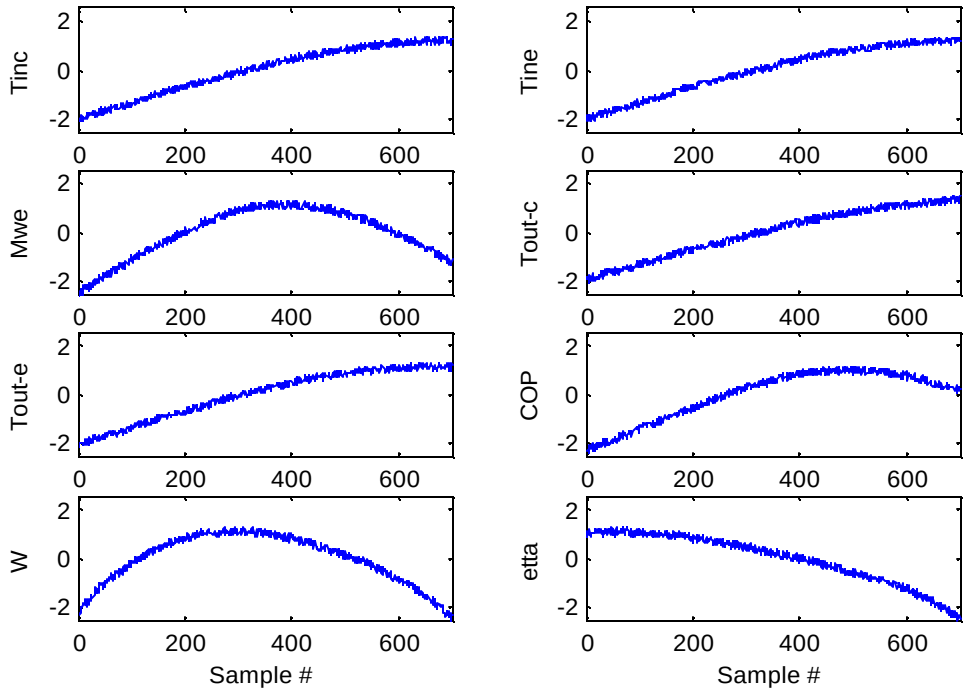

Figure 5-10, Training set with 5% noise

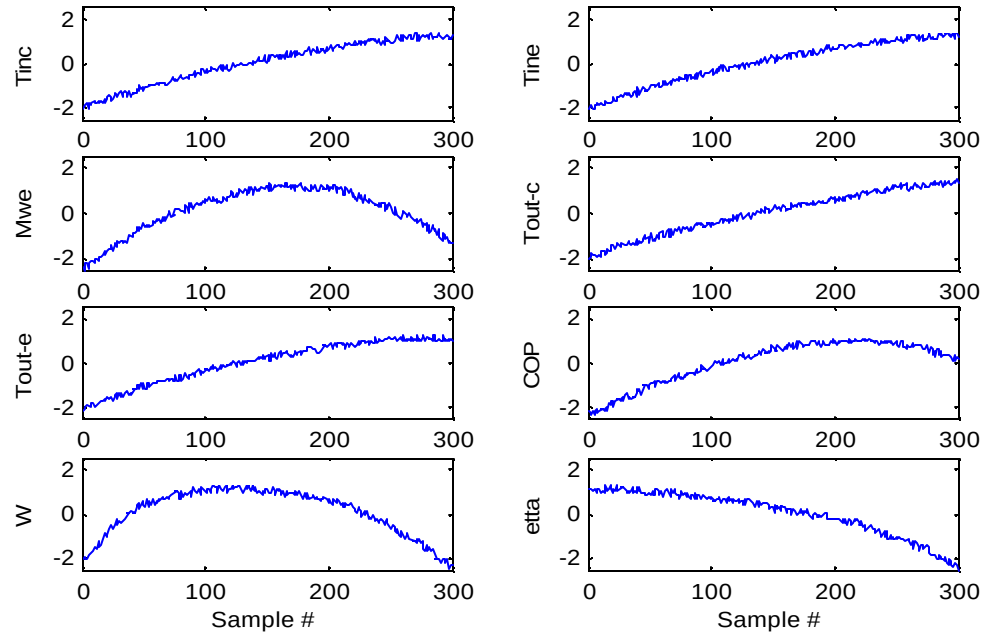

Figure 5-11, Test set with 5% noise

6. ENHANCED AUTOASSOCIATIVE NEURAL NETWORK (E-AANN)

AANNs are appropriate tool for sensor diagnostics as they reproduce the input data at the output. In sensor diagnostics, all data measured from the real system (the input and output variables) constitute the input vector to the AANN (Figure 6-1).

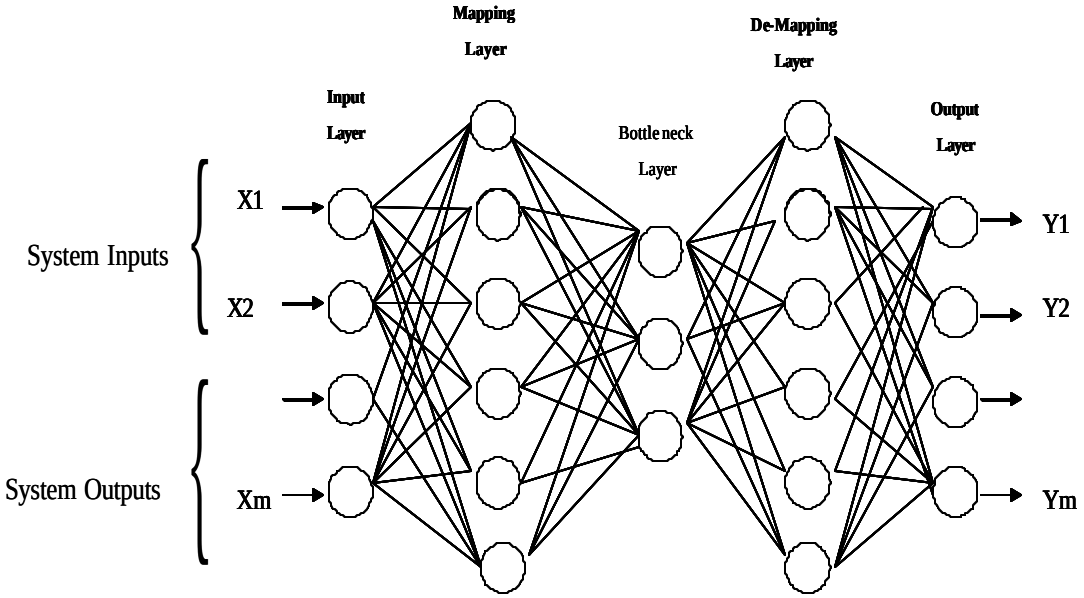

Figure 6-1, AANN structure

The AANN is trained to match the inputs as closely as possible over the training set. When non-faulty data is fed to the trained AANN, the difference between the input and output of the AANN is ideally zero. When the data is contaminated (a sensor is faulty), the difference between the input and output of the AANN will be non-zero. The AANN

approach can be used to determine whether there is a sensor problem but the issue of locating the faulty sensors still remains.

6.1 Hines Approach

At the University of Tennessee, Hines and his colleagues have proposed a method to locate the faulty sensors using AANN [5 and 6]. They believe that the difference between each AANN input and output contains enough information to capture faulty sensors. As noted by Hines [6] "When a sensor that is 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 (Figure 6-2). The estimate sensor output is then compared to the actual sensor output. The difference is called an error or residual. The residual normally has a mean of zero and a variance related to the amount of the noise in the sensor's signal. When a sensor is faulty, it's associated residual's mean or variance changes. This can be detected via statistical decision logic."

This means that when the difference between each AANN input and output is zero, the corresponding sensor is healthy otherwise the sensor has a problem. However, we were unable to reproduce the same results as Hines, perhaps due to an error or unreported assumptions in the original work. We believe that due to the inherent non-orthogonality of the AANN, when one of the AANN inputs is contaminated, this affects all the AANN outputs. The mapping function between the AANN input and output is nonlinear (due to the nonlinearity of the node functions) so we cannot expect to see no change when there is perturbation in one of the inputs (even if that perturbation is small). Finding the difference between the input and output of the AANN can be used to determine sensor problems but is not sufficient to localize the faulty sensors.

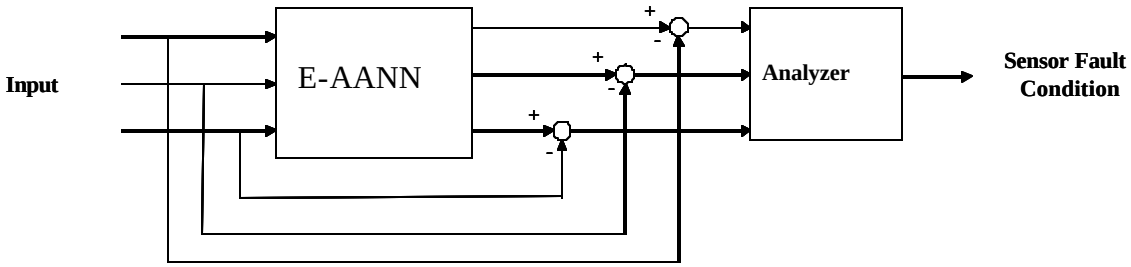

Figure 6-2, Approach recommended by Hines

6.2 Enhanced Autoassociative Neural Network

In order to remedy this situation, an extension to the AANN concept has been developed. This extension is based on the fact that when the input to the AANN is fault free, AANN output will be the same as input. In this situation the Sum Squared Error between the inputs and output should be ideally zero:

$$J = (X - Y)^T (X - Y) = \sum_{i=1}^{N} (Y_i - X_i)^2 = 0$$

Where:

$$X = [X_1, X_2, \ldots . X_m]^T$$
$$Y = [Y_1, Y_2, \ldots \ldots . Y_m]^T$$

From a procedural standpoint, the proposed extension can be viewed as an optimization process that determines $X^* = \arg\min(J)$:

$(X_1, X_2, \ldots\ldots\ldots\ldots X_m)$

$J_{\min} = \infty$

$For \quad i = 1,2,\ldots\ldots\ldots\ldots, m$

$\{$

$\Delta X = \dfrac{Max(X_i) - Min(X_i)}{N}$ Where N is user selectable

$J_{i,\min} = \infty$

$For \quad k = 1,2,\ldots\ldots\ldots\ldots, n$

$\{$

$\hat{X}_k = [X_1, X_2, \ldots\ldots X_i', \ldots\ldots\ldots X_n] \qquad where \quad X_i' = \min(X_i) + \Delta X_i * k$

$\hat{Y}_k = Net(\hat{X}_k)$

$J_k = (\hat{X}_k - \hat{Y}_k)^T (\hat{X}_k - \hat{Y}_k)$

$if \quad J_k < J_{i,\min} \quad then \quad J_{i,\min} = J_k$

$\}$

$if \quad J_{i,\min} < J_{\min} \quad then \quad J_{\min} = J_{i,\min} \; and \quad X^{\bullet} = Arg\min(J_{\min})$

$\}$

Once X^* has been determined, the difference, $X^* - X$, can be used to identify the faulty sensor and the extent of the fault induced error.

E-AANN can be used to locate the faulty sensor and also estimate the real value of the faulty sensor output. The input to the E-AANN is the data measured from the real system (both input variables and output variables) and the output to the E-AANN is the data reconstructed by the E-AANN. If the input data is fault free, then E-AANN output will be the same as the input and the difference between the output and the input will be zero. When one of the inputs is contaminated, the corresponding output will not track the input and their difference will not be zero anymore. In order to locate the faulty sensors, we only need to find the difference between each E-AANN input and output. If the difference is roughly zero, the input is fault free and the corresponding sensor is healthy. If the difference is not zero, the input is contaminated and the corresponding sensor is faulty.

6.3 Examples

Consider the chiller model explained in section 5 (an 8-sensor system). The model was used to generate 1000 sets of steady state data. 700 data sets were used to train the network and the remaining 300 sets were used as the test set. The AANN embedded in the extended E-AANN is an 8-11-5-11-8 neural network.

Example 1: In this example the E-AANN performance is evaluated in a range of time domain. Figure 6-3 shows the test data with 1% noise and no induced fault. Figure 6-4 shows the same test data but this time, sensor #3 has drift error. E-AANN output is shown in Figure 6-7. For better comparison we have zoomed in on the third graph of Figure 6-3 (sensor #3 output in non-faulty condition), the third graph of Figure 6-5 (sensor #3 output after inducing drift error), and the third graph of Figure 6-7 (the data reconstructed by E-AANN) in Figure 6-6.

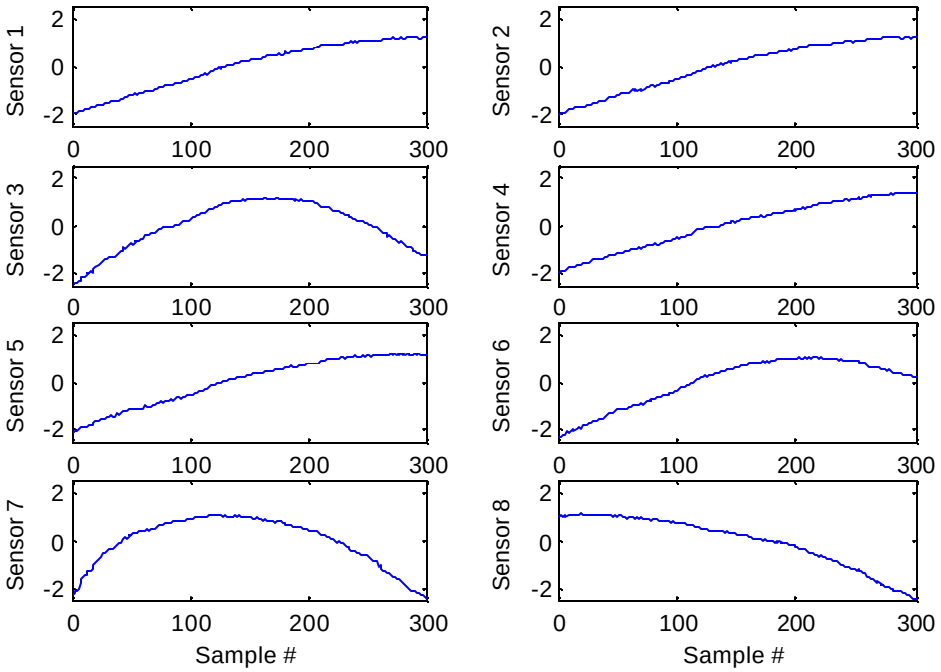

Figure 6-3, Test data with 1% noise

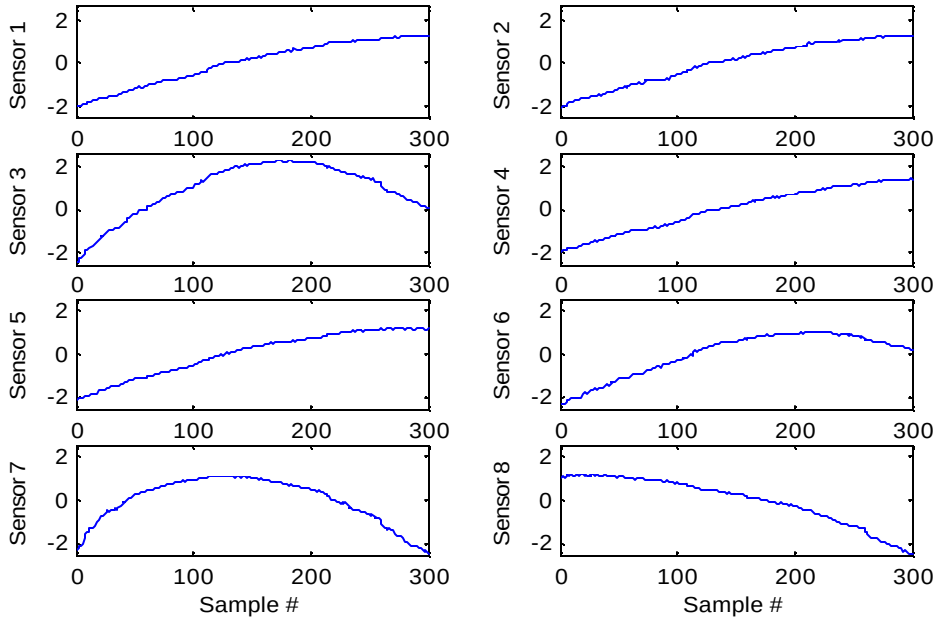

Figure 6-4, Test data with 1% noise, sensor #3 has drift error

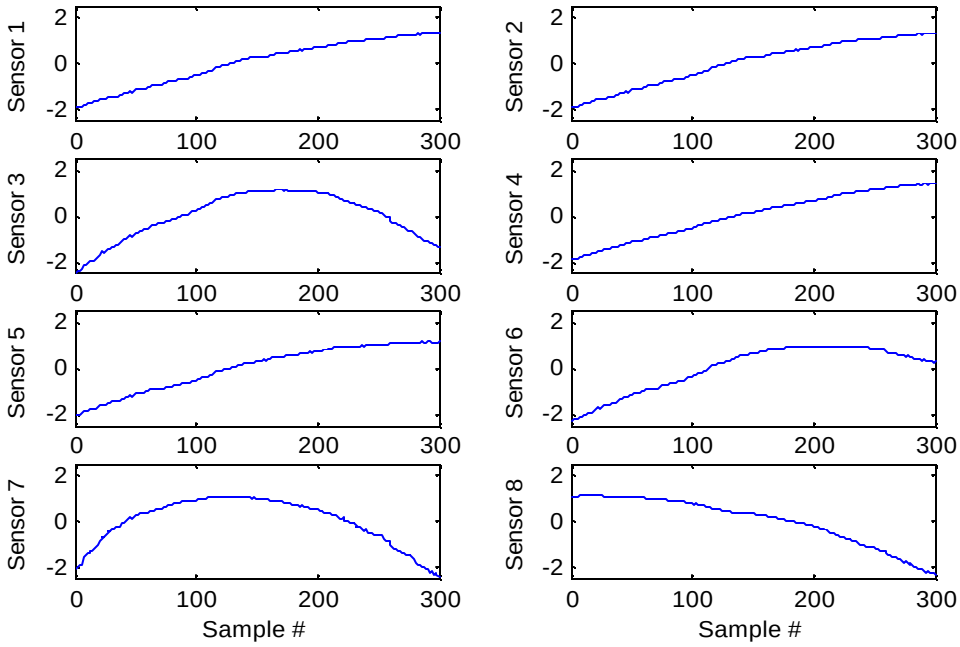

Figure 6-5, E-AANN output

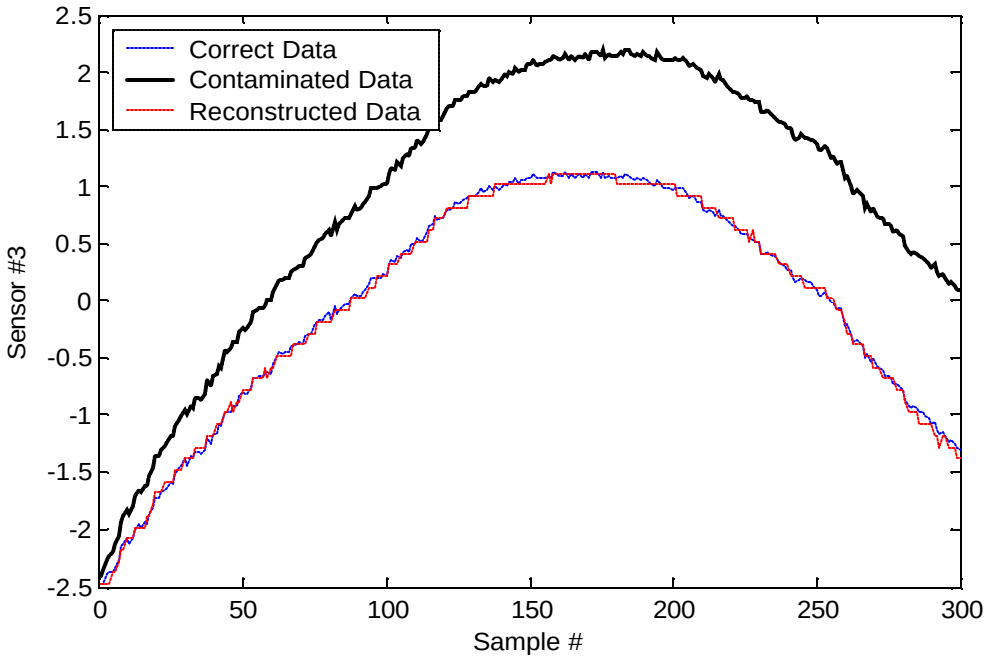

Figure 6-6, Sensor #3 output before contamination, after inducing drift error, and the data reconstructed by E-AANN

Example 2: In this example one of the samples in the test set is chosen. The first sensor is intentionally induced an error (offset). Then the contaminated sample is fed to the E-AANN. The original data, contaminated data and reconstructed data are shown in Table 6-1. Figure 6-7 shows the changes of the cost function. You see that the cost function is minimum for sensor #1 somewhere between -1.5 and -2 (note that the data is normalized). This means that E-AANN has found the first sensor faulty and its real value is between -1.5 and -.2. Figure 6-8 shows the counter plot of the cost function.

Table 6-1. Original data, contaminated data and reconstructed data

	Synthetic Data	Contaminated Data Value	E-AANN Output
Sensor #1	-1.674	0.1	-1.6749
Sensor #2	-1.674		-1.674
Sensor #3	-1.764		-1.764
Sensor #4	-1.628		-1.628
Sensor #5	-1.715		-1.715
Sensor #6	-1.879		-1.879
Sensor #7	-0.880		-0.880
Sensor #8	1.0486		1.0486

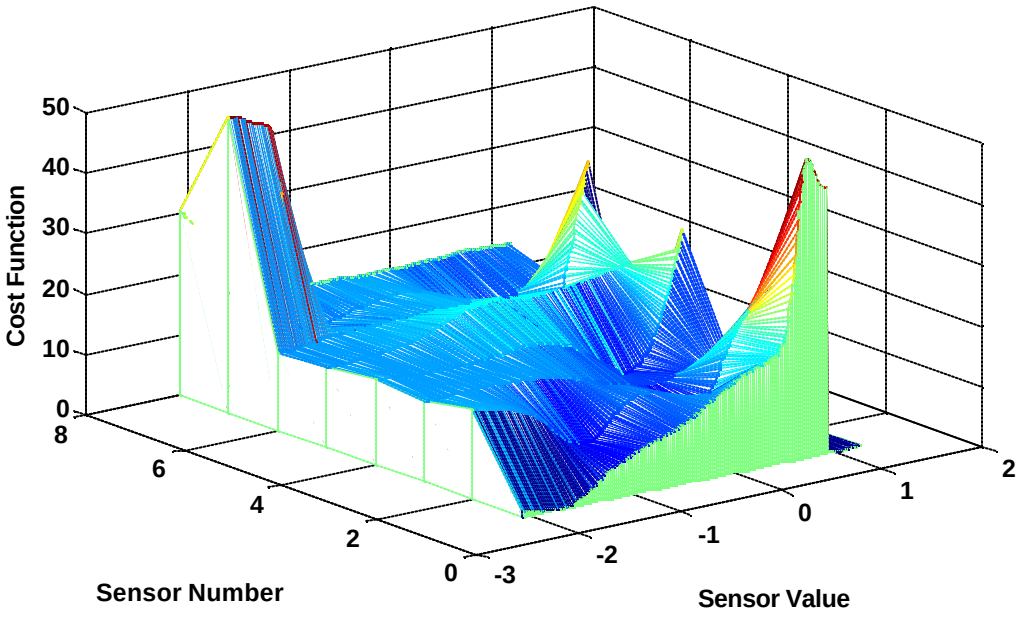

Figure 6-7, Cost function

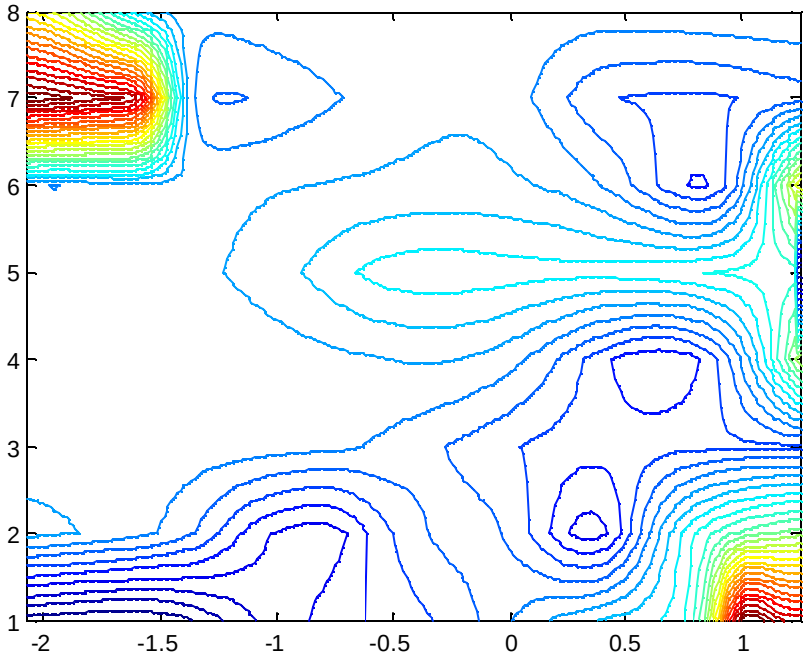

Figure 6-8, Contour plot of cost function

7. E-AANN SPECIFICATIONS

The E-AANN can handle two tasks simultaneously; 1- isolation of faulty sensors and 2- reconstruction of faulty sensor outputs. In this section we characterize E-AANN performance. We will discuss E-AANN accuracy in reconstructing sensor outputs, influential parameters on E-AANN accuracy, E-AANN ability to locate faulty sensors in noisy conditions, and influential parameters on E-AANN performance in noisy situations.

7.1 E-AANN Accuracy

It is important to see how precisely E-AANN can reconstruct sensor outputs. In order to improve the E-AANN accuracy, at first we need to know effective parameters. Our results show that in noise free situations, E-AANN accuracy is mainly affected by two parameters: I) AANN accuracy and limitations and II) E-AANN step size.

AANN Accuracy: As E-AANN is an extension of the AANN, its accuracy is affected by the AANN accuracy and limitations. The accuracy of the AANN depends on the AANN training accuracy[16]. If the training accuracy is high, AANN can map and generalize the data precisely. If it is low, then AANN accuracy is poor.

On the other hand, when the training accuracy is too high, it has opposite effect on the AANN performance. Too high a training accuracy makes the AANN to memorize the data instead of learning the correlation among the variables (for example in section 4, it was shown that the training accuracy of 0.00001 mage the AANN to memorize the data). Therefore the AANN can map the samples of training set perfectly but its generalization capability is poor and it cannot map the samples of test set well.

[16] The issue of training accuracy has been discussed in details in section 3.

E-AANN Step Size: E-AANN locates the faulty sensors through a numerical search. It divides each input range into N points (N is user selectable). Then it calculates the value of the cost function (J) at each point. When the cost function is minimum, the corresponding point is used to locate the faulty sensor. The value of N has influence on E-AANN accuracy. The more N is increased, the smaller step size[17] we have. Smaller step size means higher resolution and better accuracy. On the other hand, E-AANN step size and E-AANN computational time move in opposite direction. When the step size is small, the computational time is high. When the step size is large, the faulty sensor is located fast but the accuracy is poor.

7.2 E-AANN Performance in Noisy Situations

Usually diagnosis mechanisms have poor performance in noisy situations because it is difficult to catch the symptoms of faulty behavior from noisy signals. A good sensor diagnosis mechanism must be able to discriminate between the perturbations coming from the noise and those coming from the sensor error. The E-AANN is capable of detecting and isolating single sensor faults even when the data is noisy. To locate faulty sensors in noisy situations, the difference between the E-AANN input and output must be evaluated in a range of time domain.

For example, consider the chiller model explained in section 5 (an 8-sensor system). The model was used to generate 1000 sets of steady state data. Seven hundred (700) data sets were used to train the network and the remaining 300 data sets were used in testing, test set. Each sensor's data was normalized to have the mean value of zero and variance of one[18]. In this example we induce 2% noise to the data. The AANN embedded in the E-AANN is an 8-11-5-11-8 neural network. Figure 7-1 shows the test data with 2% noise without any induced fault. Figure 7-2 shows the same data but this time the sensor

[17] Step Size = range / N
[18] The detail of the process of data generation can be found in section 4.

#3 has drift error[19]. For better comparison we have zoomed in on the third graph of Figure 7-1 and the corresponding graph of Figure 7-2 as depicted in Figures 7-4.

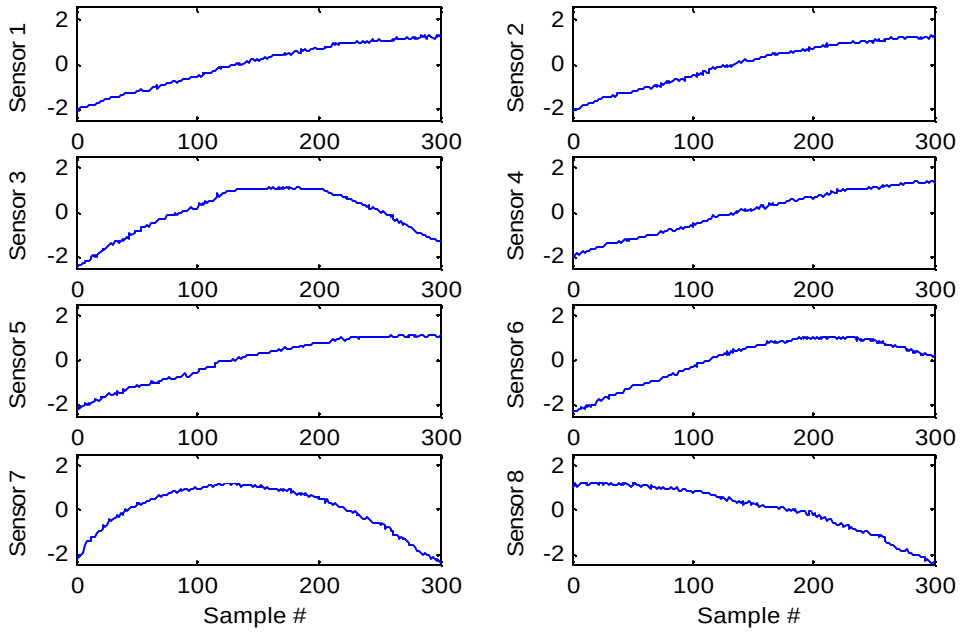

Figure 7-1, Test set with 2% noise

[19]Drift Error Faults a sensor fault in which the sensor error occurs gradually (Figure 7-3). Initially, the error is very small but tit grows slowly with time. There is another kind of sensor error, shift error, in which the sensor error occurs abruptly (Figure 7-3).

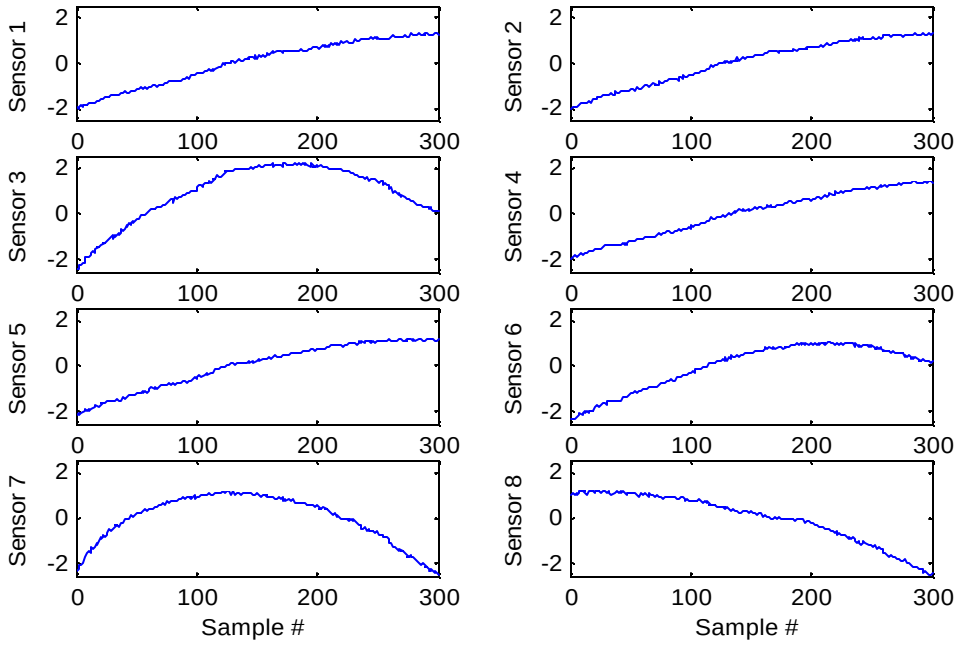

Figure 7-2, Test set with 2% noise, sensor 3 has drift error

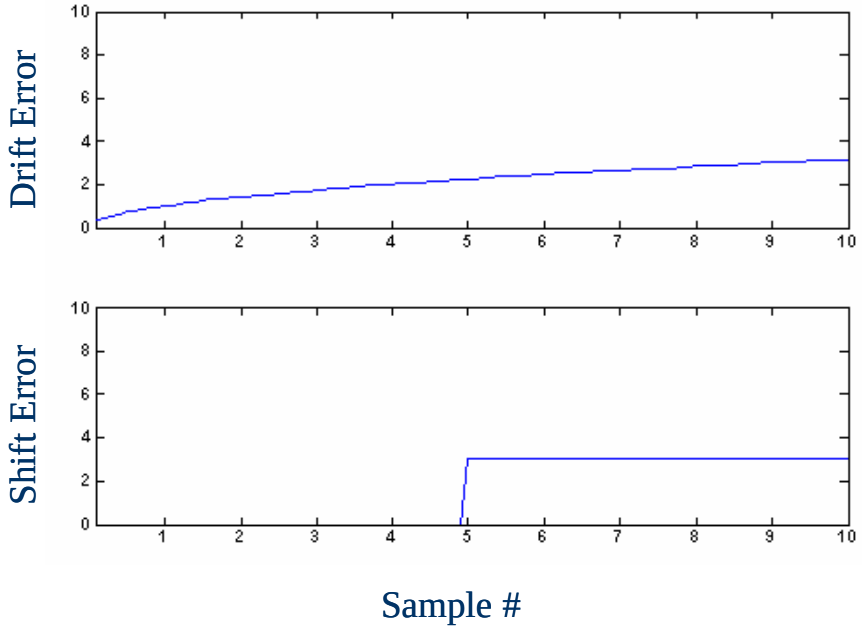

Figure 7-3, Drift error and shift error

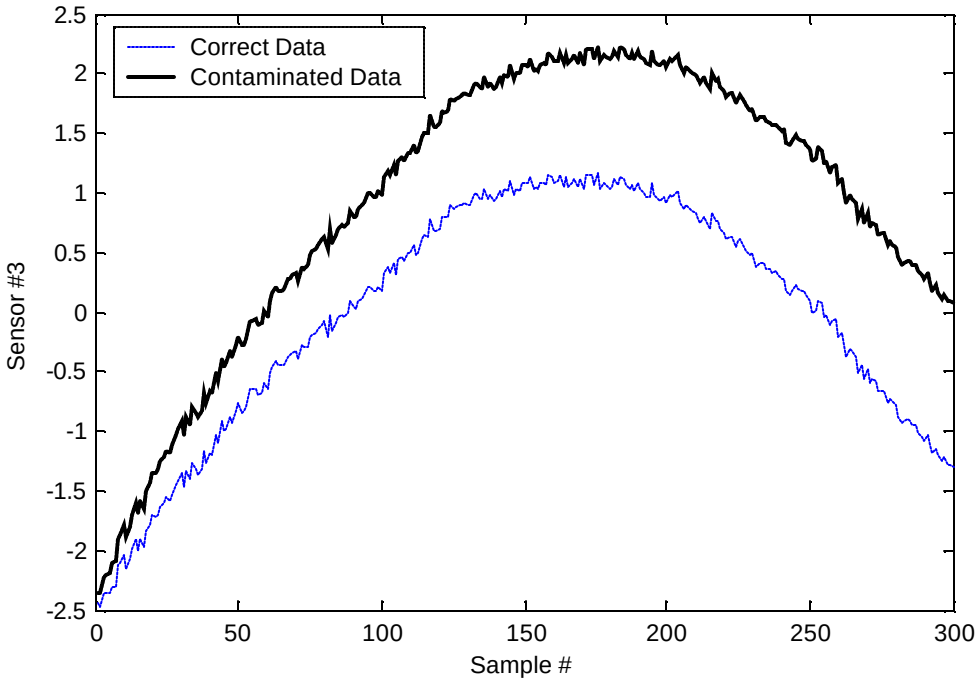

Figure 7-4, Sensor 3 output before and after inducing drift error, noise level is 2%

The E-AANN output is shown in Figure 7-5. The sensor #3 output before contamination, after inducing drift error, and the data reconstructed by E-AANN is shown in Figure 7-6. The difference (error) between the E-AANN input and output is shown in Figure 7-7. The third graph of Figure 7-7 is different from others. All graphs have small perturbations (i.e. noisy shape signals) and zero mean value except for the third one, which has non-zero mean value. We conclude that the perturbations in all graphs except the third one are due to the noise and the corresponding sensors are healthy but the corresponding sensor to the third graph (sensor #3) is faulty. This means that the existing noise in the system does not have any influence on the mean value of the graphs of Figure 7-7. When the mean value is zero, the sensor is healthy. When the mean value is non-zero, the sensor has errors.

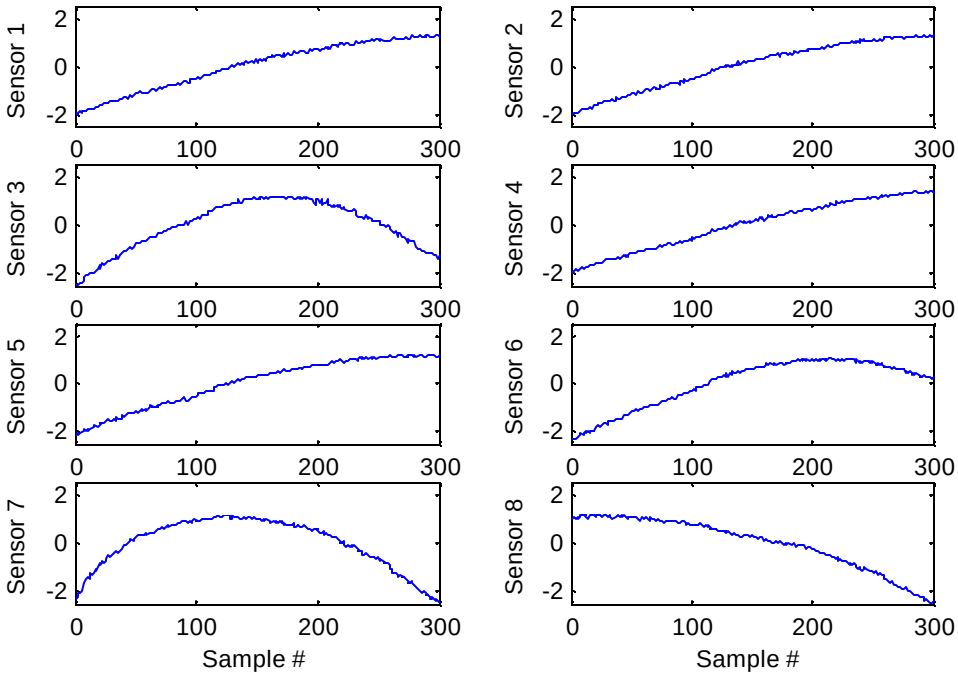

Figure 7-5, E-AANN output; the input data had 2% noise and drift-error

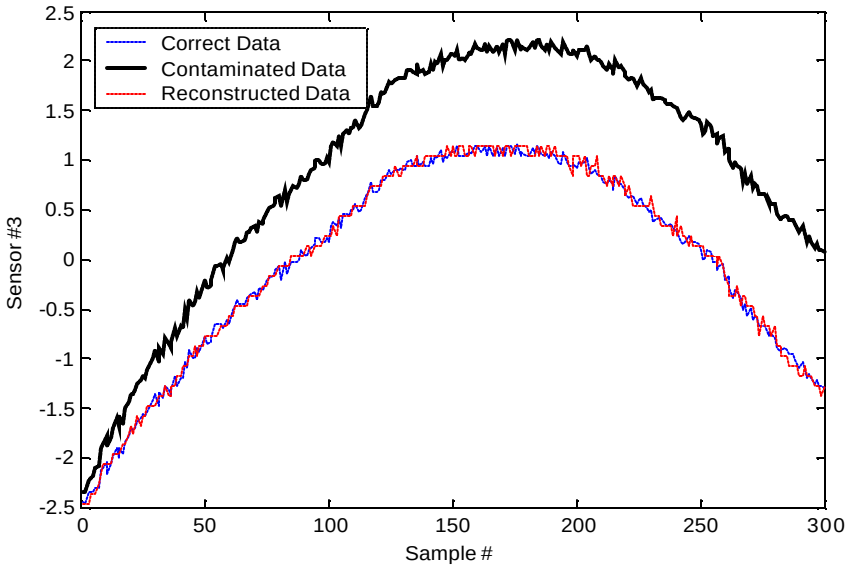

Figure 7-6, Sensor #3 output before contamination, after inducing drift error, and the data reconstructed by E-AANN

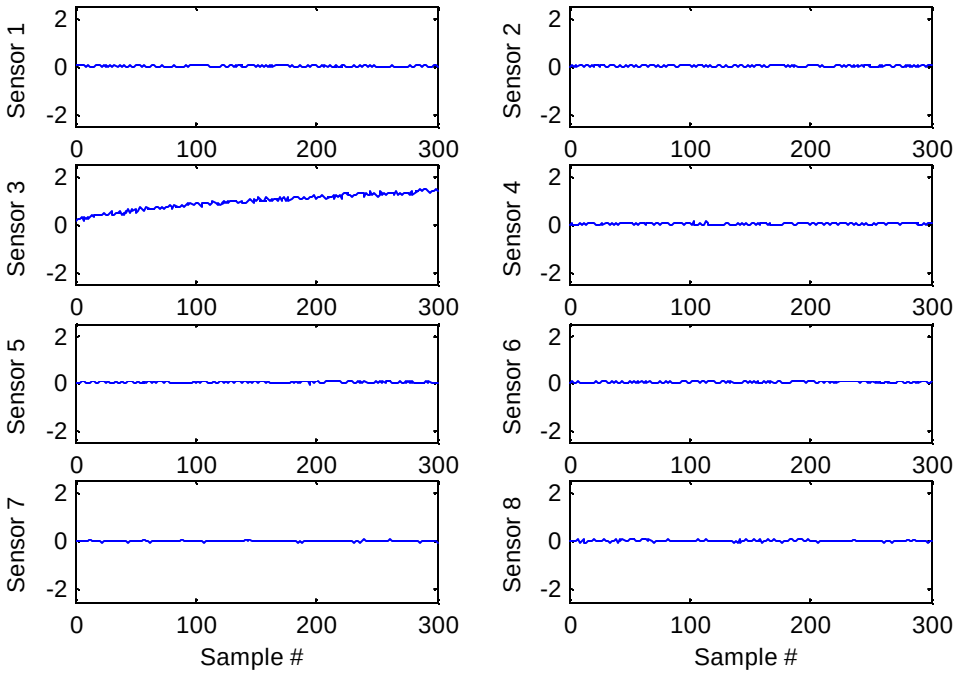

Figure 7-7, The difference between E-AANN input and output. The input data had 2% noise and drift error

Let test the E-AANN with another kind of sensor fault. This time we work with the same data with the same level of noise, but instead of inducing a drift error we induce a shift error (in shift error the sensor error occurs abruptly, Figure 7-3). Figure 7-8 shows the test data with 2% noise and induced fault (sensor #3 has drift error). For better comparison we have zoomed in on the third graph of Figure 7-8 and 7-1 as depicted in Figure 7-9. The induced noise and shift error to sensor #3 is shown in Figure 7-10 ([sensor #3 output + noise + drift error]-[sensor #3 output]).

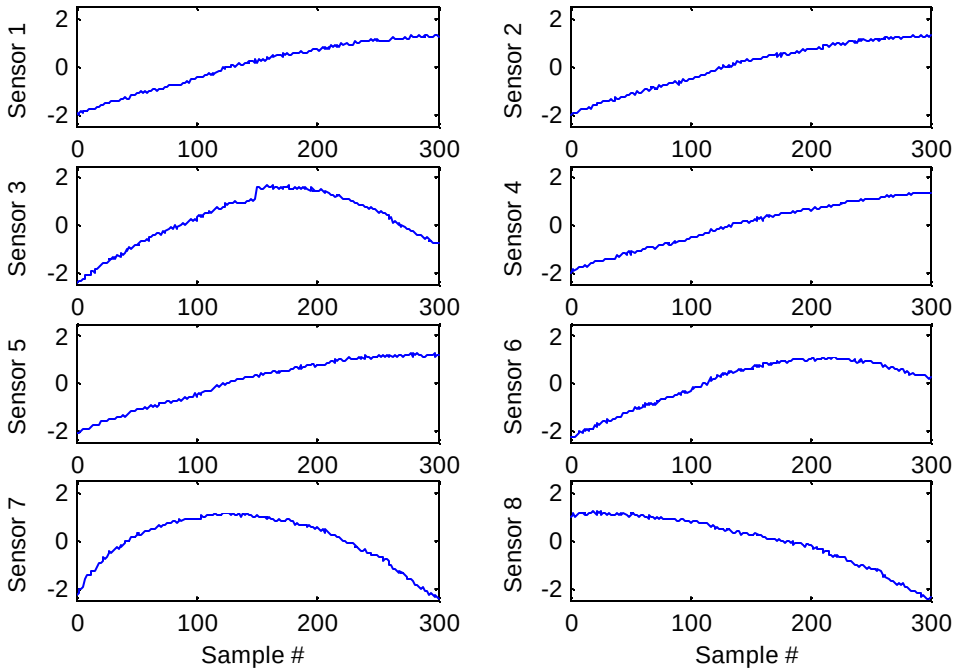

Figure 7-8, Test set with 2% noise, sensor 3 has drift error

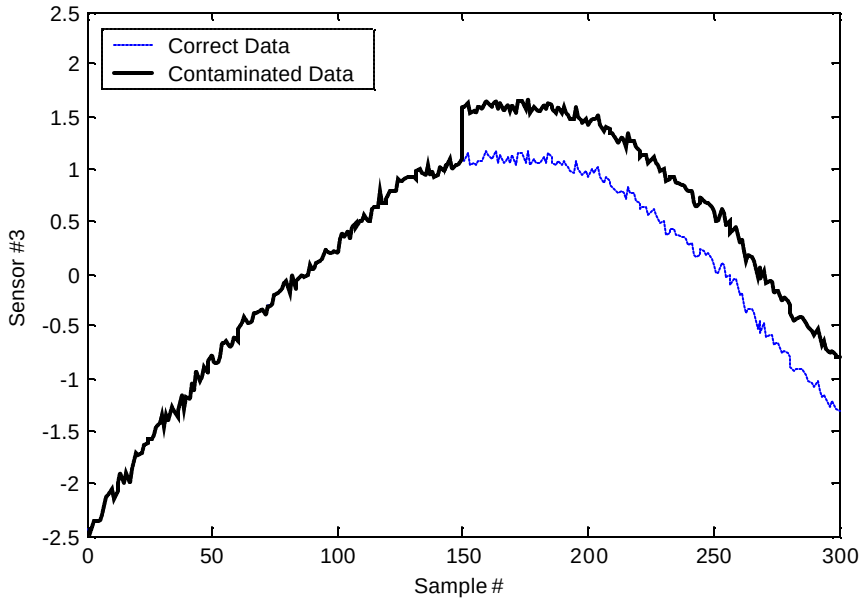

Figure 7-9, Sensor 3 output before contamination and after inducing shift error, noise level 2%

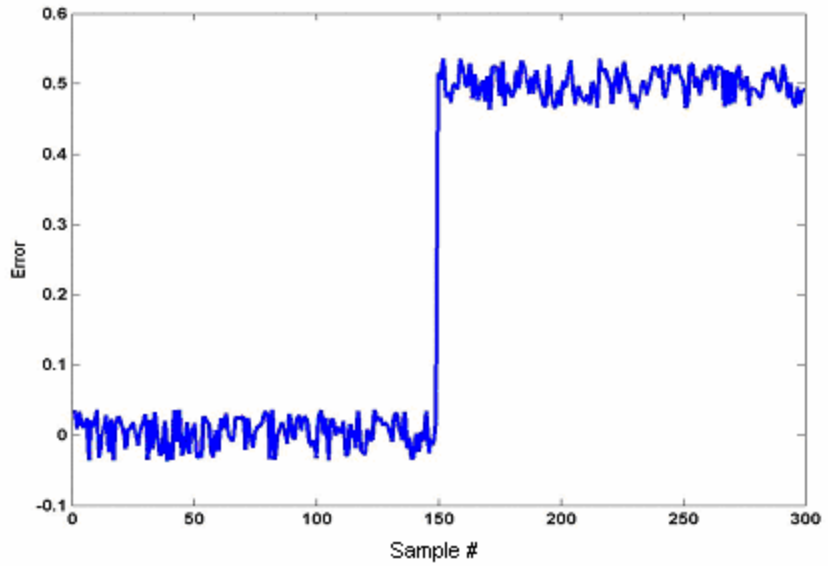

Figure 7-10, The induced 2% noise and shift error to sensor #3
([sensor #3 output + noise + drift error]-[sensor #3 output])

The E-AANN output is shown in Figure 7-11. The sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN is shown in Figure 7-12. The difference between the E-AANN input and output is shown in Figure 7-13. Again we see that the third graph of Figure 7-13 is different from others. All graphs have perturbations with zero mean value but the mean value of the third graph is non-zero. Again this means that the corresponding sensor to the third graph (sensor #3) has a problem.

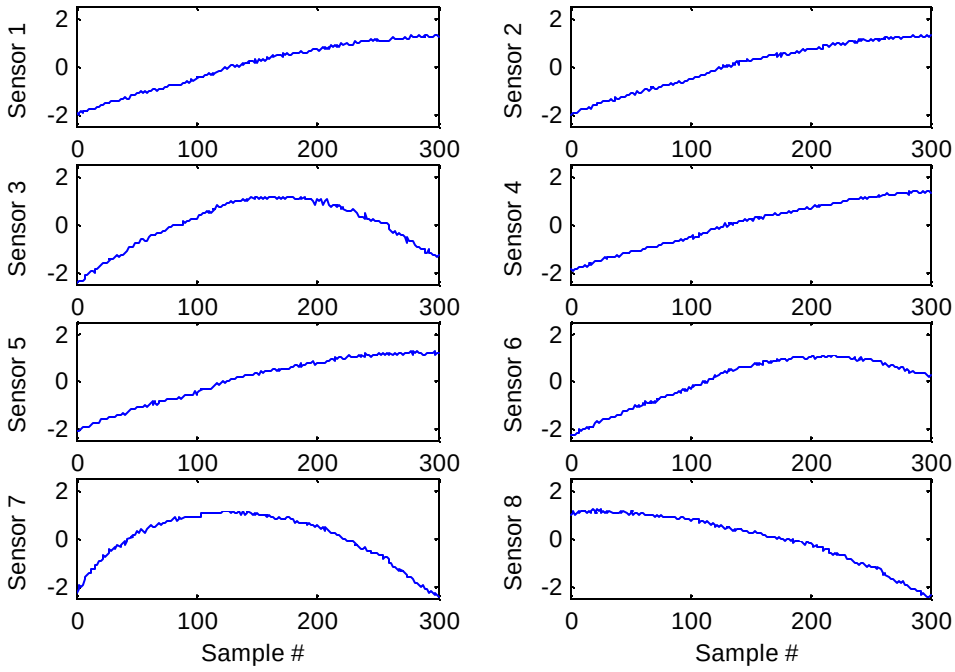

Figure 7-11, E-AANN output; the input data had 2% noise and shift-error

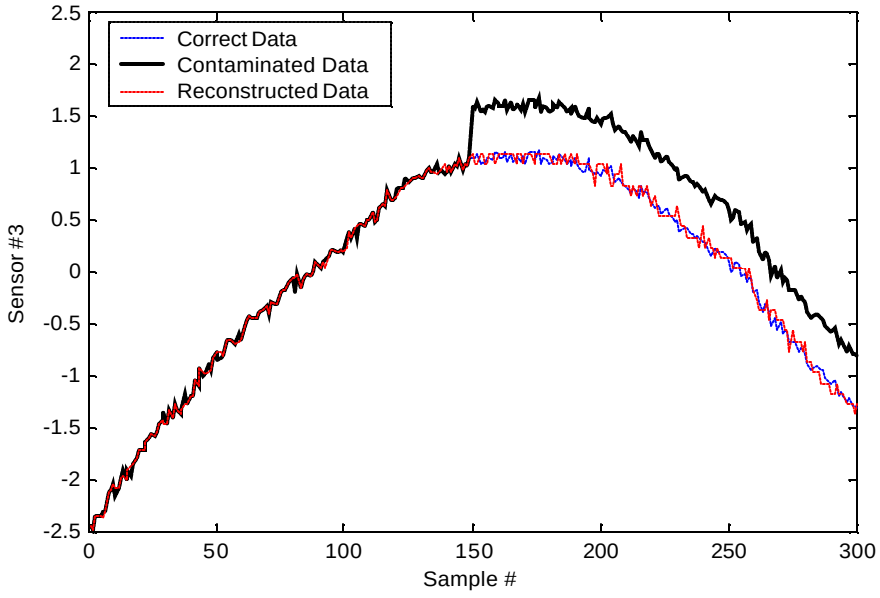

Figure 7-12, Sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN

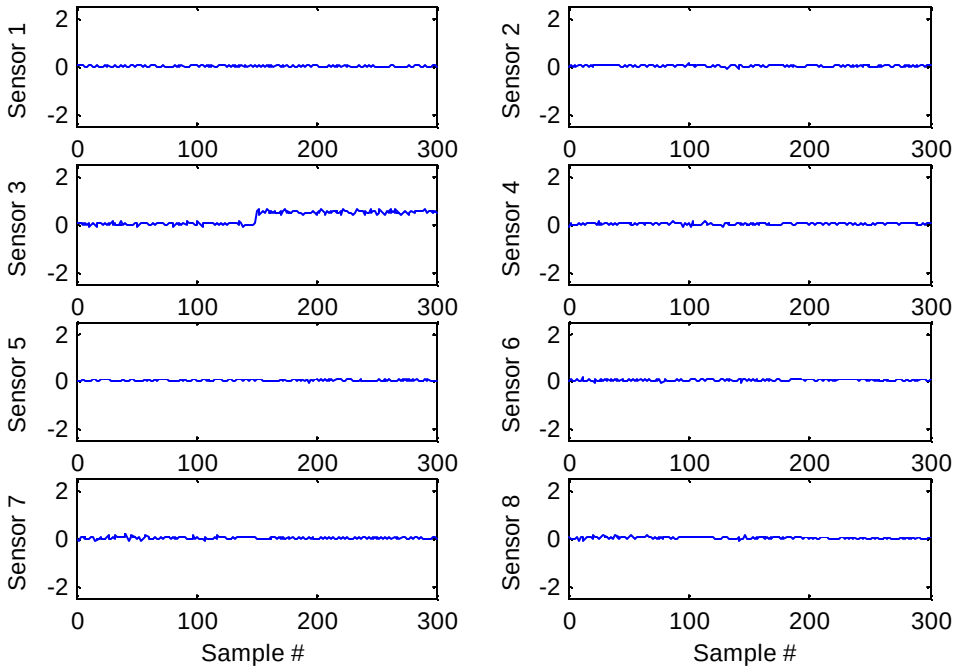

Figure 7-13, The difference between E-AANN input and output. The input data had 2% noise and shift error

In another example, we increase the noise level to see if E-AANN is still capable of locating faulty sensors. Figure 7-14 shows the test data with 5% noise induced and no fault. The data is contaminated by the same sensor faults (drift error and shift error).

I) Drift error: Figure 7-15 shows the contaminated data with drift error (sensor #3 has drift error). For better comparison we have zoomed in on the third graph of Figure 7-15 and the corresponding graph of Figure 7-1 as depicted in Figure 7-16. The induced noise and drift error to sensor #3 is shown in Figure 7-17 ([sensor #3 output + noise + drift error]-[sensor #3 output]). The E-AANN output is shown in Figure 7-18. The sensor #3 output before contamination, after inducing drift error, and the data reconstructed by E-AANN is shown in Figure 7-19. The difference between the E-AANN input and output is shown in Figure 7-20. Again we see the third graph of Figure

7-20 has a different mean value comparing with others. The only difference comparing with previous example is that in Figure 7-20 graphs seem to have more perturbations.

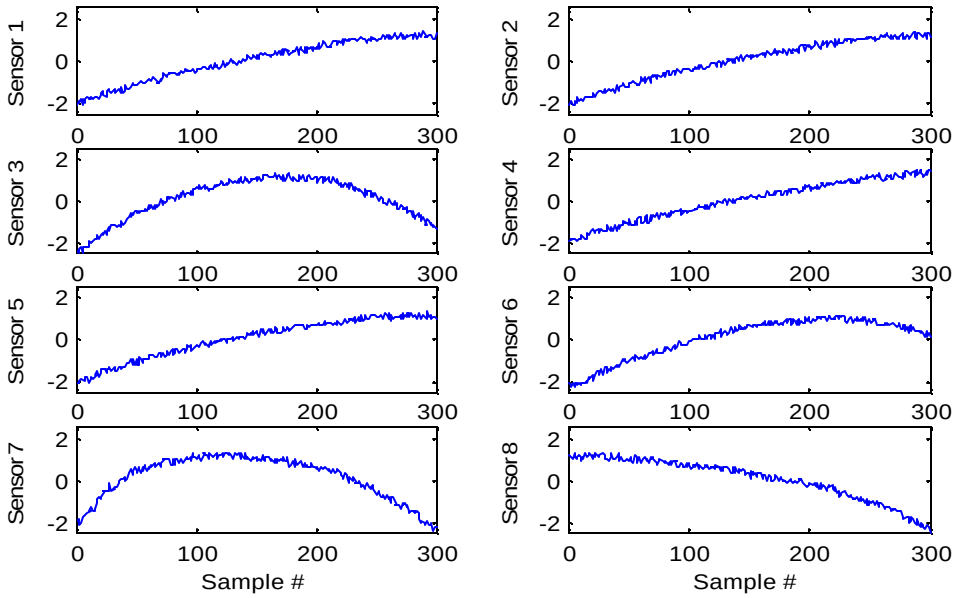

Figure 7-14, Test data with 5% noise

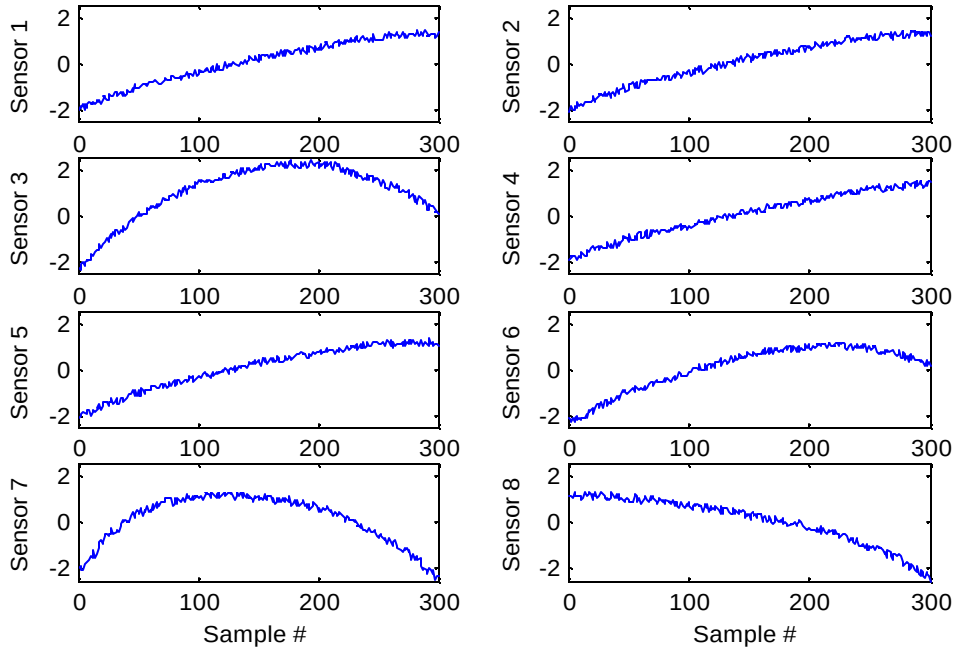

Figure 7-15, Test set with 5% noise, sensor 3 has drift error

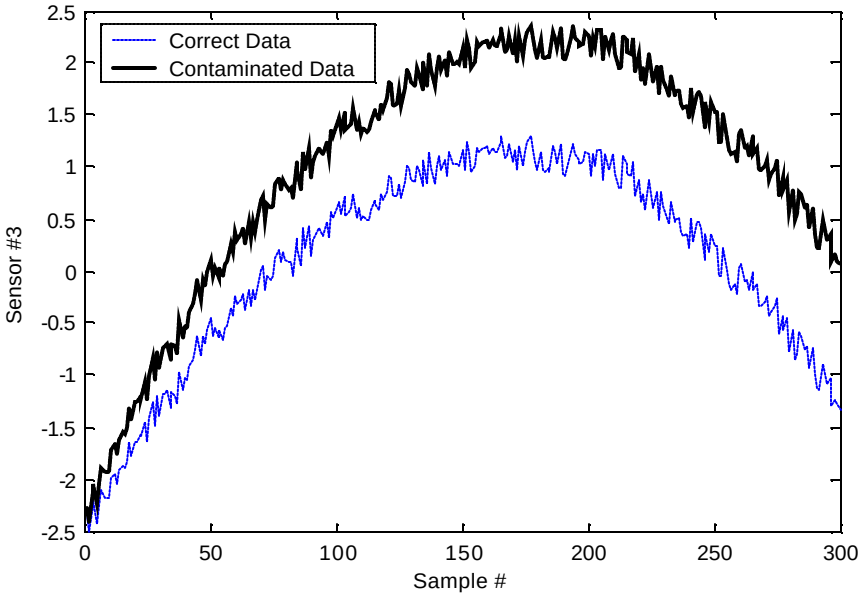

Figure 7-16, Sensor 3 output before contamination and after inducing drift error, noise level is 5%

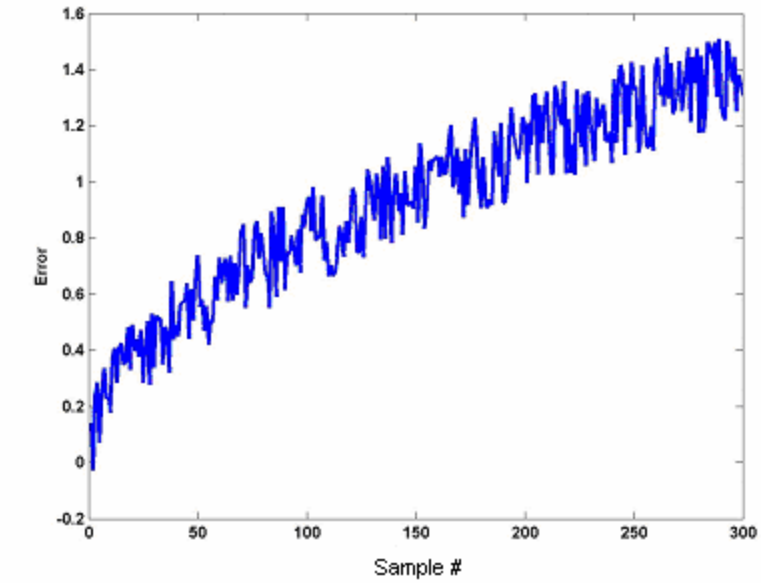

Figure 7-17, The induced 5% noise and drift error to sensor #3

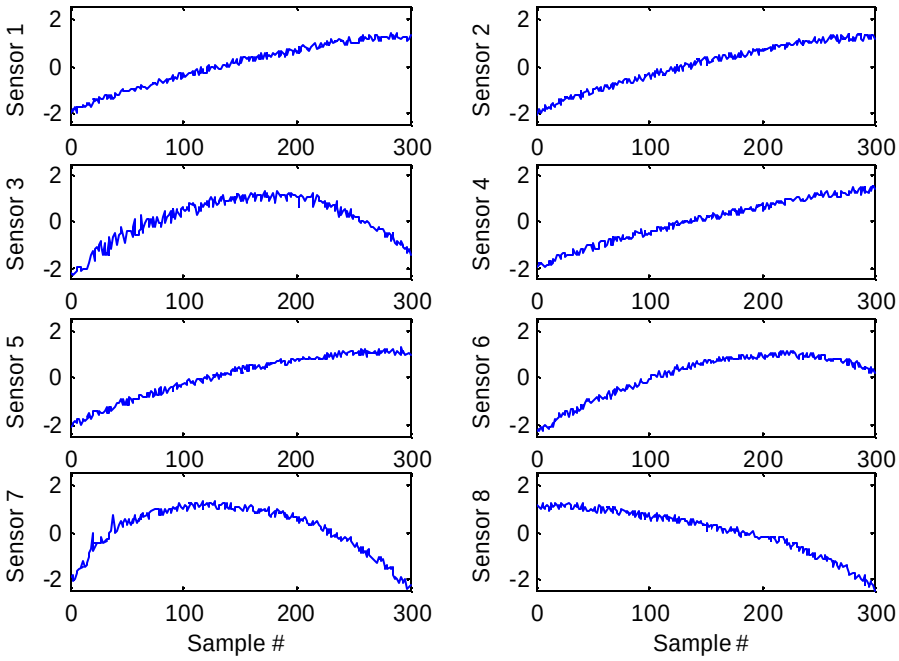

Figure 7-18, E-AANN output; the input data had 5% noise and drift-error

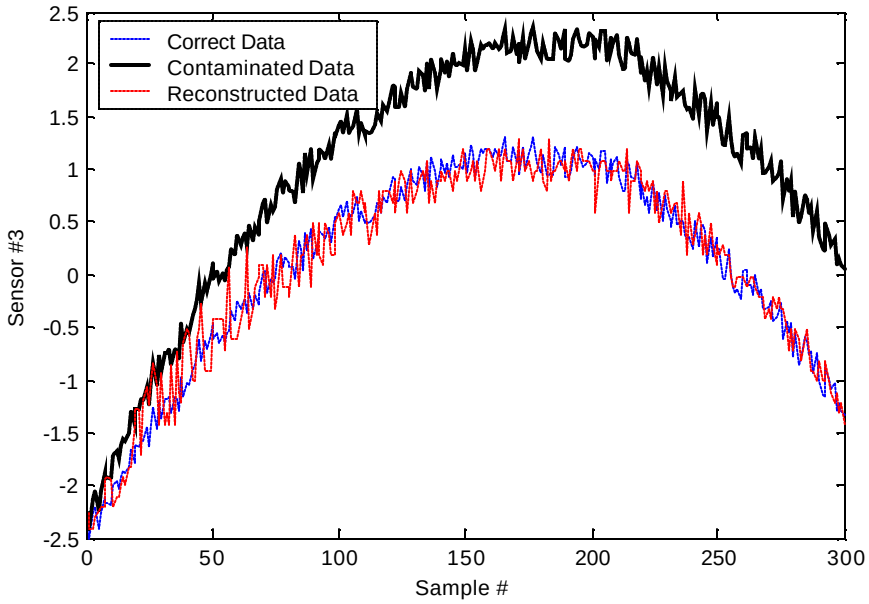

Figure 7-19, Sensor #3 before contamination, after inducing drift error, and the data reconstructed by E-AANN, noise level is 5%

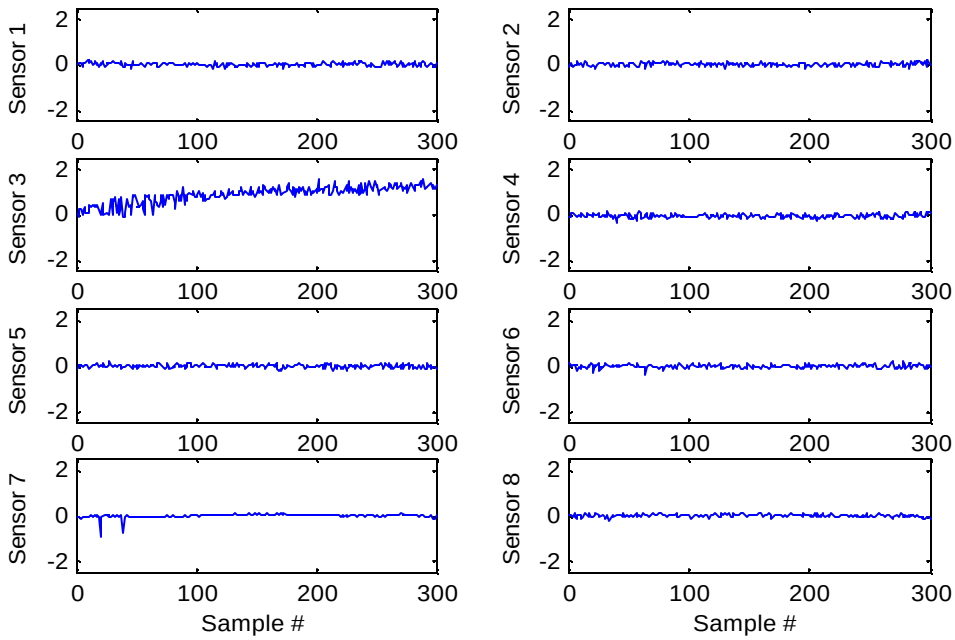

Figure 7-20, The difference between E-AANN input and output. The input data had 5% noise and drift error

II) Shift error: Figure 7-21 shows the contaminated data (sensor #3 has shift error). we have zoomed in on the third graph of Figure 7-21 and the corresponding graph of Figure 7-1 as depicted in Figure 7-22. The induced noise and drift error to sensor #3 is shown in Figure 7-23. The E-AANN output is shown in Figure 7-24. The sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN is shown in Figure 7-25. The difference between E-AANN input/output is shown in Figure 7-26. Again we see the third graph of Figure 7-26 has a different mean value comparing with others.

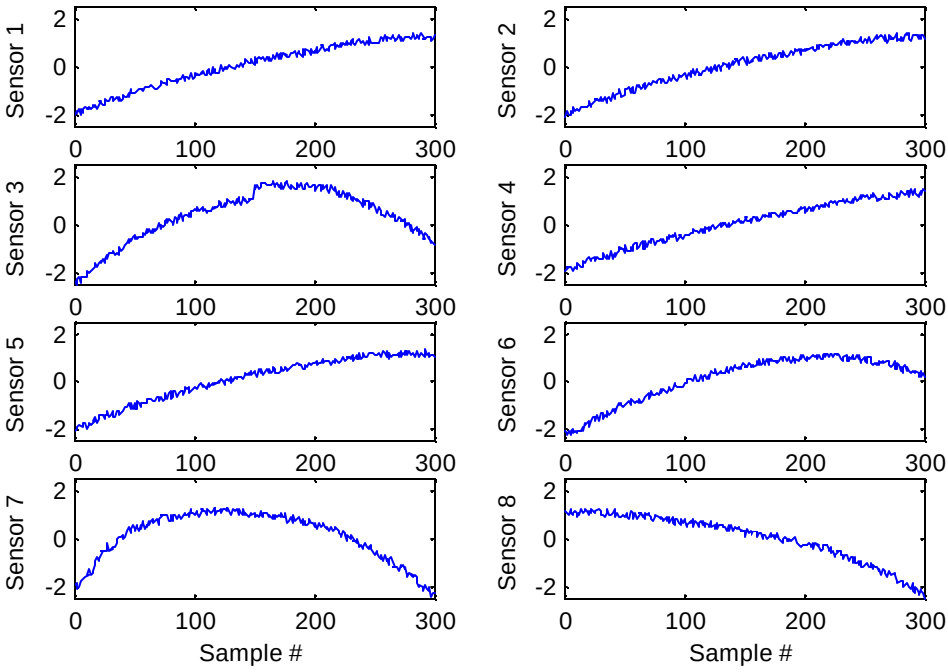

Figure 7-21, Test set with 5% noise, sensor 3 has shift error

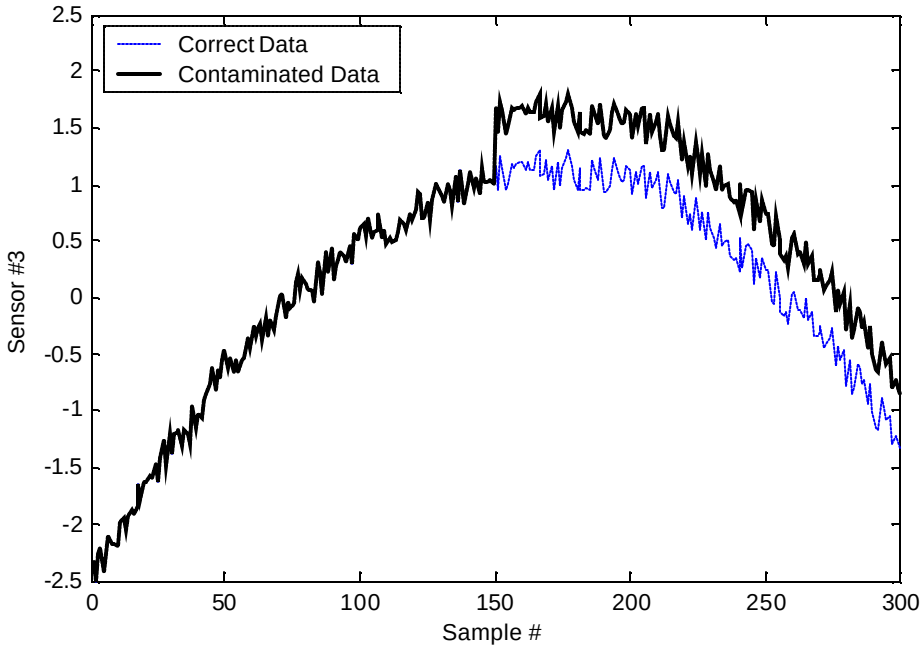

Figure 7-22, Sensor 3 outputs before contamination and after inducing shift error, noise level is 5%

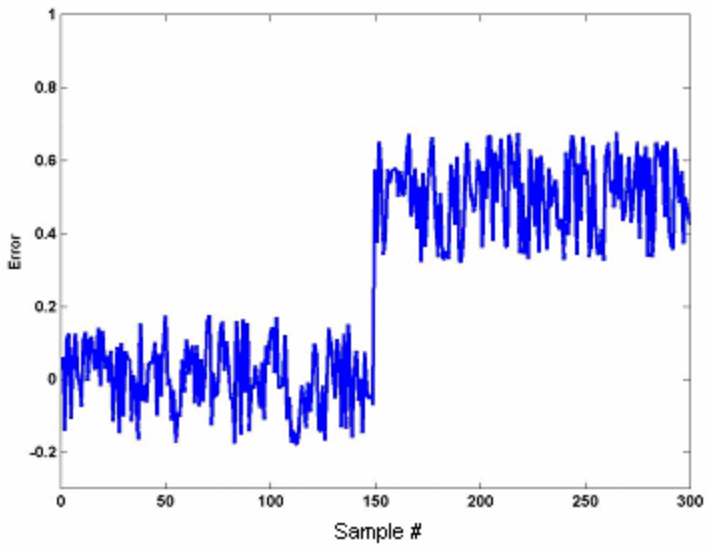

Figure 7-23, The induced 5% noise and shift error to sensor #3

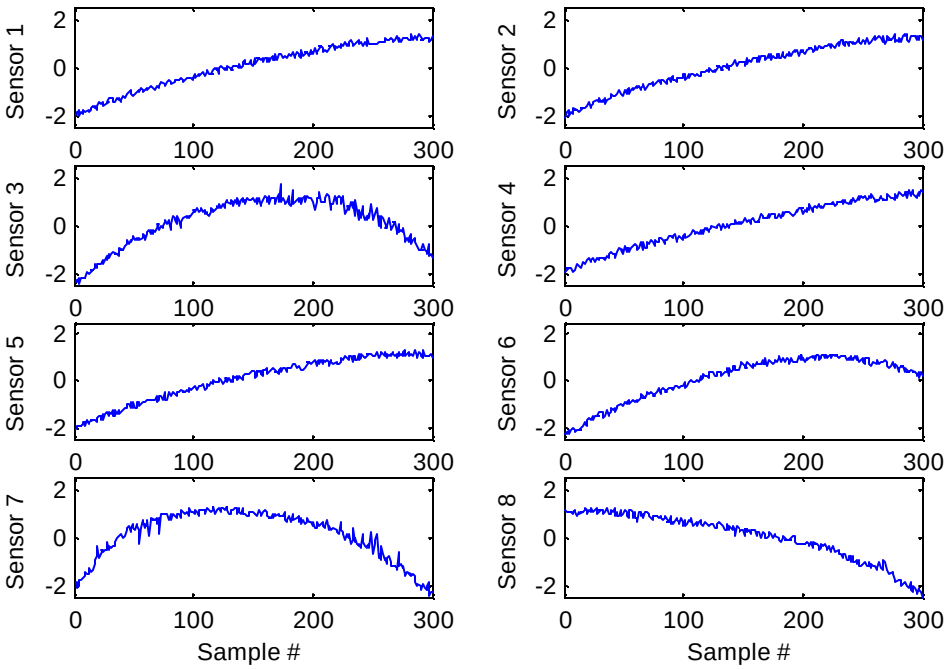

Figure 7-24, E-AANN output; the input data had 5% noise and shift-error

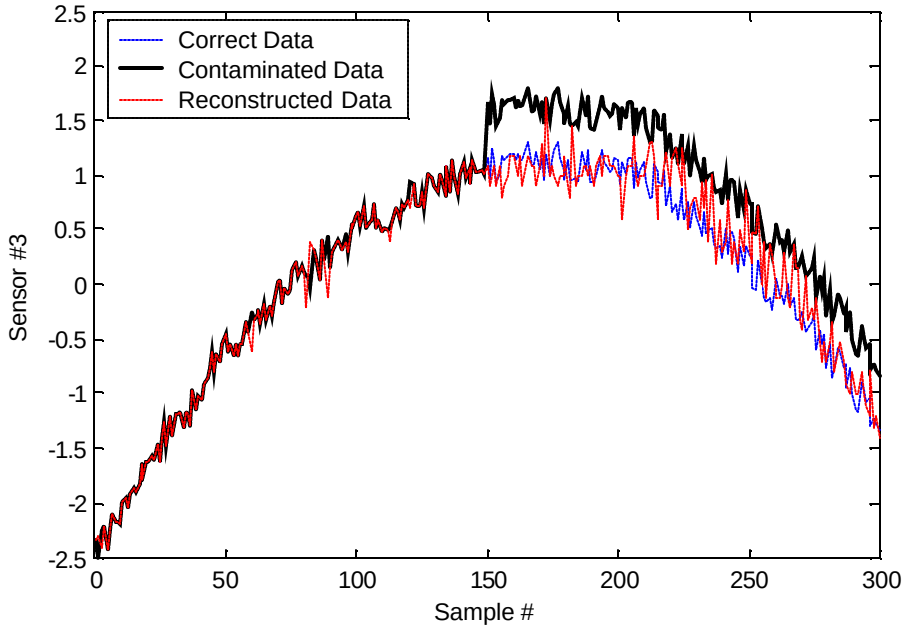

Figure 7-25, Sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN, noise level is 5%

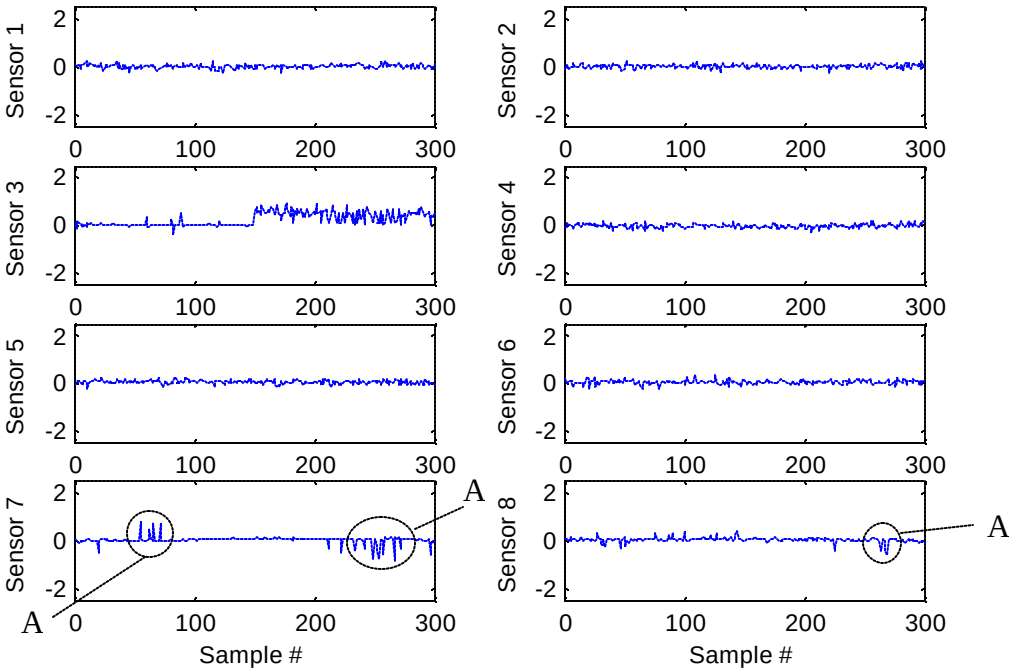

Figure 7-26, The difference between the E-AANN input and output. The input data had 5% noise and shift error

As it was mentioned, when a data is noisy, it has a level of uncertainty. This uncertainty might confuse E-AANN sometimes. Due to the uncertainty, E-AANN might minimize the cost function by varying a non-faulty sensor. This is the reason of the behaviors named as A in the Figure 7-26. In these cases E-AANN has located the faulty sensor wrong. As the level of the uncertainty depends on the noise level, we might expect more behaviors like A in higher noisy situations. If you see Figure 7-7 which shows the difference between E-AANN input and output when the noise level is 10%, you see that there exist more behaviors like A.

7.3 Maximum Level of Noise Tolerated by E-AANN

It is important to see what would be the E-AANN performance when the level of noise is too high. Is there any distinct boundary as the maximum level of noise for the E-AANN? Since the E-AANN is an extension of the AANN, its performance is affected by the AANN limitations in high noisy situations. When noise level is high, AANN reconstructs the input data with degraded accuracy. Therefore we must expect degraded performance from the E-AANN in high noisy situations (for example in chiller model, it will be shown that when the noise level is 10% to 20% the E-AANN performance degrades).

Our results show that there is no distinct boundary as the maximum noise level for the E-AANN. As the noise level increases, the E-AANN accuracy decreases, so the E-AANN may not catch small sensor faults when noise level is high.

In the previous example E-AANN was used to catch sensor faults when the data had 5% noise. The noise level was increased to 10% (same data) to see if E-AANN can catch the same drift error and sensor error or not. Figure 7-27 shows the test data with 10% and no induced fault. Figure 7-28 shows the same data but sensor #3 has a drift error. For better comparison we have zoomed in on the third graph of Figure 7-27 and the

corresponding graph of Figure 7-28 as depicted in Figure 7-29. The induced noise and drift error to sensor #3 is shown in Figure 7-30. The E-AANN output is shown in Figure 7-31 and the zoomed of the third graph is shown in Figure 7-32. The difference between the E-AANN input and output is shown Figure 7-33.

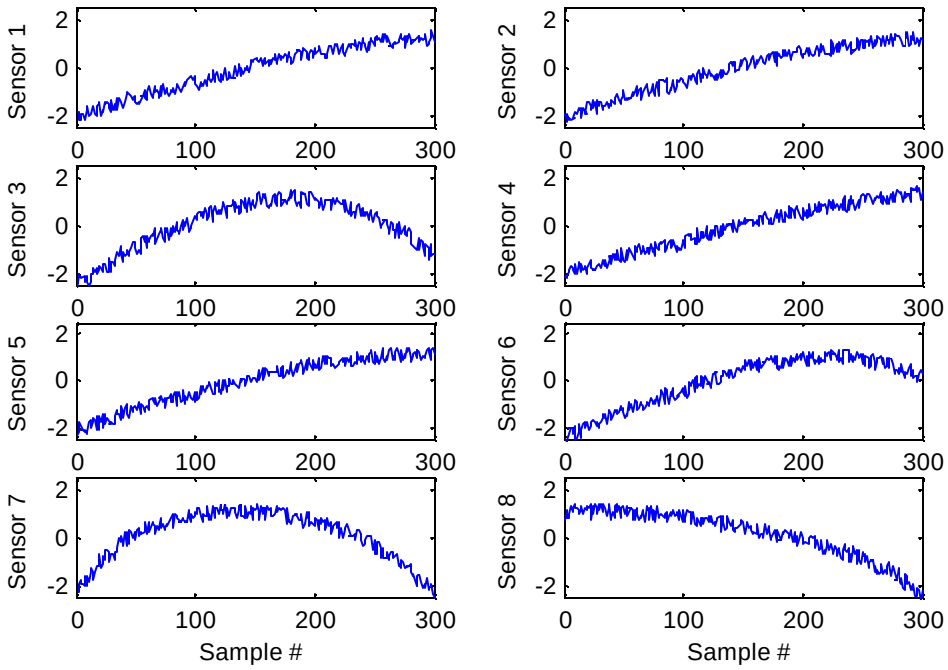

Figure 7-27, Test data with 10% noise

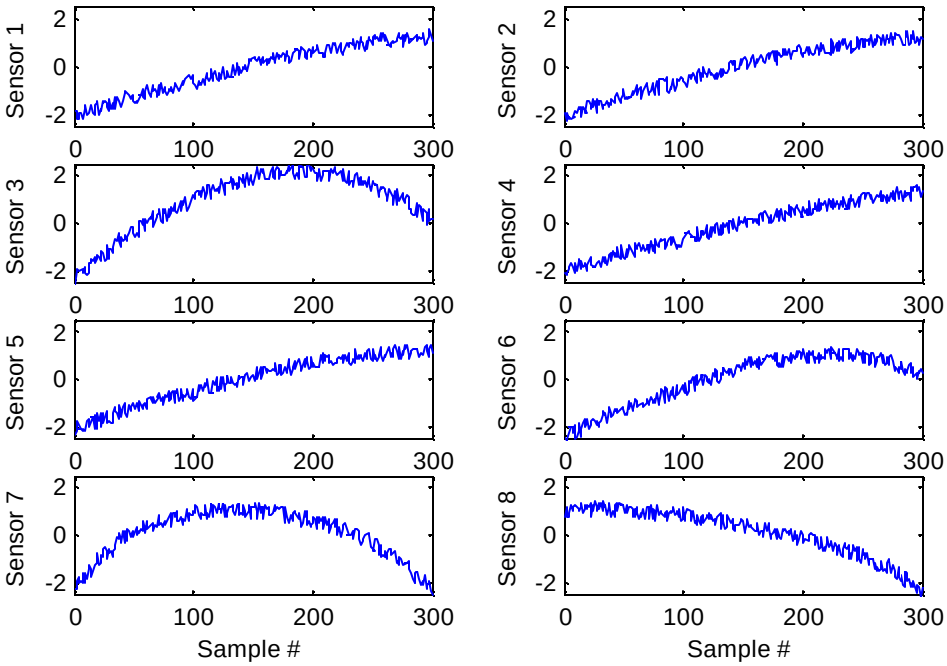

Figure 7-28, Test set with 10% noise, sensor 3 has drift error

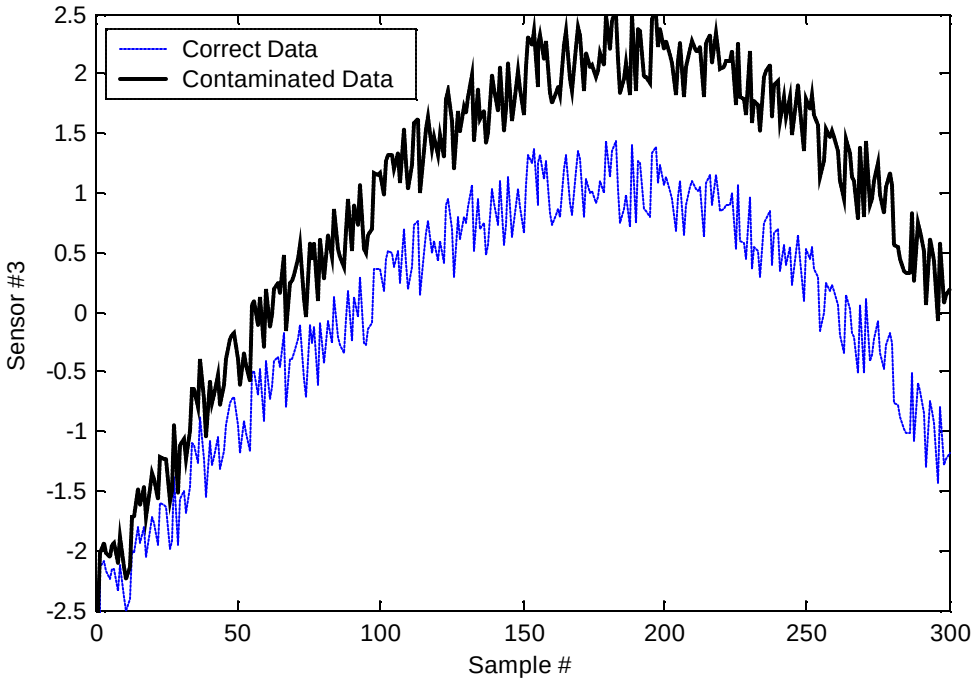

Figure 7-29, Sensor 3 output before contamination and after inducing drift error, noise level 10%

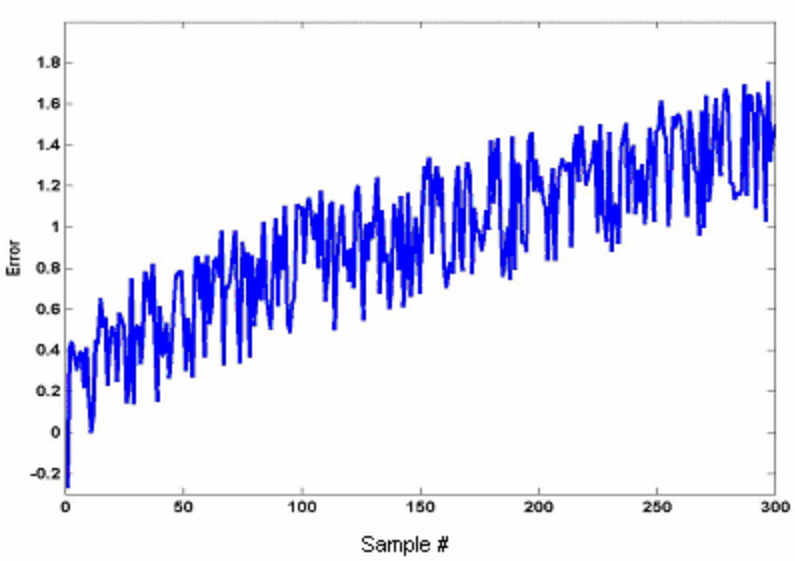

Figure 7-30, The induced 10% noise and drift error to sensor #3

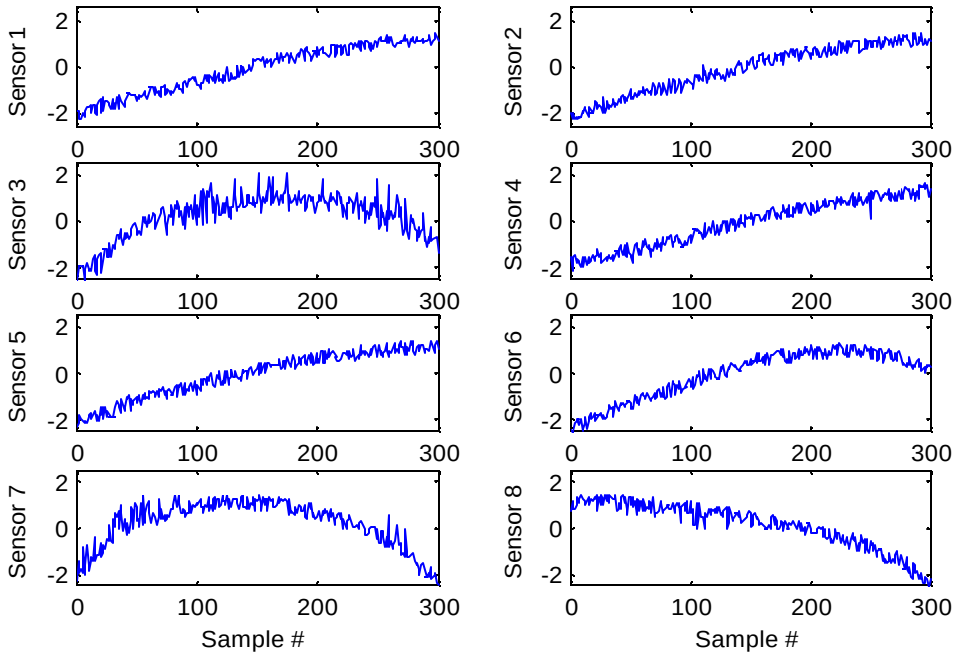

Figure 7-31, E-AANN output; the input data had 10% noise and drift-error

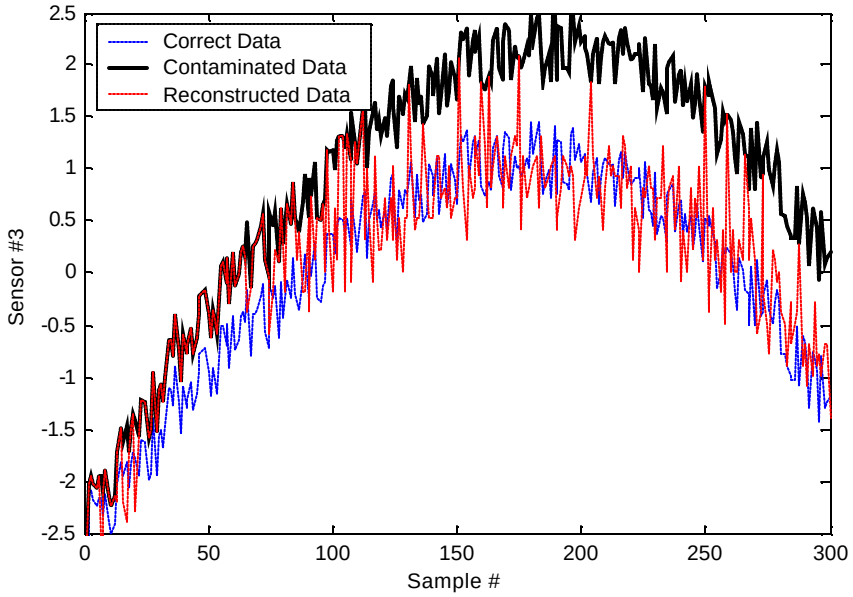

Figure 7-32, Sensor #3 outputs before contamination, after inducing drift error, and the data reconstructed by E-AANN, noise level is 10%

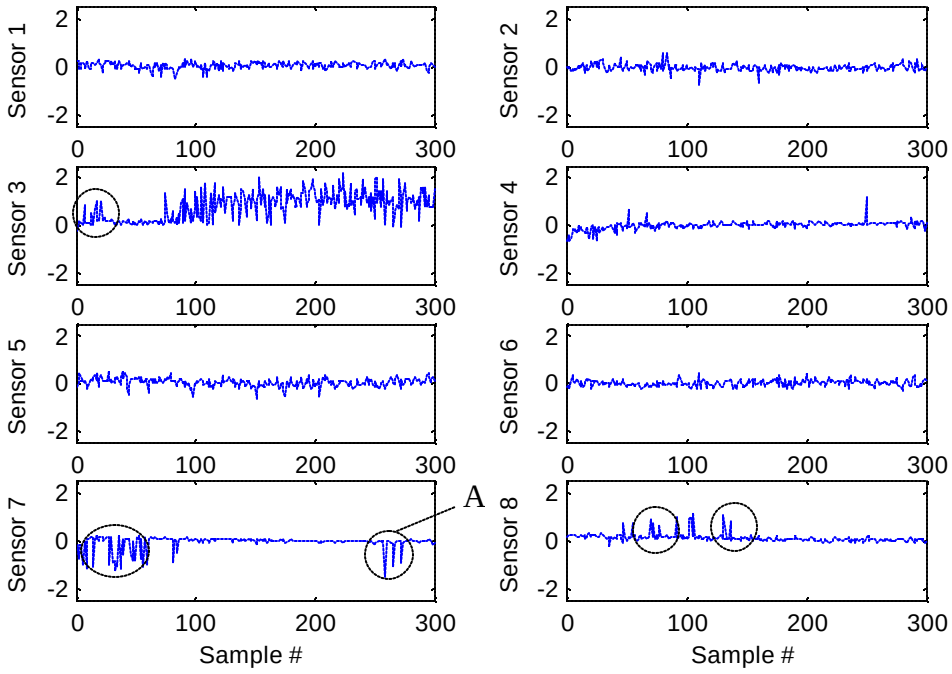

Figure 7-33, The difference between E-AANN input and output. The input data had 10% noise and drift error

Again as we see in Figure 7-33, all graphs have perturbations with zero mean values except the third one. Therefore we conclude the same. Now let us see E-AANN response to the shift error. Figure 7-34 shows the test data while sensor #3 has shift error. We have zoomed in on the third graph of Figure 7-34 and the corresponding graph of Figure 7-27 as depicted in Figure 7-35. The induced noise and drift error to sensor #3 is shown in Figure 7-36. The E-AANN output is shown in Figure 7-37 and the zoomed of the third graph is shown in Figure 7-38.

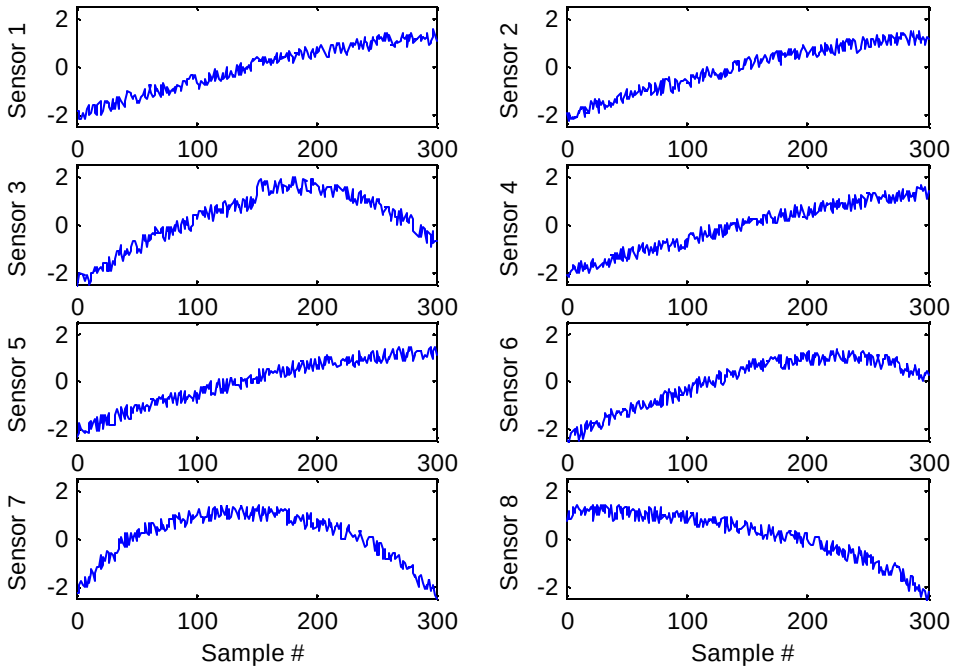

Figure 7-34, Test set with 10% noise, sensor 3 has shift error

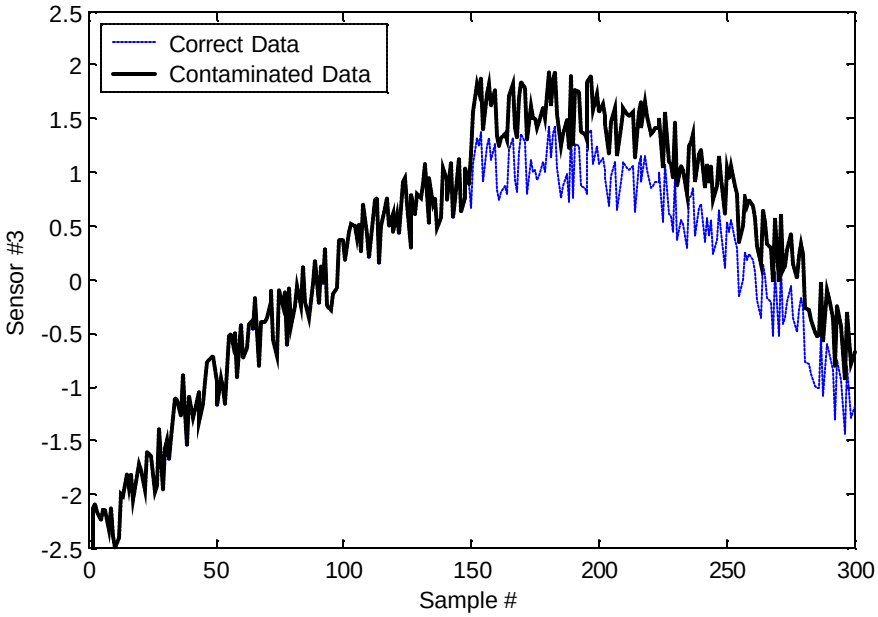

Figure 7-35, Sensor 3 output before contamination and after inducing shift error, noise level is 10%

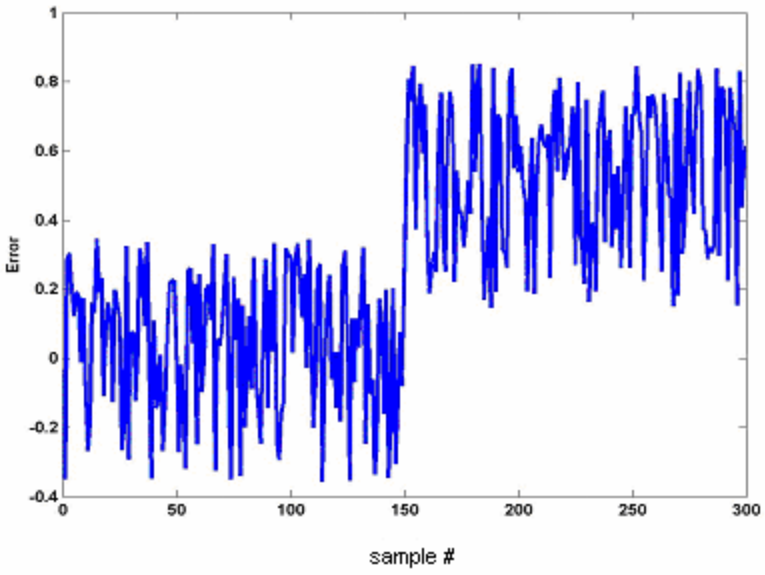

Figure 7-36, The induced 10% noise and shift error to sensor #3

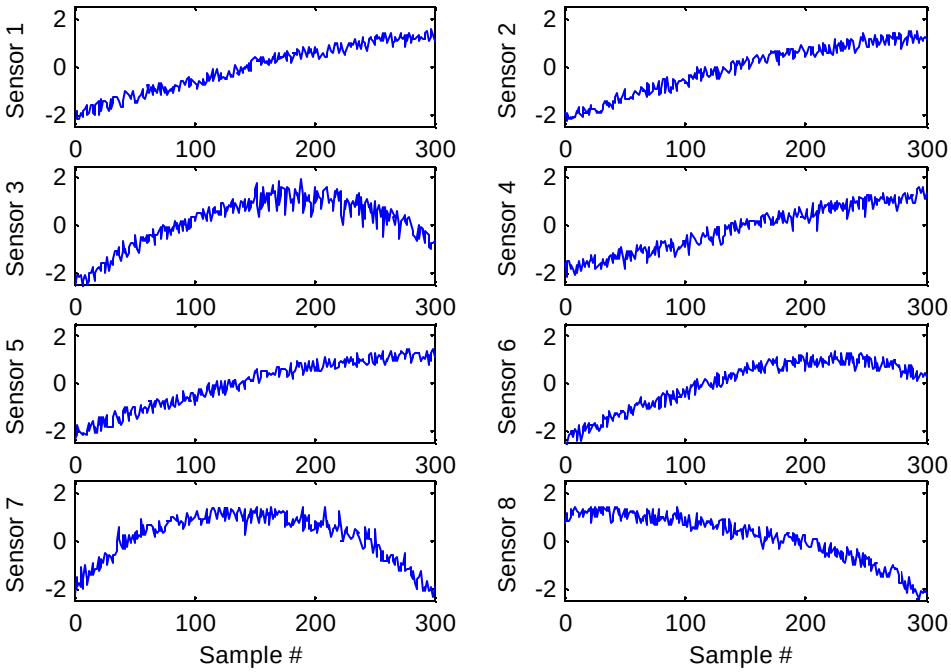

Figure 7-37, E-AANN output; the input data had 10% noise and shift-error

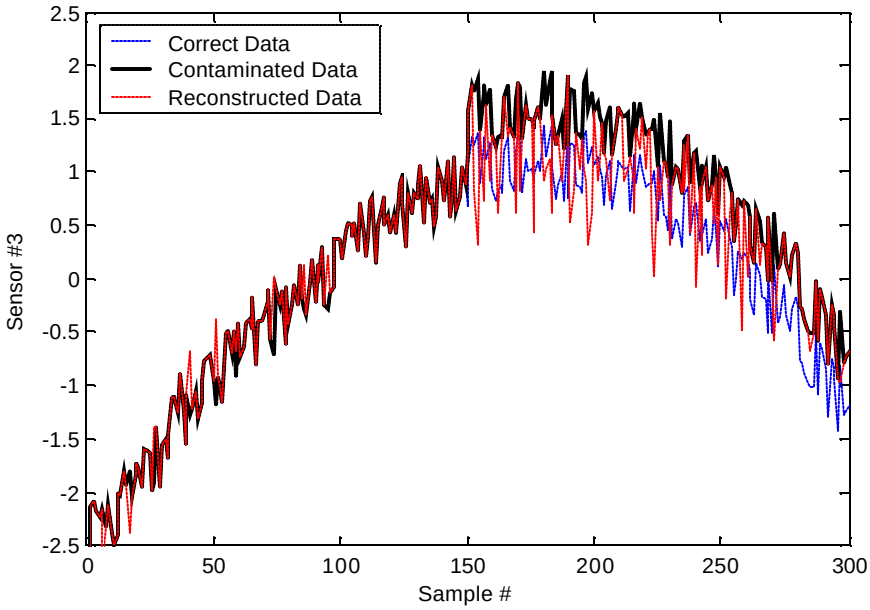

Figure 7-38, Sensor #3 output before contamination, after inducing shift error, and the data reconstructed by E-AANN, noise level is 10%

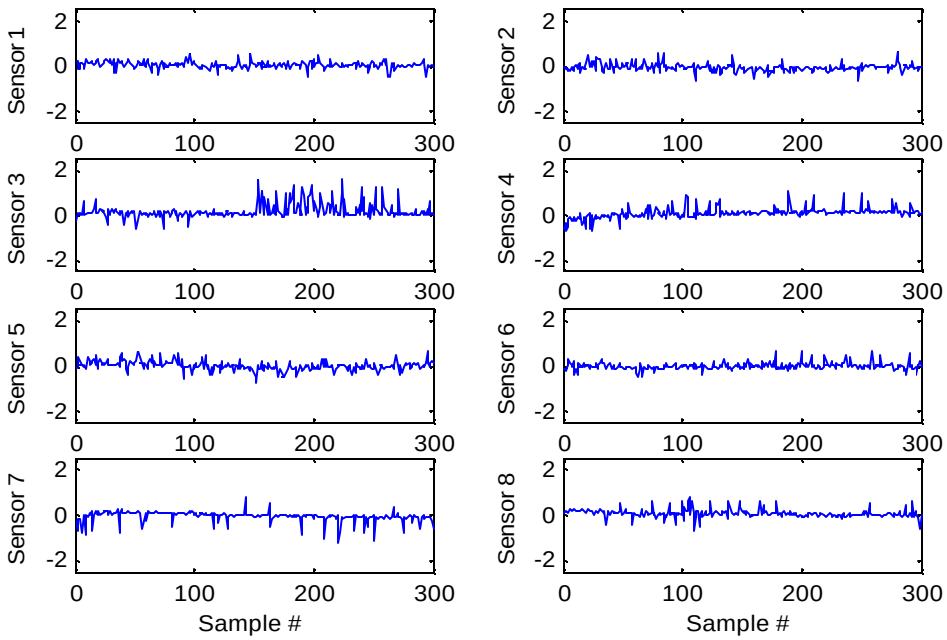

Figure 7-39, The difference between E-AANN input and output. The input data had 10% noise and shift error

The difference between the E-AANN input and output is shown Figure 7-39. As the third sensor was contaminated by the shift error, we expect the third graph of Figure 7-39 to be similar to the third graph of Figure 7-26. However, as you see, there is no shift in the mean value of the third graph signal. This is due to the high noise level existing in the system. The graphs in Figure 7-39 are dominated by noise and the symptoms of faulty response are hidden behind the noise. If the induced shift error were bigger, we would see a shift in the mean value of the third graph of Figure 7-44. The use of Filter could remove the noise and show the shift in the delta graph (Future work).

8. APPLICATION OF E-AANN IN SENSOR DIAGNOSTICS IN DYNAMIC SYSTEMS

8.1 Introduction

In previous sections it was shown that E-AANN was capable of locating different types of sensor errors in different situations. We showed how E-AANN could catch faulty sensors in both noise free and noisy situations. We discussed the issue of maximum level of the noise tolerated by the E-AANN. However all the systems used in previous sections were static systems. Now the question is what if the system is dynamic? Is E-AANN still capable of locating faulty sensors in dynamic situations or not?

In static systems there is no delay between the input change and its effect on the output. When the system input changes, the output responds to that change immediately. There is no transition period between the input change and the output response. In dynamic systems, the output does not respond to the input change immediately. There is a delay between the input change and the appearance of its effect on the output

Our results show that in the case of dynamic systems, E-AANN is still capable of locating faulty sensors if the training data has some specifications. We start with the evaluation of E-AANN behavior in dynamic situations and non-faulty conditions. Then we will do through the E-AANN capabilities of locating faulty sensors in dynamic situations.

8.2 E-AANN Response to Dynamic Systems in Non-faulty Situations

As it was mentioned, in dynamic systems the system output does not respond to the input change immediately. In dynamic systems we have two types of data. The data that is measured when the system is in transition period and the data measured when the system has passed the transition period. We name the first one as Dynamic Data and the second one as Static Data.

In order to use the E-AANN for sensor diagnostics in dynamic systems, the AANN embedded in the E-AANN should be trained based on Static Data. For test data, there is no limitation.

Example 1: Consider the chiller model explained in section 3. That model simulates a static system with three inputs and five outputs (Figure 8-1).

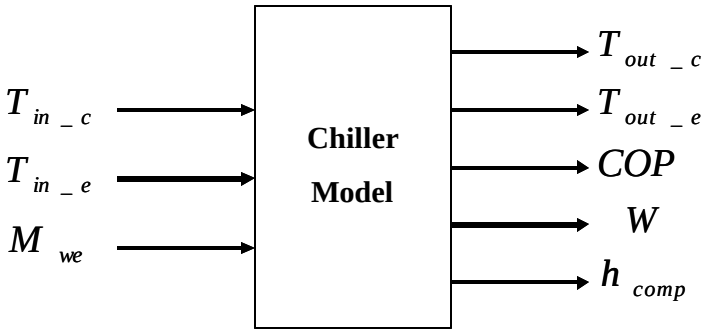

Figure 8-1, Chiller model inputs/outputs

In this section the chiller model is modified to mimic dynamic systems. As a simple dynamic system, in this example the model has been modified such that the first output (Tout_c) has a delay with four samples (Figure 8-2). Therefore the first output responds to the input changes four samples later[20].

[20]In a dynamic system, the system output responds to the input change either with delay or gradually. In this example the model has been modified such that the first output responds to the input change with delay not gradually.

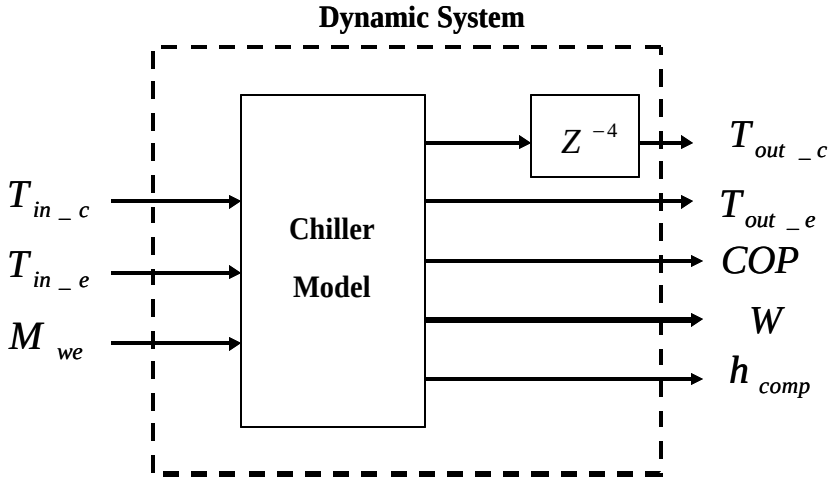

Figure 8-2, Static model with delay on one sensor to mimic dynamic system, Z^{-4} means a delay of 4 samples

Figure 8-3 shows the E-AANN diagram. You see that the first output of the chiller model corresponds to the fourth input/output of the E-AANN. The training data, which is made up of static data, is shown in Figure 8-4. The test data contains both Dynamic Data and Static Data. It has no induced sensor error. Figure 8-5 shows the test data. It has been prepared by varying the system input every 10 samples[21] (Step = 10).

[21] In this example the system delay is less than the step size of the data (system delay is 4 and step size is 10). In next examples we will go through the cases in which the system delay is bigger than step size.

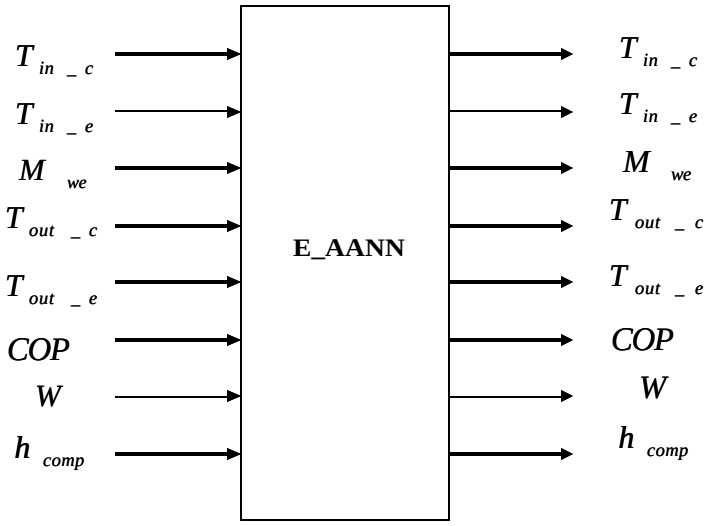

Figure 8-3, E-AANN inputs and outputs

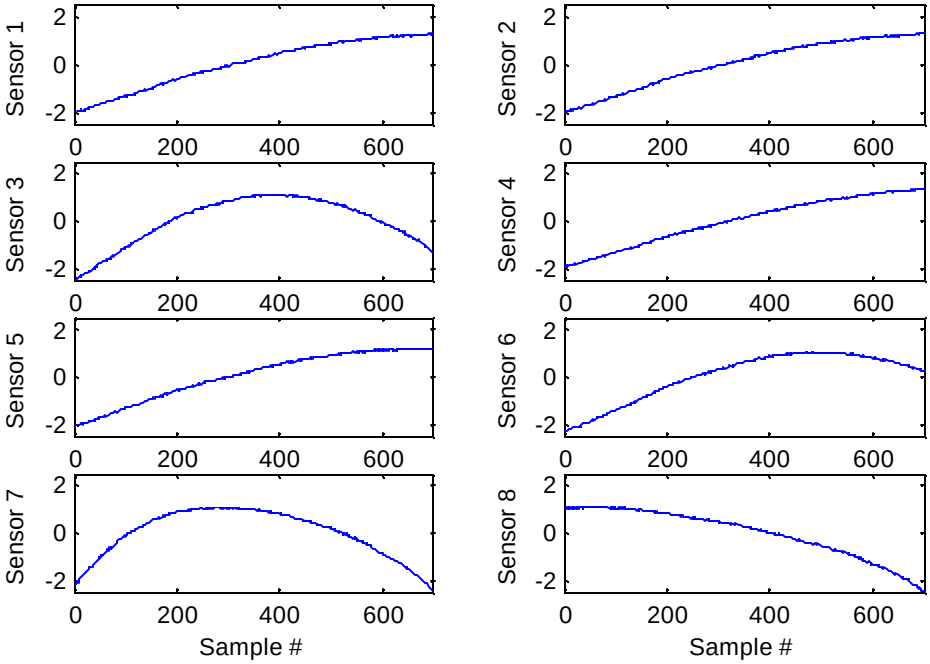

Figure 8-4, Training set

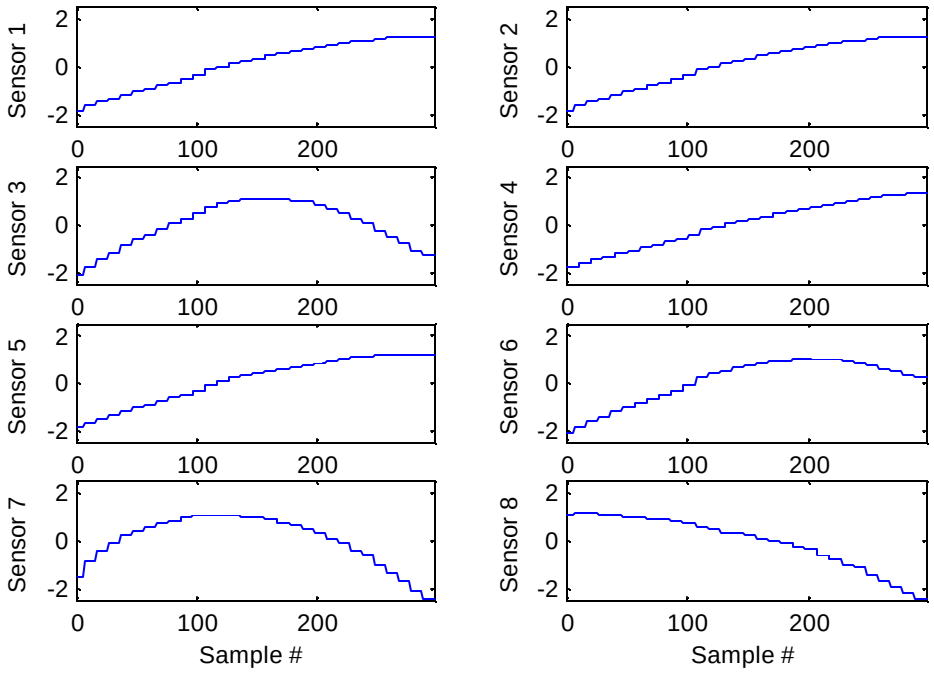

Figure 8-5, Test set, step size is 10

The E-AANN output is shown in Figure 8-6. The difference between the E-AANN input and output is shown in Figure 8-7. For better comparison we have zoomed in on the graphs of Figure 8-7 as depicted in Figure 8-8. As you see all graphs are straight lines except the fourth one. The fourth graph (the graph corresponding to the first output of the chiller model) has some perturbations but still its mean value is zero. We conclude that the perturbations in the fourth graph are from the system dynamics.

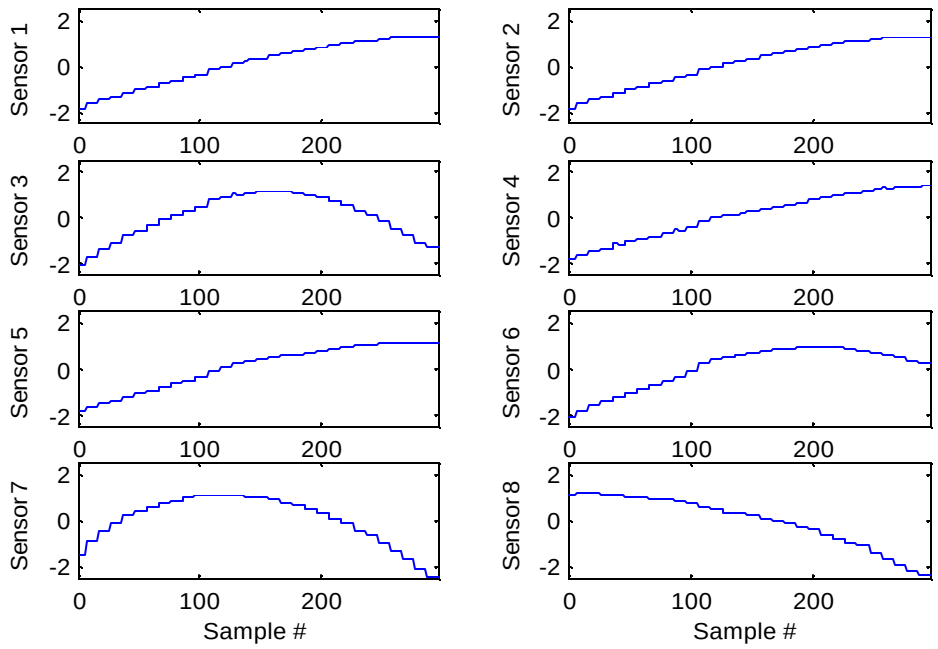

Figure 8-6, E-AANN output

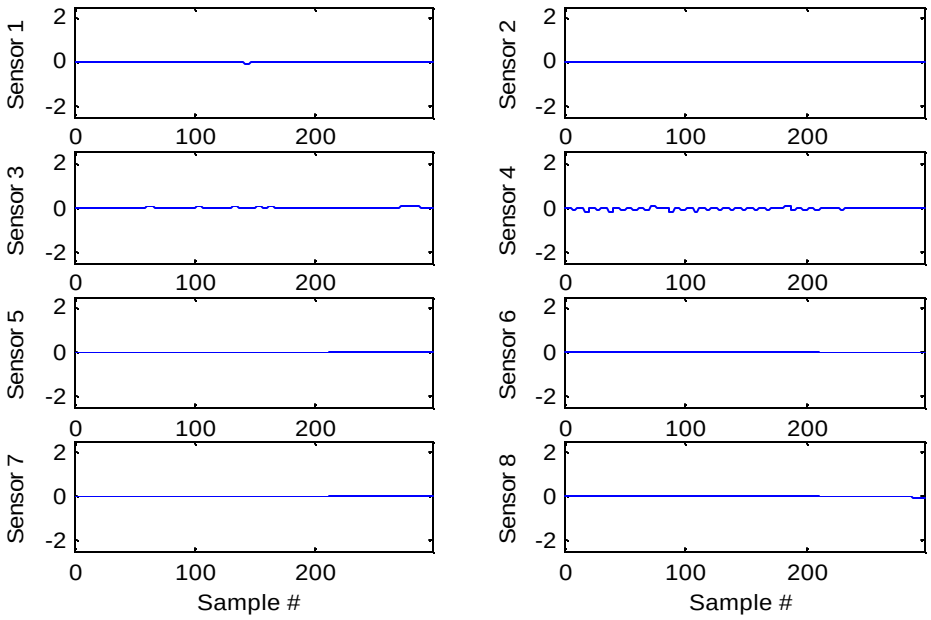

Figure 8-7, The difference between the E-AANN input and output
(delay = 4 steps)

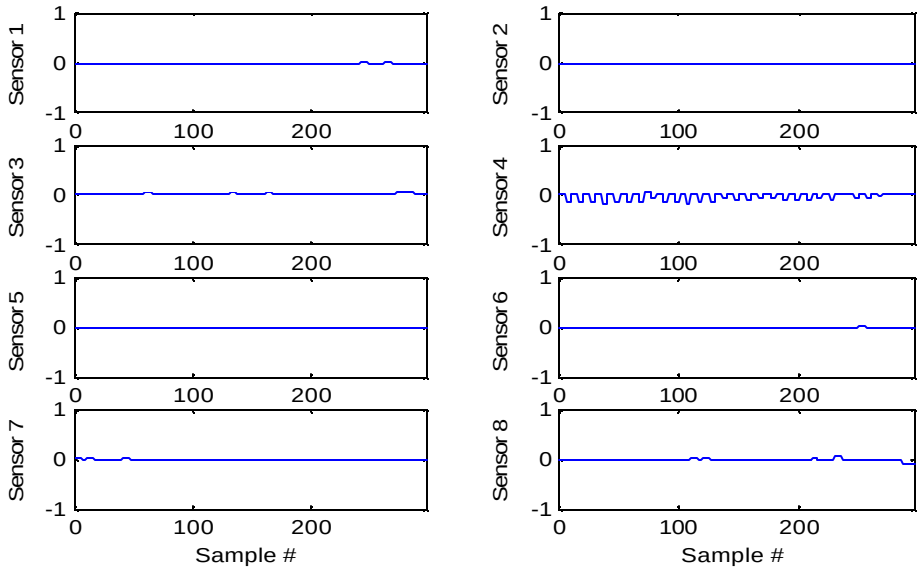

Figure 8-8, The difference between the E-AANN input and output zoomed in (delay = 4 steps)

Example 2: In this example we consider the same system but this time we increase the delay of the first output to 12 steps (Figure 8-9). The other conditions are the same. The E-AANN output is shown in Figure 8-10. The difference between E-AANN input/output is shown in Figure 8-11. For better comparison we have zoomed in on the graphs of Figure 8-11 as depicted in figure 8-12. Again you see, except the fourth graph, all graphs are straight lines. The fourth graph has some perturbations.

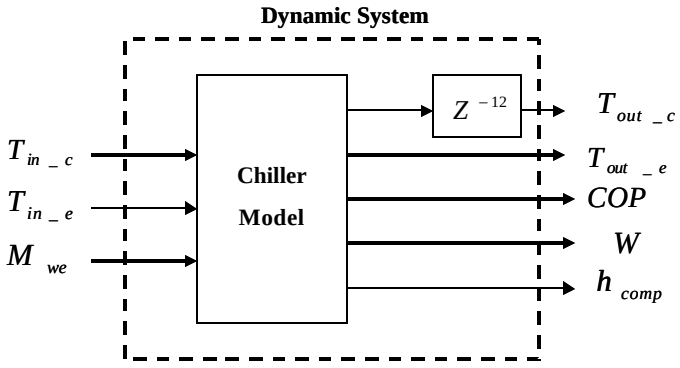

Figure 8-9, Static model with delay on one sensor to mimic dynamic system, Z^{-12} means a delay of 12 samples

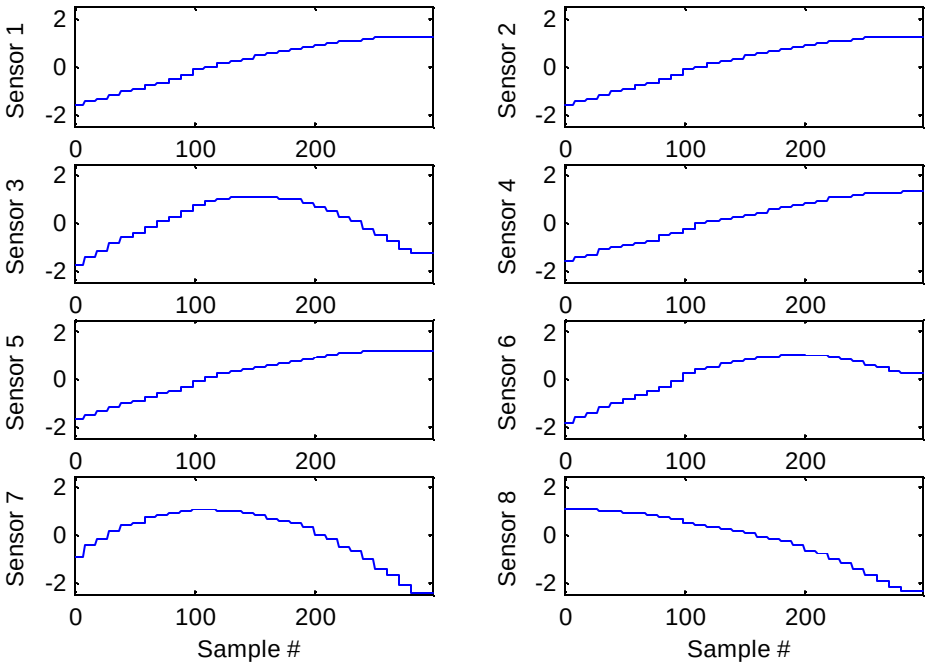

Figure 8-10, E-AANN output

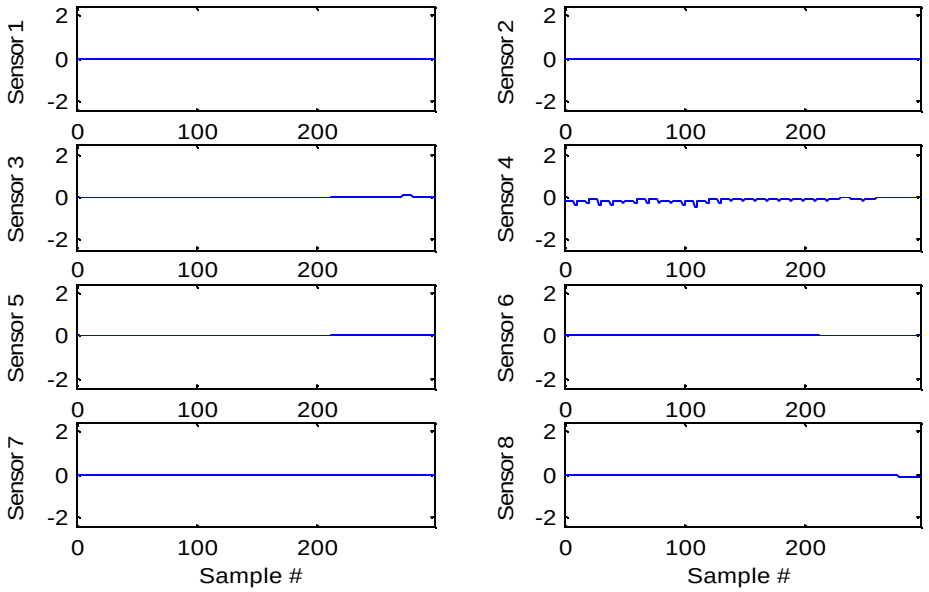

Figure 8-11, The difference between the E-AANN input and output
(delay =12 steps)

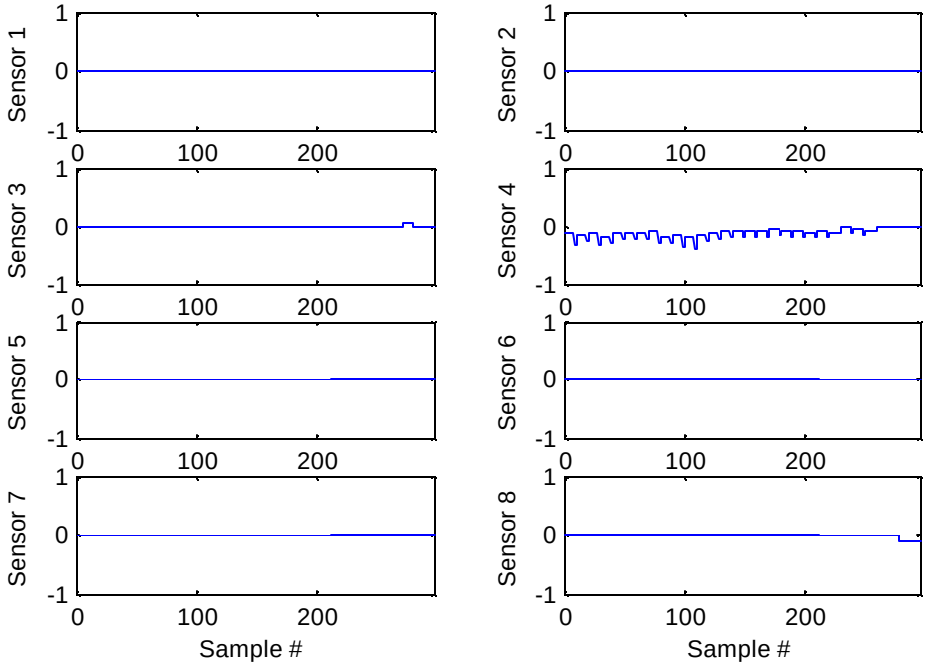

Figure 8-12, The difference between the E-AANN input and output zoomed in (delay = 12 steps)

Now the question is why E-AANN has such a behavior? Let us start with the first example. When a dynamic data set is fed to the E-AANN (In example 1, a data set whose first output still has not respond to the input change), at first E-AANN thinks that the fourth sensor (the sensor corresponding to the first output) has a problem. This is due to the fact that AANN embedded in the E-AANN has been trained based on the static data. In this situation, E-AANN reconstruct the value of the fourth sensor and change it. Therefore the difference between E-AANN input/output for the fourth sensor becomes non-zero. After a while when the first output responds to the input change, the difference between E-AANN input/output becomes zero. The above process repeats whenever the system input changes. The effect of that is small perturbations appearing in the graphs that show the difference between E-AANN input/output. Therefore we can say that the system dynamics causes small perturbations in the graphs showing the difference between E-AANN input and output.

If we zoom in on the fourth graph of Figure 8-8, it will be similar to the graph shown in Figure 8-13. In this graph there is a parameter (d) which we are interested in. The value of d depends on two parameters; 1- The system delay (in the first example the system delay was 4) and 2- The data step size (in example 1, the step size was 10). d depends on the difference between the step size and system delay ($d \propto (StepSize - Delay)$). The bigger the difference is, the bigger d is. In the first example, the difference was 6 (10 – 4 = 6) and you saw d in the Figure 8-7. In Example 2 the difference was negative (10 – 12 = -2). Before speaking about the effect of negative difference on graphs, let us at first talk about the systems that have negative difference. Do we have real systems with negative difference in reality?

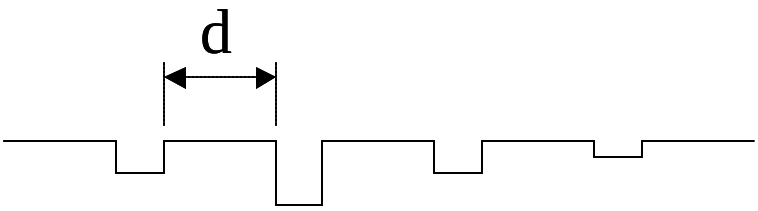

Figure 8-13, Fourth graph of Figure 8-8 zoomed in

When the difference is negative, it means that the input to the system changes before the system output responds to the previous change completely. In reality it is common to have a small negative difference. Real systems are usually in feed back control loops and sometimes the controller changes the system input before waiting to see the effect of the previous change at the output completely. However a big negative difference does not have any real application. In the case of big negative difference, the system input changes too fast to let the system output respond. This is system is unstable.

Now let see what would be the E-AANN behavior in the case of negative difference. In example 2 the difference was -2. The difference between the E-AANN input and

output was shown in Figure 8-11 and 8-12. You see that the fourth graph still has some perturbations but the important thing is that the graph fluctuates around zero. In fact the negative difference has made the fluctuation. As another example, consider the system in the second example but this time the delay of the first output is 20 (Figure 8-14). In this case the difference is -10 (10 – 20 = -10). The difference between E-AANN input and output is shown in Figure 8-15. You see that the fourth graph again has a smooth fluctuation around zero

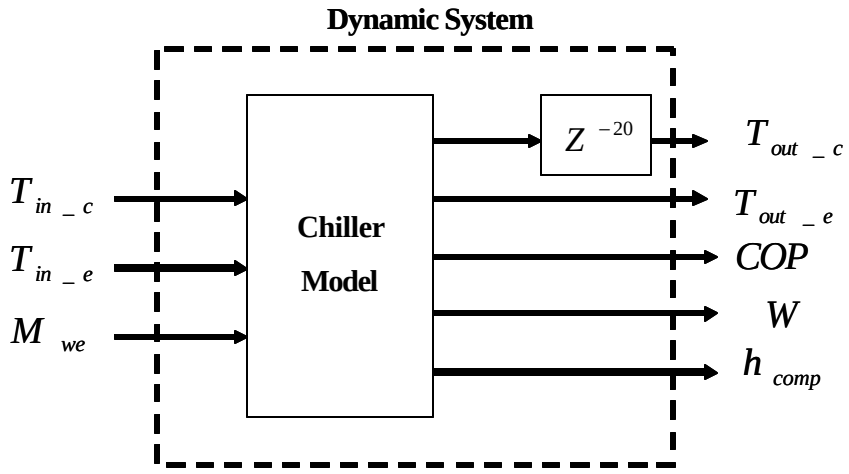

Figure 8-14, Static model with delay on one sensor to mimic dynamic system, Z^{-20}means a delay of 20 samples

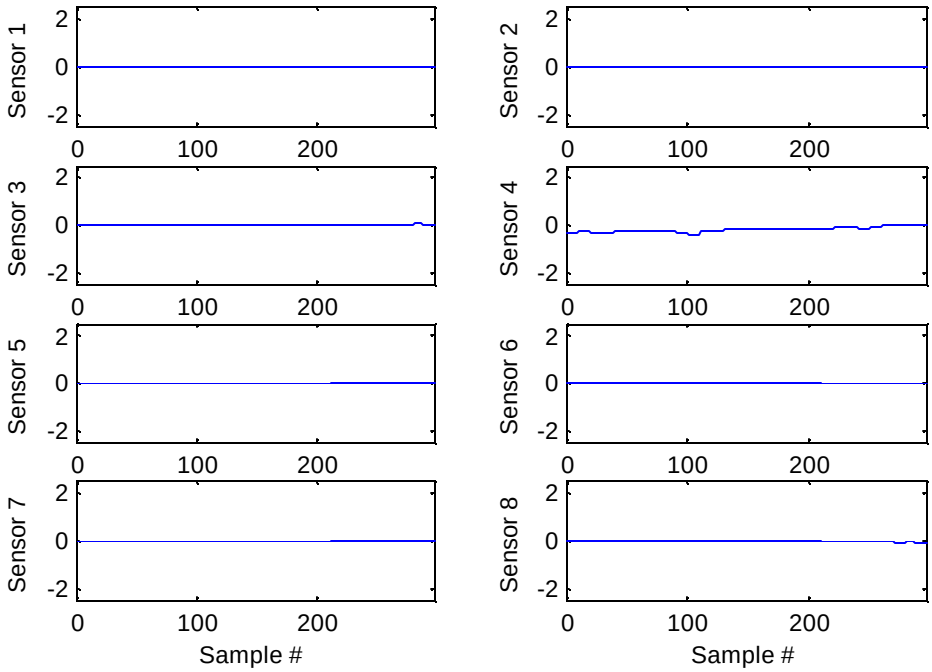

Figure 8-15, The difference between the E-AANN input and output
(delay = 20 steps)

Example 3: In this example we work on a more complicated dynamic system. We modify the chiller model in a way that the last three outputs of the system have delay with four samples (Figure 8-16). The training data is the same as the training data shown in Figure 8-4. The test data is shown in Figure 8-17. The E-AANN output is shown in Figure 8-18. The difference between E-AANN input and output is shown in Figure 8-19. In spite of the fact that some graphs have small perturbations, the mean value of all graphs is zero.

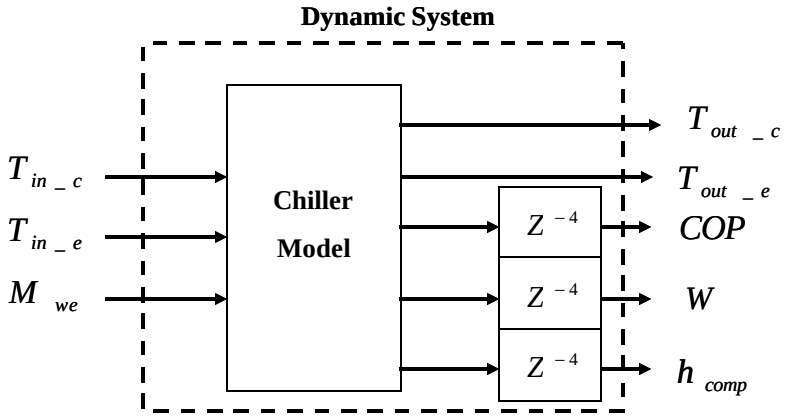

Figure 8-16, Static model with delay on multiple sensors to mimic dynamic system

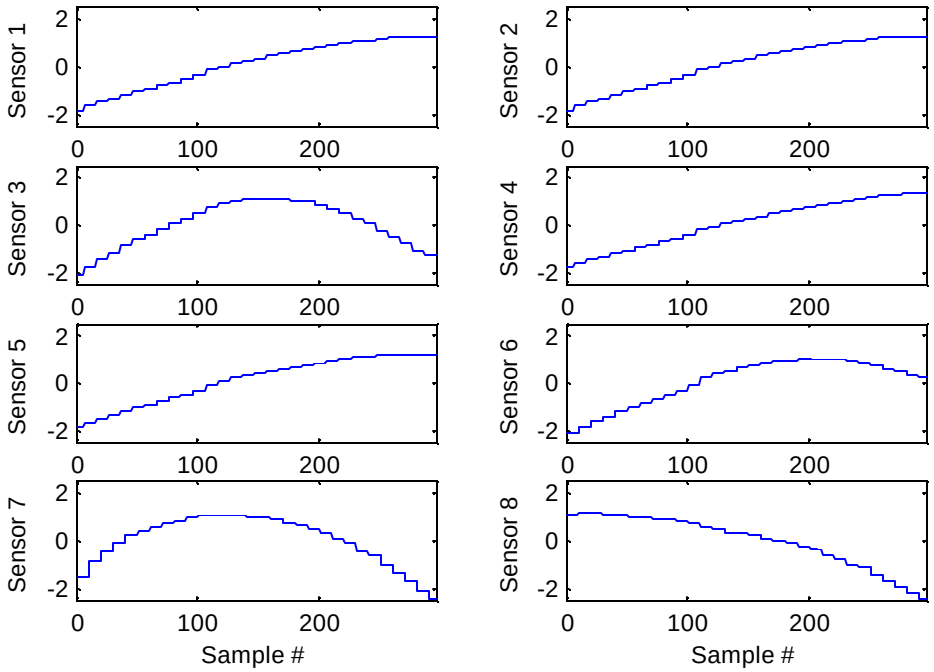

Figure 8-17, Test data

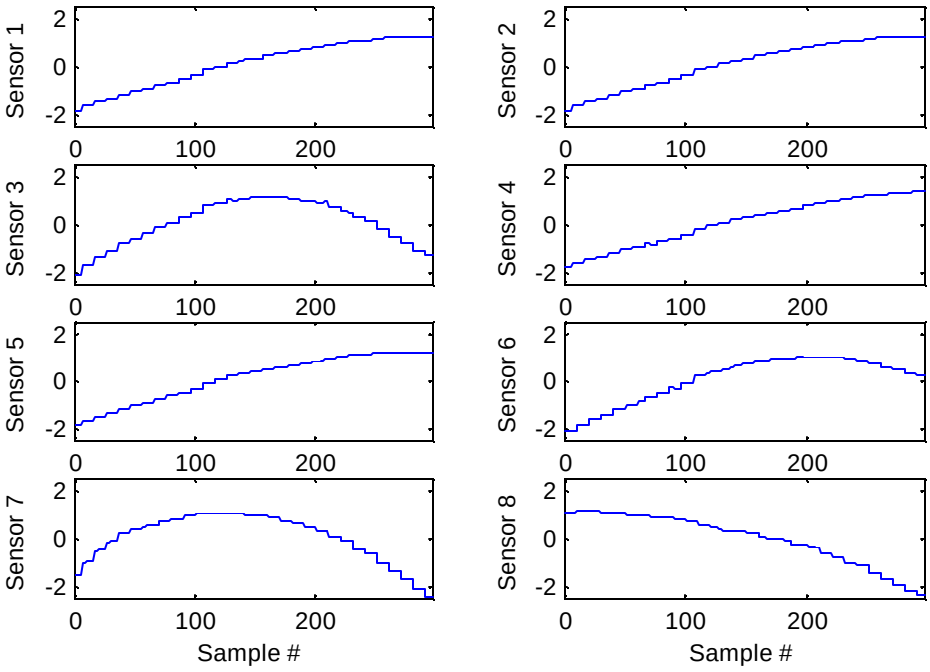

Figure 8-18, E-AANN output

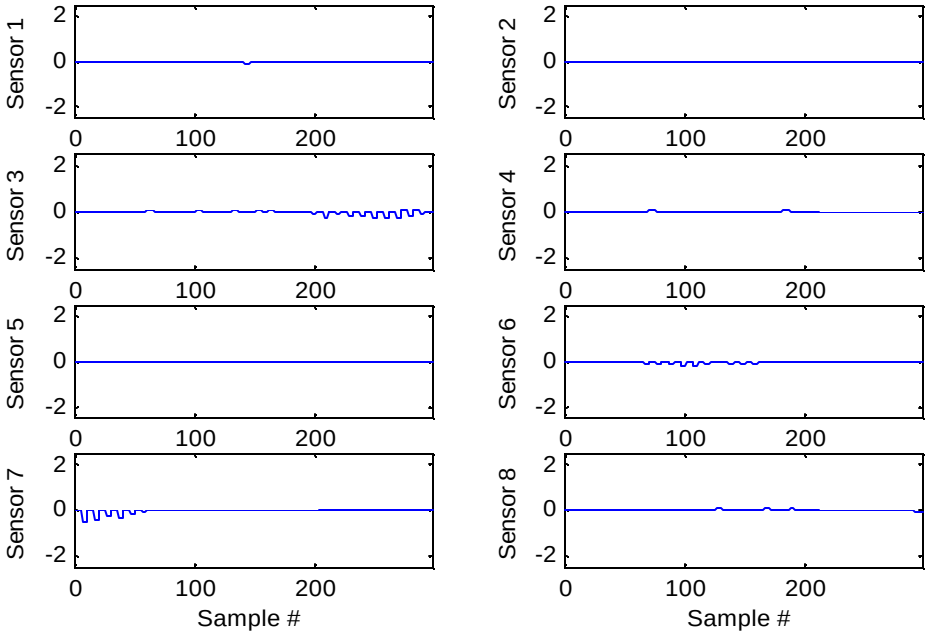

Figure 8-19, The difference between the E-AANN input and output

8.3 E-AANN Application in Sensor Diagnostics in Dynamic Situations

As you saw in previous examples, system dynamics causes some perturbations in the graphs showing the difference between the E-AANN input/output. The mean values of the graphs are zero in the case of positive difference between the data step size and system delay or non-zero in the case of negative difference. We also explained that in real applications the difference between the step size and system delay is positive or small negative. This means that the mean value of the graphs showing the difference between E-AANN input/output is zero or close to zero.

In previous sections we showed that the sensor errors effectively change the mean value of the graphs showing the difference between E-AANN input and output. Therefore we believe that E-AANN can still be used to catch sensor faults in dynamic situations. In the case of positive difference between the data step size and system delay, the mean values of the graphs are zero as far as there is no sensor error. When the mean value is non-zero, it means that there is sensor problem. In the case of small negative difference, the mean values of the graphs are zero or close to zero. In this situation E-AANN might not catch small sensor faults.

Example 4: Consider the third example. Here we induce a drift error[22] to the sensor measuring the third output. The test set is shown in Figure 8-20. The E-AANN output is shown in Figure 8-21. The difference between the E-AANN input and output is shown in Figure 8-22. Now you see that the sixth graph (the graph corresponding to the sensors measuring the third output) has non-zero mean value which means that the corresponding sensor has a problem.

[22] The same drift error we had in previous sections

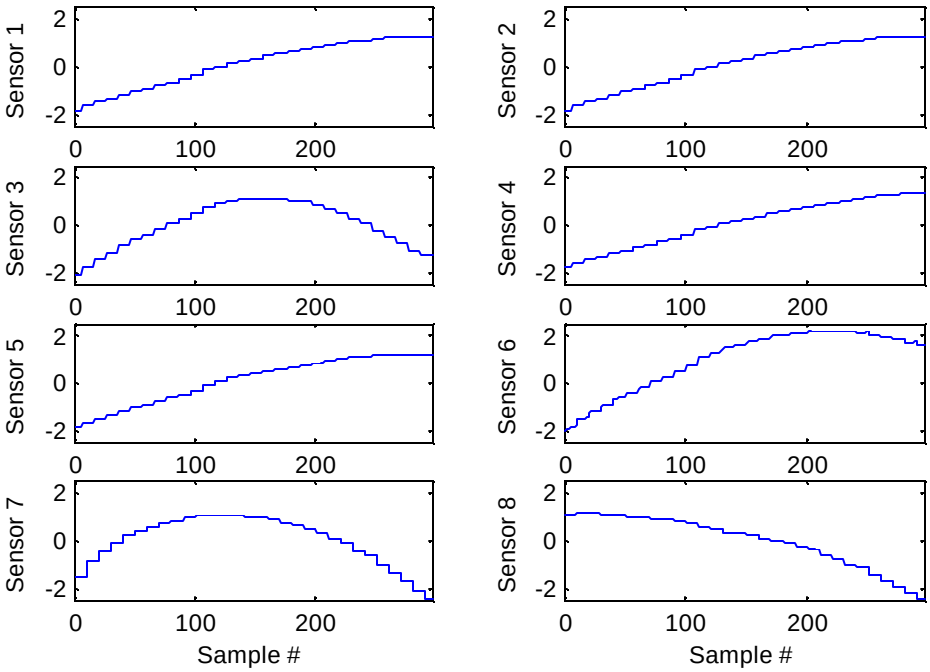

Figure 8-20, Test Data, the sixth sensor has drift error

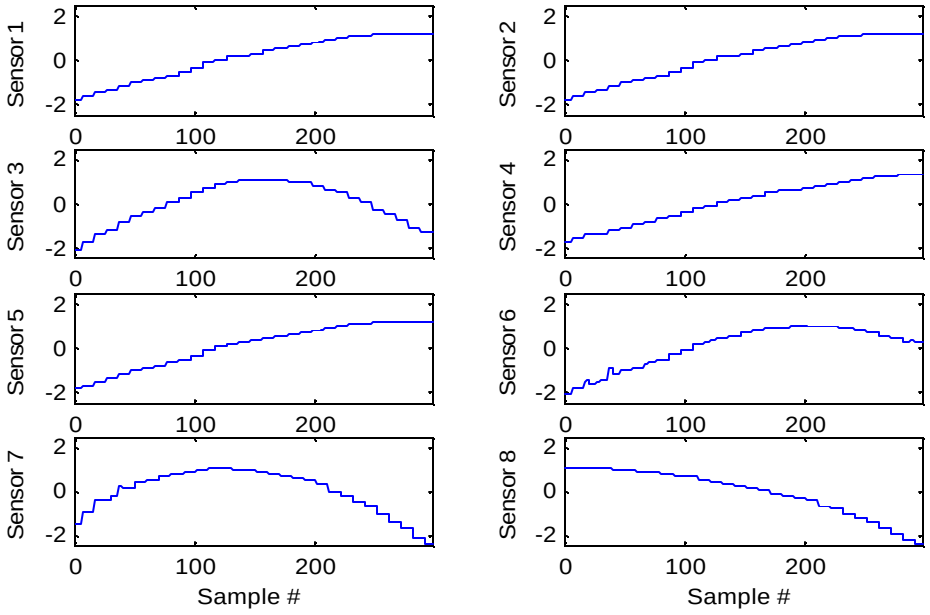

Figure 8-21, E-AANN output

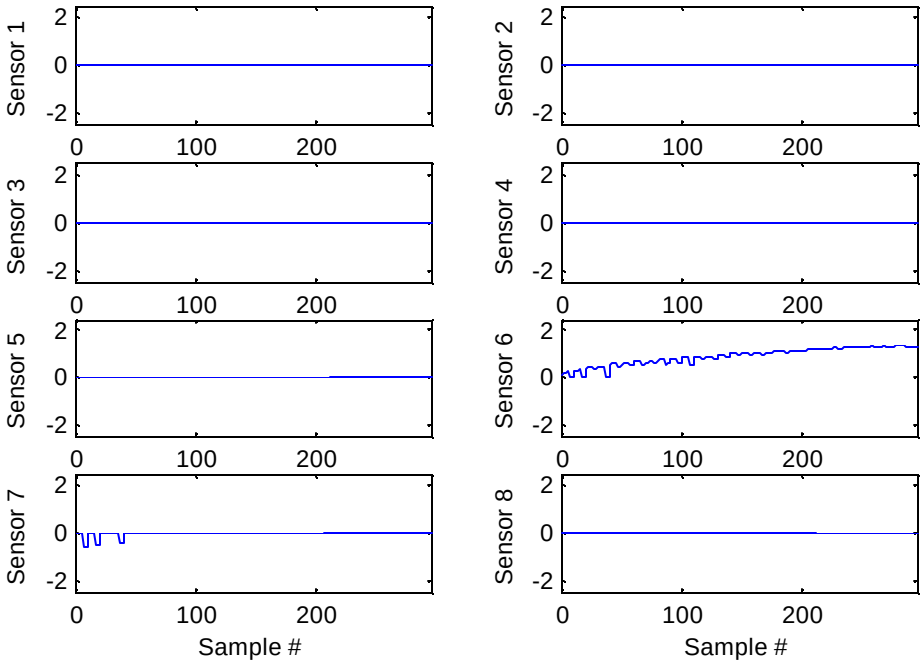

Figure 8-22, The difference between the E-AANN input and output

Based on the dynamic systems stated in this section, E-AANN can still be used to catch sensor faults. You saw that system dynamics causes small perturbations on the delta graphs (the graphs show the difference between E-AANN input and output) but still the mean values are zero. However, E-AANN performance must be evaluated in the case of more complicate dynamic systems.

9. CONCLUSION

The use of AANN in sensor diagnostics has been extended to allow the isolation of individual sensor faults. This extension implements a secondary optimization process, which enables the user to make effective use of the AANN concept in diagnosing sensor faults. The examples given illustrate the effectiveness of this approach. E-AANN performance was studied under noisy conditions. The results showed that sensor faults can be detected and corrected in noisy situations with the E-AANN method described.

In high noise situations, the E-AANN performance degrades. The results show that as the noise level increases, the E-AANN accuracy decreases. In other words, E-AANN may not catch small sensor faults when the noise level is high. This is due to the fact that in high noise situations, the symptoms of fault response are hidden. The use of filters could improve E-AANN performance in high noise situations (future work).

E-AANN performance in simple dynamic systems was also evaluated. It was shown that E-AANN is still capable of locating sensor faults in dynamic systems if the training data is based on static data. However more study needs to be done to evaluate E-AANN performance in more complicated dynamic systems

The next major investigation is to deal with multiple sensor faults. In this project, it was assumed that only one sensor was faulty. However, in the case of multiple sensor faults, the problem is much more complicated.

REFERENCES

1. M.A. Kramer, “Autoassociative neural networks”, *Computers in Chemical Engineering*, vol. 16, no 4, pp. 313-328, 1992.

2. M.A. Kramer, “Nonlinear principle component analysis using autoassociative neural networks”, *AIChE Journal*, vol. 37, no. 2, pp. 233-243, February 1991.

3. R. Prabhu, “A neuro computational approach to chiller fault identification and isolation”, *Master’s Thesis, Texas A&M University, Mechanical Engineering*, 2002.

4. M. Shajith Ikbal, H. Misra, B. Yegnanarayana, "Analysis of autoassociative mapping neural networks", in *International Joint Conference on Neural Network*, vol. 5, pp 3025 –3029, 1999.

5. J.W. Hines and R. E. Uhrig, "Use of autoassociative neural networks for signal validation", *Journal of Intelligent and Robotic Systems*, pp.143-154, 1998.

6. J.W. Hines, D.J Wrest, and R.E. Uhrig, “Plant wide sensor calibration monitoring, intelligent control”, *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, pp. 378 –383, 1996.

7. W.Y. Lee, C. Park, and G.E. Kelly, "Fault detection in an air-handling unit using residual and recursive parameter identification methods", *ASHRAE Transactions*, vol. 102, no. 1, pp. 528-539, 1996.

8. Keyhani and S.M. Miri, "Observers for tracking of synchronous machine parameters and detection of incipient fault", *IEEE Transaction on Energy Conservation*, vol. 1, pp. 184-190, 1998.

9. H. Konrad, R. Isermann, and N. Heintz, "Model based fault detection in milling by classification of estimated cutting parameters", in *IEEE International Conference on Intelligent Systems for 21st Century*, vol. 3, pp. 2193-2198, Vancouver, 1995.

10. M.Y. Chow and S.O. Yee, "Methodology for on-line incipient fault detection in single phase squirell cage induction motors using artificial neural networks", *IEEE Transaction on Energy Conservation*, vol. 6, 536-545, 1991.

11. M.Y. Chow and S.O. Yee, "Application of neural networks to incipient fault detection in motors", *Journal of Neural Network Computing*, vol. 2, no. 3, pp. 26-32, 1991.

12. M.Y. Chow, P.M. Mangum, and S.O. Lee, "A neural network approach to real-time condition monitoring of induction motors", *IEEE Transaction Ind. Electron.*, vol. 38, pp. 448-453, 1991.

13. M.Y. Chow, R.N. Sharpe, and J.C. Hung, "On the application of artificial neural networks for motor fault detection", *IEEE Transaction Ind. Electron.,* vol. 40, pp. 181-196, 1993.

14. X. Z. Gao and S. J. Ovaska, "Soft computing methods in motor fault diagnosis," *Applied Soft Computing Journal*, vol. 1, no. 1, June 2001.

15. H. Nejjari and M.E.H. Benhouzid, "Application of fuzzy logic to induction motors condition monitoring", *IEEE Power Eng. Rev*., vol. 19, pp. 52-54, 1999.

16. M. Yumoto, T. Ohkawa, N. Komodo, and F. Miyasaka, "Practical application of stochastic qualitative reasoning to fault detection of building air conditioning systems", in *Proc of Tenth International Workshop on Qualitative Reasoning*, pp. 283-291, 1996.

17. M. Yumoto, T. Ohkawa, N. Komoda, and F. Miyasaka, "An approach to automatic model generation for stochastic qualitative simulation of building air conditioning systems", in *Proc of IEEE International Symposium on Industrial Electronics*, pp. 1037- 1042, Trento, Italy, 1996.

18. W.W. Tan and H. Hou, "An on-line neurofuzzy approach for detecting faults in induction motors", *IEEE International Conference on Machines and Drives*, pp. 878-883, 2001.

19. M.E.H. Benhouzid and H. Nejjari, "A simple fuzzy logic approach for induction motors stator condition monitoring", *IEEE International Conference on Machines and Drives*, pp. 634-639, Boston, 2001.

20. JPH. Bourdouxhe, M. Grodent, JJ. Lebrun, C. Saavedra, KL. Silva: "A toolkit for primary HVAC system energy calculation- Part 2: Reciprocating chiller models." *ASHRAE Transaction*, vol. 100, no. 2, pp. 774-786, 1994.

21. KC. Ng, HT. Chua, W. Ong, SS. Lee, JM. Gordon: "Diagnostics and optimization of reciprocating chillers: Theory and experiment." *Appl. Thermal Eng*., vol. 17, no. 3, pp. 263-276, 1996.

22. K C Ng, H T Chua, A S Ong. Experimental verification of a diagnostic model for reciprocating chillers. *Journal of Process Mechanical Engineering (Part E),* vol. 211, no. 4, pp. 259-265, 1997.

23. Pacific Gas and Electric, “CoolTools: A toolkit to optimize chilled water plants”, San Francisco, CA, 2001, http://www.hvacexchange.com/cooltools

24. TS. Steele, WE. Koran, MB. Kaplan: “DOE-2.1C model calibration with short-term tests versus calibration with long-term monitored data” in *Proceedings from the ACEEE 1994 Summer Study on Energy Efficiency in Buildings.* Washington, D. C.: American Council for an Energy Efficient Economy.

25. P. Sreedharam and P. Hvaes: “Comparison of chiller modes for model-based fautl detection” available at: http://www.energy.ca.gov/pier/buildings/presentations/40099012_E5_P5_3c.pdf

26. M. Stylianou and D. Nikanpour: “Performance monitoring, fault detection and diagnosis of reciprocating chillers.” *ASHRAE Transactions*, vol. 102, no.1, pp. 667-679, 1996.

27. JE. Braun and TM. Rossi: “A statistical, rule-based fault detection and diagnosis method for vapor compression air conditioners.” *International Journal of HVAC & Ref. Research,* vol. 3, no. 1, pp. 19-37, 1997.

28. HT. Grimmelius, JK. Woud, and G. Been: “On-line failure diagnosis for compression refrigeration plants.” *International Journal of Refrigeration*, vol. 18, no. 1, pp. 31-41, 1995.

29. S. Wang and J.B. Wang: “Law-based sensor fault diagnosis and validation for building air-conditioning systems”, *International Journal of HVAC & Ref. Research,* vol. 5. pp: 353-380, 1999.

30. J.M. Gordon and K.C. Ng "Predictive and diagnostic aspects of a universal thermodynamic model for chillers". *International Journal of Heat and Mass Transfer*, vol. 38, pp. 807-818, 1995.

31. A. Rizzo and M.G. Xibilia, “An innovative intelligent system for sensor validation in tokamak machines”, *IEEE Transactions on Control Systems Technology*, vol. 10, no. 3, pp. 421-431, May 2002.

32. S. Satoh, M.S. Shaikh, and Y. Dote, “Fast fuzzy neural network for fault diagnosis of rotational machine parts using general parameter learning and adaptation”, in *Proc of the 2001 IEEE Mountain Workshop on Soft Computing and Industrial Applications,* pp. 87-91, 2001.

33. T. Yamasaki, M. Yumoto, T. Ohkawa, N. Komoda, and F. Miyasaka, “Automatic parameter tuning of stochastic qualitative model of building air conditioning

systems", in *Proc of IEEE International Conference on Intelligent Engineering Systems*, pp.415-420, 1997.

34. P. Amann, J.M. Perronne, G.L. Gissinger, and P.M. Frank, "Identification of fuzzy relational models for fault detection", *Control Engineering Practice*, vol. 9, no. 5, pp. 555-562, 2001.

35. Class Notes: "Introduction to Pattern Recognition", R. Guiterrez-Osuna, CPSC 689. Fall 2002, Texas A&M University, available at: http://faculty.cs.tamu.edu/rgutier/courses/cpsc689_f02/index.html

VITA

Massieh Najafi was born in 1977 in Iran. He received his B.Sc. degree in Mechanical Engineering from University of Tehran in Iran in 2000. He then started his graduate studies in the Mechanical Engineering Department at Texas A&M University. Mr. Najafi has been research assistant in the Mechanical Engineering Department of Texas A&M from September 2001 to February 2002, research assistant at Energy System Laboratory at Texas A&M University from February 2002 to August 2003. His research interests are fault detection, sensor diagnosis, intelligent systems, and controls.