Artificial Intelligence Techniques Applied to Fault Detection Systems

by

Daniel Fischer

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Doctor of Philosophy

McMaster University
April 2004

DOCTOR OF PHILOSOPHY (2003) McMASTER UNIVERSITY
(Electrical and Computer Engineering) Hamilton, Ontario

TITLE: Artificial Intelligence Techniques Applied to Fault Detection Systems

AUTHOR: Daniel Fischer,
B.A.Sc. (E.E.) University of Toronto
M.A.Sc. University of Toronto

SUPERVISOR: W.F.S. Poehlman, Professor, Department of Computing and Software

CO-SUPERVISOR: B. Szabados, Professor, Department of Electrical and Computer Engineering

NUMBER OF PAGES: VIII, 195

Abstract

This thesis presents novel applications of Artificial Intelligence-based algorithms to Failure Detection Systems. It shows the benefits intelligent, adaptive blocks can provide along with potential pitfalls. A new fault detection structure is introduced which has desirable properties when dealing with missing data, or data corrupted by extraneous disturbances. A classical alarm generation procedure is extended by transformation into an optimum, real-time, adaptive block.

Two techniques, artificial Neural Networks, and Partial Least Squares, complement each other in one of the failure detection applications exploiting their respective non-linear and de-correlation strengths.

Artificial Intelligence techniques are compared side by side with classical approaches and the results are analyzed.

Three practical examples are examined: The Static Security Assessment of Electric Power Systems, the Oil Leak Detection in Underground Power Cables, and the Stator Overheating Detector. These case studies are demonstrated since each one represents a class of failure detection problems.

The Static Security Assessment of Electric Power Systems is a class of problems with inputs which are somewhat correlated, and which has very little learning data. While the time required for the system to learn is not a concern, the recall time must be short, providing for real-time performance.

The Oil Leak Detection in Underground Power Cables represents the class of problems where one has vast amounts of data indicative of a properly functioning system, however data from a failed system are very sparse. Unlike the Static Security Assessment problem, the oil leak detector has to consider the time dynamics of the system. Special

provisions must be made to accommodate missing data which would interrupt contiguous data sets required for proper operation. This case study shows ways to exploit the slight sensor redundancy in order to detect sensor breakdown along with the detection of the main system failure.

A third class of problems is showcased by the Electric Generator Stator Overheating detector. This application must deal with highly correlated inputs, along with the lack of fault data to be used for learning. Physical system non-linearities as well as time dynamics must also be addressed.

Acknowledgements

The author wishes to thank Dr. Skip Poehlman for his encouragement and supervision during the course of this work.

He also gratefully thanks Dr. Barna Szabados for his invaluable comments and guidance regarding this thesis and technical papers written on related subjects, as well as for his financial support that allowed the author to attend and present papers at several conferences.

The author would like to thank Dr. Sekerinski for serving as a member of his Supervisory Committee and for his continuing interest.

Thanks are due to the author's family members who have been consistently supportive of his efforts.

Last, but not least, the author would like to thank Arun Narang of Kinectrics Inc. for the many, long, discussions on topics this work is based on.

Table of Contents

List of Figures

CHAPTER I

1. Introduction to the Fault Detection Systems Problem

A Fault Detection System (FDS) is an algorithm whose purpose is to produce an alarm signal whenever the monitored device experiences the beginning of a breakdown. While classical implementations of different building blocks used by FDS have been described in the literature, Artificial Intelligence based approaches can make a positive contribution. This chapter introduces generic Fault Detection Systems, followed by the Problem Definition, Objectives, and Methodology of this work. The introduction of three real-world FDS classes follows. A list of contributions made in this research followed by the organization of the thesis, completes the current chapter.

1.1 A Generic Fault Detection System

Modern systems are often faced with unexpected changes such as component faults resulting in a degradation of overall performance [1]. In order to maintain a high level of reliability, fault tolerant systems must be able to detect such changes as soon as possible. A monitoring system, which has the capability of detecting faults, is called a Fault Detection System (FDS). A FDS can be implemented by using either hardware redundancy or analytical redundancy. Hardware redundancy implies the use of several identical or similar sensors whose outputs are compared for consistency [2], [3]. While a valid and often used approach, hardware redundancy results in an enlargement of equipment, needing additional volume, weight, complexity, and cost.

Analytical (or functional) redundancy is based on inherent relationships among measured quantities present in the system [4]. Analytical redundancy is also known as Quantitative Model-Based Fault Detection.

A literature search of the analytical redundancy approaches, shows that the published methods fall into one of the two groups:

1. parameter estimation methods
2. state (or output) estimation methods

There is a relatively small, published, body of work that uses the parameter estimation technique. This approach detects a fault by tracking the abnormal variation of model parameters. This implies the need for on-line identification and estimation, a computationally intensive proposition. Secondly, the estimated coefficients belong to a mathematical model of the process, and not to the physical process itself. The fact that there are several mathematical models which could equally well correspond to a physical system, implies that the variation of identified coefficients does not necessarily mean a fault, but could just be a computational byproduct.

The state estimation approach is the analytical redundancy method of choice. Like the parameter estimation methods, it also requires some sort of mathematical model of the system under consideration. However, unlike the parameter estimation technique which needs an on-going re-estimation of model coefficients, the state estimation makes use of a model computed only once (or seldom enough for the physical system to be considered stationary).

Figure 1-1 shows a block diagram of the state estimation technique. The Nominal Model block shows a parametric mathematical model that computes the state variables a properly functioning system would produce with no fault present. The Estimated Operating Model computes the actual state variables taking into account the physical system output vectors. If the state values monitored by the FDS are the physical system outputs, then the Estimated Operating Model is reduced to simple connections (one line for each monitored output). The Residual Generator computes the distance between the estimated nominal and actual state vectors. The value of the distance is fed into a Decision Block responsible for issuing the alarm marking a detected fault.

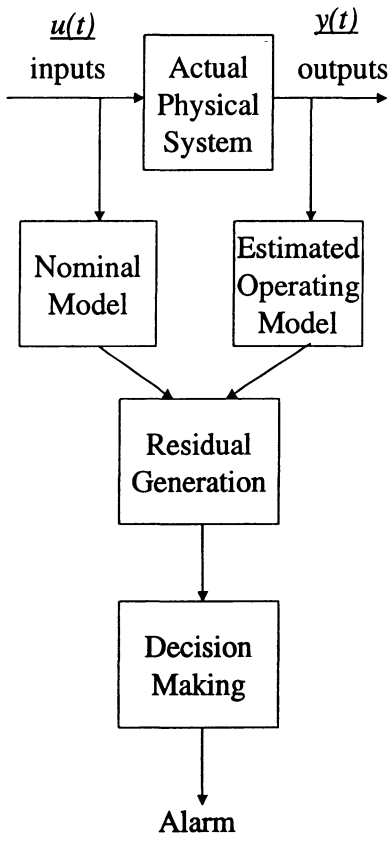


Figure 1-1 Fault Detection System block diagram.

A FDS design must consider a number of implementation choices. The mathematical models used can be linear [1], [2], [3], [4], [5], [6] (model linearized around an operating point), or non-linear [7], [8], [9], [10], [11], [12], [13]. The mathematical models can be constructed using observers (filters) in a deterministic setting, or in a stochastic setting (Kalman filters) [14], [15], [16]. The model choice will strongly impact the type of pre-processing needed to condition the input and output data. Such operations as: mean removal, re-scaling, differentiation, dimensionality reduction, feature extraction, are often called for. The goal of the mathematical model is to allow the computation of a residual, which is ideally zero if no system fault is present, and non-zero, otherwise. A zero residual is unlikely in practical implementations, model uncertainty being one of the

important issues affecting the performance of an analytical redundancy scheme [11], [12]. Model uncertainty refers to the degree to which a mathematical model can estimate the state of a physical system. As Petrick and Wigdorowitz [17] have pointed out, "The engineer who is under the impression that modeling will give rise to the "true" system is bound to be bitterly disappointed, especially if the system exhibits a variety of nonlinear dynamic behaviour in the operating region of interest, useful for the application in mind". There are no correct models. Some are useful.

The Decision Block must use the residual value and decide if an alarm output should be asserted. A decision process may be as simple as a threshold test applied on the instantaneous value of the residual, or may be implemented based on the statistics of the residual. The main characteristic of the Decision Block is its accuracy. When implementing the decision algorithm, a balance is struck between a tendency to false alarm and one to fail to alarm. The choice of decision threshold, coupled with, in certain instances, an ability to adjust the threshold depending on the residual values, may result in superior fault detection capabilities. Regardless of whether or not the residual statistics are explicitly used in the construction of the Decision Block, the alarm algorithm together with the residual are characterized by a false alarm and fail to alarm probability. If the costs of a fail to alarm and a false alarm are known (or estimated), a classifier, which minimizes the cost of making a wrong decision, can be constructed. Details of the classifier design will be available in the presented test cases.

Another characteristic of the Decision Block is its decision time. Typically, by low pass filtering the residual value, the Decision Block can improve on its false trip performance. Of course, low pass filtering delays the initiation of an alarm.

In our work, we will consider three real-life case examples of Failure Detection Systems:

1. The Electric Power Systems Static Security Assessor
2. The Underground Power Cable Oil Leak Detector
3. The Flow Restrictions in Water-Cooled Generator Windings Detector

Besides being practical applications of an Artificial Intelligence based FDS methodology, these examples represent distinct classes of problems.

The Static Security Assessment problem must deal with a large number of inputs which can be grouped into sets that have some inter-set correlation level. The amount of input / output learning data is extremely small, resulting in a high danger of model overfitting. This fact must be mitigated by the adopted implementation. Finally, the input / output relationship is nonlinear.

The Oil Leak Detection problem has a small number of input signals. Unlike the Static Security Assessment problem, the Oil Leak Detector must deal with system time dynamics. There are significant time lags between changes in the input signals and changes in the outputs. Large, non-linear, physical system perturbations corrupt the data. Also, data loss is a major challenge in this class of problems.

The Stator Bars Blockage problem deals with a large number of highly correlated inputs. While this correlation is beneficial, resulting in resilience to hardware breakdowns, it has to be properly handled in order to avoid near-singularities during numerical processing. Published approaches use a non-linear input / output model. By exploiting the mentioned data correlations, we show how this class of problem can be linearized, resulting in faster learning while maintaining a very high accuracy.

1.2 Problem Definition

Several papers have proposed solutions to the fault detection problem. All approaches involve system modeling, the calculation of a residual, and a decision (alarm) block. There are many approaches to the system modeling state: "classical" models such as Least Mean Squared (LMS), Kalman filtering, Partial Least Squares (PLS) as well as

Artificial Intelligence based systems such as Neural Networks, Fuzzy Logic, and Expert Systems.

Based on the literature research that has been performed, no published work compares these techniques in the context of fault detection system applications and, in the process, outlines their respective strengths and weaknesses. Issues such as: linear vs. non-linear modeling, the effect of correlated inputs, the effect of small learning data sets, the effect of the total lack of learning data representative of a failed system, the effect of missing data samples in the learning and testing sets, have not been addressed as applied to fault detection systems.

1.3 Objectives

The objective of this work is to extract the positive features of the classical model and alarm generation techniques and integrate them within an Artificial Intelligence unified framework. The developed methodology will then be applied to three real-life test cases that represent three classes of fault detection problems:

1. The Static Security Assessment to Electric Power Systems
2. Oil Leak Detection in Underground Power Cables
3. Power Generator Stator Bars Coolant Blockage Detector

1.4 Methodology

Various technologies proposed in the literature will be analyzed and their performance assessed, having in mind our classes of fault detection problems. System Identification and Time Series Analysis will provide the methodology for realizing linear process models simultaneously with accounting for disturbances via a disturbance model. In order to account for non-linearities, or to perform unsupervised learning for clustering,

Neural Networks will be used. Fuzzy Logic will be used to combine the outputs of several fault detection methodologies into one final decision alarm signal. Partial Least Squares will be applied for both modeling as well as providing assistance to Neural Networks in a new unsupervised data clustering algorithm.

In order to produce an alarm output signal, a classifier is needed. A Bayes classifier will be used in several of the test cases. The classifier will be extended into a new, real-time module, with on-line learning capability.

1.5 Case Studies

1.5.1 The Static Security Assessment of Electric Power Systems

Voltage stability is concerned with the ability of a power system to maintain acceptable voltages at all buses in the system under normal conditions and after being subjected to a disturbance [18]. A system experiences voltage instability, when an increase in load demand, or a change in system conditions, coupled with a certain system operating point, result in a progressive and uncontrollable decline in voltage. The main reason causing this voltage collapse is the inability of the system to meet the demand for reactive power [18], [19].

The weak link in the power system security assessment task is the calculation of the operating limits (e.g. transmission line flows, bus voltages, transmission line thermal levels). Due to the numerically intensive nature of the used algorithms, these limits are computed off-line. Two of the input requirements for these calculations, are future customer demand and equipment availability. Future customer demand is not available at the time of computation and must be forecasted introducing uncertainty. This uncertainty is mitigated by increasing the security limits.

To mitigate the high computational burden caused by these studies, the limits are derived based on off-line simulations for only a limited number of operating conditions. Experts for each area must pick the critical contingencies and configurations, based on past experience. The initial base case (power flow for Winter peak, Summer peak or Spring low load conditions) for each area is selected based on the type of problem under consideration. For example, Summer peak load conditions are normally used for studies of thermal limits. The total system or area load level is based on forecasts, which are updated annually to reflect changing customer plans. The generation dispatch is selected based on historical operation and planned outage data. Significant differences between any of these assumptions and the prevailing conditions will affect the accuracy of the limits. An extra margin is built into the limits to allow for these sources of inaccuracy. The critical parameter is reduced from the point of simulated instability to arrive at the limit. This approach is considered crude and may present an opportunity for improved economic operation through reduction of the error margin. The off-line limits are reviewed at regular intervals (annually) or whenever major facility changes are planned.

During the power system operation, data are transmitted from all critical facilities (230 KV and 500 KV) to the Control Centre every 2 seconds (Ontario Hydro practice). These data are collected and processed by the Data Acquisition and Computing System (DACS). Every 6 seconds, a snapshot of the prevailing system state is created and compared with the limits that have been computed off-line. Major differences between the studied cases and the existing conditions, must be handled by assigning penalty factors to the limits, or performing simulations for the new configuration.

To summarize, the real-time portion of the present power system security assessment process involves the following steps:

- acquire a snapshot of measurable quantities pertinent to the power system
- perform state estimation to determine unknown quantities and detect/adjust inconsistent data
- compare operating levels with off-line calculated limits based on forecasted customer demand and equipment availability

- provide operator with feedback on the present state of the power system (e.g. limit violations, operating percentage of limit) and, in some cases, recommend actions

This process must be performed at regular intervals or whenever the operating state of the power system has changed. The objective of the power system security assessment is to prevent:

- voltage decline (post-contingency steady-state voltage below an acceptable level)
- voltage instability (dynamic collapse of voltage at one or more points as a result of insufficient dynamic voltage support)
- plant instability (loss of synchronism of generators following contingencies)
- area instability (unacceptable levels of power oscillations associated with groups of generators)
- thermal (thermal overload of overhead conductors or transformers following loss of elements)
- short circuits (excessive fault current levels)

The current approach to power system operation has served utilities well for many years. This was due, in part, to the close coordination between the planning and operating functions. The system was operated close to its designed point. With the ongoing movement toward open access to transmission facilities, utilities recognize that conservative past practices may not be adequate resulting in potential revenue loss and unused transmission capability. Beyond the economic penalties associated with conservative operation, there is a growing concern that the integrated North American power grid may be susceptible to higher risks than initially anticipated. This increases the probability of a multiple-contingency event. In fact, the North American blackout of August 14, 2003, was caused by a number of transmission lines and power plant outages, which, in turn, caused two large power surges as power from southern Ohio tried to reach loads in northern Ohio [20]. Approximately 50 million users were left without power.

The use of forecasted equipment availability, of a selected number of operating base cases based on forecasted demand and generation dispatch, as well as the inevitable mismatch between the off-line studies and the current conditions, underline the need for an on-line security tool that would permit operators to assess system vulnerability based on real-time system conditions.

An on-line voltage stability assessment application can be thought of as a Fault Detection problem. The residual is a vector whose components are computed by taking the difference between the computed post-contingency bus voltages and the bus voltage nominal value. The Alarm Decision Block will typically select the vector component having the largest value, and decide if a voltage violation has taken place. The current off-line implementations use traditional analytical techniques such as dynamic simulations and load-flow algorithms, which are based on models of the individual power system components (transmission lines, generators, loads, reactive compensating devices, etc.). Because of being numerically intensive, these methods are best applied to power system planning tasks.

An on-line static security assessment application, implemented in the Failure Detection Systems framework, can benefit from using the so-called black-box models, as the Nominal Model Block in Figure 1-1. Black-boxes are data-driven models which do not model a physical component, but rather represent an input/output relationship established by the system which is monitored. The advantage of these models is their high computational speed. In this thesis, Neural Networks and Partial Least Squares models will be used for the static security assessment problem. Also, special care will be used to data conditioning, thus allowing a Neural Network algorithm to reliably learn on a very limited data set.

1.5.2 The Underground Power Cable Oil Leak Detection Problem

Electric power utilities around the world have a number of underground power cables that are immersed in oil. Such a system consists of pipes connected to an oil tank via an

oil pump and return valves. Oil pressure sensors control the pump(s) which turn on whenever the pressure decreases below a threshold. Once the pressure value reaches a set point, the pump turns off. If the pressure exceeds an upper limit due to the oil thermal expansion, return valves allow the oil to spill back in the tank(s). In general, such a system is reliable, oil leaks being rather infrequent. However, should an oil leak occur, this fact should be detected as soon as possible.

This case study is the description of an oil leak detection system which makes use of advanced signal processing techniques in general and Artificial Intelligence technologies in particular. Its purpose to (a) describe the design of an oil leak detection system that is very cost effective, using a minimum of data sensors, and (b) describe the steps involved in using state-of-the art Artificial Intelligence technologies in the process of solving such a practical field problem.

Because of environmental concerns, oil leaks must be detected as soon as possible. Traditionally, human operators have been assigned the task of deciding whether or not an oil leak is present. A human would examine the oil pressure record and build a mental time series model that would explain the usually periodic behaviour of the pressure. Specifically, the person would look for pump operations and decide if a pump operation is historically justified, or if it is a sign of a leak. If too many pump operations seem to be taking place, the operator checks the oil temperature record and looks for temperature drops. If such a drop takes place, the extra pump operations may be justified and an alarm may or may not be called. However, should an oil temperature increase take place during an oil leak, the pressure drop may be masked by the pressure increase caused by the oil expansion.

The use of a human operator to detect oil leaks has several disadvantages:

- it is a relatively expensive solution
- it requires a long training time, and the operator becomes an essential element in the detection system
- the task is boring and stressful for humans

- the detection is unreliable
- the experience acquired by one operator is lost once he or she stops performing the function

1.5.3 The Detection of Flow Restrictions in Water-Cooled Generator Windings Problem

General economical and environmental aspects are the main reasons for increased use of monitoring, analysis, and diagnostics systems in power plants [21]. Such FDS are employed to prevent or reduce down time, to enhance operating reliability and plant availability, and to extend the service life of equipment. The main goal of a FDS is to achieve early recognition of progressive miss-operation, so that corrective action can be taken thus preventing permanent damage.

The stator windings of modern high-capability generators are water-cooled [22]. All, or only a few strands of a stator bar are hollow and passed by pure water. In a generator stator for a nuclear plant, water pressurization is provided by air, nitrogen or hydrogen [23]. General knowledge of copper behaviour in pure water is very limited and does not provide an understanding of the mechanisms involved in the partial obstruction of hollow conductors [23]. It is imperative to monitor the temperature of the generator stator bars and to alert the operator of any abnormal higher values present for an unreasonably long period of time perhaps signaling the onset of blockage.

A number of such systems have been described in the published literature [22], [24]. Both systems rely on determining the stator bar temperatures based on physical factors such as: stator electric current, coolant temperature, and machine parameters. While such physical models can perform well, if they are sufficiently detailed, they have two major downsides:

- The model will be only as good as the designer's knowledge about the physical phenomena that take place in the machine.
- The model will be highly customized to a particular machine. A system for a different generator will have to be hand crafted again.

We propose a different approach that will overcome the above problems by exploiting the high correlation between the temperature values amongst the different stator bars. We will prove that this method has several advantages:

- It is resilient to sensor breakdowns
- A strongly nonlinear model is replaced by an almost linear one, decreasing learning time
- This approach is self-adapting being applicable to other generator installations

1.6 Contributions

1. A clustering technique using the Partial Least Squares (PLS) approach is used to group contingencies in an Electric Power Systems. This is the first known usage of PLS in an electrical application. Previous applications of PLS have been in Chemical Engineering for process control. The clustering technique is applicable to any modeling problem where rather than using one input-output model for the whole input set, a number of models are constructed for parts of the input set. It will be shown that the novelty of the approach lies in the fact that the clustering process is based on a set of models which are themselves build on a certain clustering level. The process is iterative mimicking the approach a human expert would use in solving this clustering task. The technique was presented in the paper "Automatic Contingency Grouping Using Partial Least Squares and Feed Forward Neural Network Technologies Applied to The Static Security Assessment Problem" at The Large Engineering Systems Conference on Power Engineering, (D. Fischer, B. Szabados, and W.F.S. Poehlman , pp. 84–89, May 7-9, 2003)

2. A Neural Network based model capable of estimating the future state of a power system following a contingency is developed. While such a network has previously been reported in the literature, data preprocessing (Principal Component Analysis - PCA, and PLS) results in a greatly reduced required learning data set and a shorter learning time, also improving generalization.
3. A number of new architectures for an Oil Leak Detection System in Underground Electric Power Cables are proposed. The proposed architectures have several advantages over other possible approaches such as physical modeling. The proposed architectures exhibit adaptive behaviour and sensor analytic redundancy.
4. For the first time, an Artificial Intelligence (AI) based system is developed for the oil leak detection application. The results from tests using real field data and the lessons learned from this implementation are presented. AI is used to perform feature extraction, modeling, as well as soft voting.
5. Custom data preprocessing leads to the possibility of a linear model to handle a strongly non-linear task. This leads to a vastly shortened learning time.
6. While a classical Neural Network supervised learning method is used to build a model in the Oil Leak Detector test case, no learning data set (a teacher) is available. A new approach shows how such a teacher can be constructed while learning takes place. The method is iterative and builds the model while constantly adjusting the training data set. It is similar to a student learning from a teacher, who then uses the student to fill in knowledge gaps, advancing the teacher's knowledge. This method provides, as a benefit, an elegant, generic, process for handling missing or corrupted data.
7. A new approach to the Electric Power Generator Stator Blockage Detection problem illustrates a way to exploit high input data correlations in order to achieve:
    - high resilience to hardware breakdowns
    - an almost linear input / output relationship resulting in fast learning

The method is described in the paper "Combining Partial Least Squares and Feed Forward Neural Network Technologies in a Fault Detection System with Large Number of Correlated Sensors" and was presented at The 19th IEEE Instrumentation and Measurement Technology Conference (D. Fischer, B. Szabados, and W.F.S. PoehIman, vol.1, pp. 829-834, 21-23 May 2002)

8. All Fault Detection Systems presented in this thesis use a Bayes classifier responsible for deciding whether or not an alarm should be initiated. A new, on-line, adaptive, extension to the classical Bayes classifier is presented. The method was published in the paper "Using a Bayes Classifier to Optimize Alarm Generation to Electric Power Generator Stator Overheating", presented at the 2002 IEEE International Symposium on Virtual and Intelligent Measurement Systems, and subsequently published in the IEEE Transactions on Instrumentation and Measurement (D. Fischer, B. Szabados, and W.F.S. PoehIman, vol. 52, no. 3, pp. 703–709, June 2003)
9. All Fault Detection Systems presented in this thesis have to deal with the lack of data showing a system fault. They must learn using data produced by correctly operating systems but recognize when inconsistent data, produced by a fault, is presented. Two new Fuzzy Logic based implementations show how statistics of failed systems data can be generated using expert knowledge.

1.7 Organization of the Thesis

As described in Chapter 1.1, a generic Fault Detection System makes use of two major techniques: a modeling algorithm (Nominal Model block, Estimated Operating Model block) and a classification algorithm (Decision Block). Chapter 2 will discuss different modeling algorithms implemented as a black-box. These algorithms build data-based models having either linear or non-linear properties. Issues such as the effect of noise on model parameters, learning and recall abilities, are discussed. The Artificial Intelligence based approaches (Neural Networks, Fuzzy Logic) are compared in performance with the classical techniques.

Chapter 3 describes a variety of classifier designs. The performance of a design is a result of the classifier complexity. The optimum design is the Bayes classifier. This algorithm can minimize the probability of a classification error, or the cost of a classification error. We present a new, on-line, real-time extension of this technique, which adapts to the data it uses. We will also present a novel way for a fuzzy logic algorithm to produce statistical features needed by the Bayes classifier.

Chapter 4 contains three case studies: the static security assessment problem, the oil leak detector in underground electric power cables, and the detector of flow restrictions in water-cooled generator windings. These case studies are classes of Fault Detection Systems. Each implementation will be based on the general FDS framework introduced in Chapter 1, and will use a mixture of Artificial Intelligence as well as classical techniques. Finally, Chapter 5 will list the advantages and disadvantages of the different methods used and will present the conclusions along with topics for further research.

CHAPTER II

2 Parametric Models

The previous chapter has introduced the problem domain as well as briefly analyzed the input and output data. One crucial building block in a fault detection system is the black box model. Such a computational entity establishes an input/output relationship between presented input and output data. Once the relation is determined during learning, the block is used on new input data to predict the value of the output.

Several types of empirical models will be briefly introduced. The linear transfer function model coupled with a noise stochastic model will first be presented. A solid design methodology has been developed over time and is presented in numerous time series analysis and system identification treatments [25], [26], [27], [28]. If a linear model is not adequate for the problem at hand, a neural network may prove beneficial. Two supervised topologies are shown, highlighting the advantages and disadvantages they present over the established linear approach. Fuzzy logic provides another modeling approach. A discussion of Kalman filtering will show another modeling approach which is beneficial when the state transition matrix of the system and noise statistics are known. If a large number of redundant information is available, the Partial Least Squares technique may come to the rescue. Advantages and disadvantages of each technique will be highlighted.

2.1 Stochastic Linear Model

Models describe relationships between measured signals. Figure 2-1 shows an output signal y which is partially determined by the input signal u. In order to account for the

mismatch between the output of the model and the actual measured signal y, a disturbance signal, e is also included.

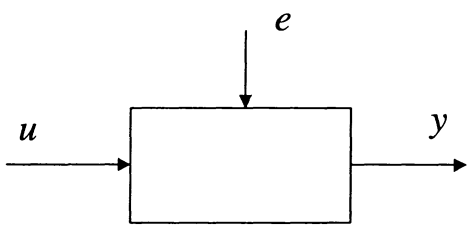


Figure 2-1 Input signal u, output signal y, and disturbance e.

The disturbance represents noise which may be present in the measured output y, input u, or unmeasured inputs present at any part of the physical system. We consider the discrete signal case where u, y, and e, are sequences of number rather than continuous functions of time.

The general linear models can be described symbolically by equation (2-1):

$$y = Gu + He \tag{2-1}$$

where G is the process transfer function, H is the disturbance transfer function, and e is white noise.

The relationship between y and u can be expressed as a linear difference equation:

$$y(t)+a_1 y(t-1)+\ldots+a_n y(t-n)=b_1 u(t-1)+b_2 y(t-2)+\ldots+b_m y(t-m) \tag{2-2}$$

or

$$\sum_{i=0}^{n} a_i y(t-i) = \sum_{i=1}^{m} b_i u(t-i) \tag{2-3}$$

If we attach polynomials having coefficients a_i and b_i, to the sequences a_i and b_i, then (2-3) becomes:

$$A(q)\,y(t) = B(q)u(t) \tag{2-4}$$

Variable q is called the time shift, or delay, operator. The power of q indicates the number of delays the operator performs. The transfer function G is now expressed as a ratio of two polynomials.

A commonly used parametric model is the ARX (auto regressive with exogenous inputs):

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{1}{A(q)}e(t) \tag{2-5}$$

Therefore, the process and disturbance transfer functions in equation (2-1) are:

$$G(q) = \frac{B(q)}{A(q)}$$

$$H(q) = \frac{1}{A(q)} \tag{2-6}$$

In general, systems can be modeled as:

$$A(q)\,y(t) = \frac{B(q)}{F(q)}q^k u(t) + \frac{C(q)}{D(q)}e(t) \tag{2-7}$$

The model (2-7) is called the Box-Jenkins (BJ) model structure, and allows for maximum flexibility. The term q^k in the process term, allows for k delay intervals between the output y and input u.

If the noise transfer function is equal to one, and the polynomial $A(q)$ is of order zero, then:

$$y(t) = \frac{B(q)}{F(q)} q^k u(t) + e(t) \tag{2-7}$$

Equation (2-7) is the so-called Output Error (OE) model. The input $u(t)$ drives the process transfer function whose output is finally affected by white noise.

In order to derive a model that accurately describes a phenomenon, a number of steps must be performed [25], [26], [27]:

1. A process and disturbance structure must be estimated. The powers of the polynomial *A, B, C, D,* and *F* are to be estimated.
2. Given the power, the polynomial coefficients can be estimated
3. Tests are performed [27] to determine if the process and noise structures are appropriate.
4. Usually the model is tested on a validation data set.

The strength of the procedure is derived from the systematic approach from which the parameters are derived [29], [30], [31]. Its weakness could arise from the fact that it uses only linear models. If non-linearities are to be accounted for, one approach is the use of Neural Networks.

2.2 *Neural Network Model*

Referred to also as connectionist architectures, parallel distributed processing systems, an artificial neural network (ANN) is an information processing approach suggested by the way the densely interconnected, parallel structure of the mammal brain processes information. ANNs are a mathematical model that emulates some of the observed properties of the biological nervous systems, one of the most important being the ability to learn from examples. Biological systems learn by adjusting their synaptic connections between neurons. ANNs also learn by adjusting parameters, called weights, that are associated with links between the artificial neurons.

There are many types of ANNs [31], [32], [33], [34], [35], [36], [37]. One major way of classifying them is based on their learning methodology. The networks that learn when presented with a solution by a teacher, use supervised learning. The ones that learn without a teacher, use unsupervised learning.

Two of the very useful Neural Networks structures successfully applied to modeling tasks are the feed forward network taught with the back propagation (BP) algorithm and the radial bases functions (RBF) network. The next sections will briefly present them. One philosophical comment should be made regarding the BP and the RBF networks. Both networks are very useful in solving practical problems. However, it is generally acknowledged that the brain does not have a teacher for learning, and if there is a teacher for some learned tasks, the feedback is usually rather vague and is provided after a long sequence of actions have been controlled by the brain. Learning how to ride a bicycle is one such a case, where there is virtually no teacher, the instructions are sparse, and the feedback is usually provided by a fall followed by pain with no instruction for the reason for failure. The BP and RBF have a fundamentally different learning approach where each network output is characterized by an exact error measure. The BP and RBF are useful, but have little biological support.

2.2.1 Feed Forward Networks with Back Propagation Learning

Figure 2-2 shows the structure of a feed forward back propagation network. An I dimensional vector whose components are real numbers is presented at the input of the network. The input layer performs no computation and is strictly to latch the input and pass it on to the next layer, the hidden layer. Computations are performed by the hidden and output layer nodes.

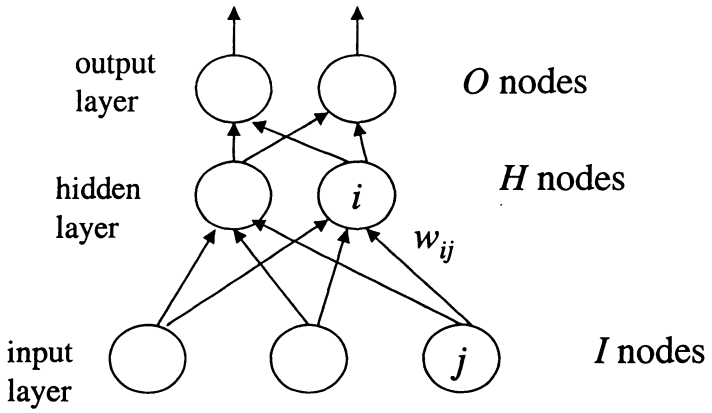


Figure 2-2 Feed forward back propagation structure.

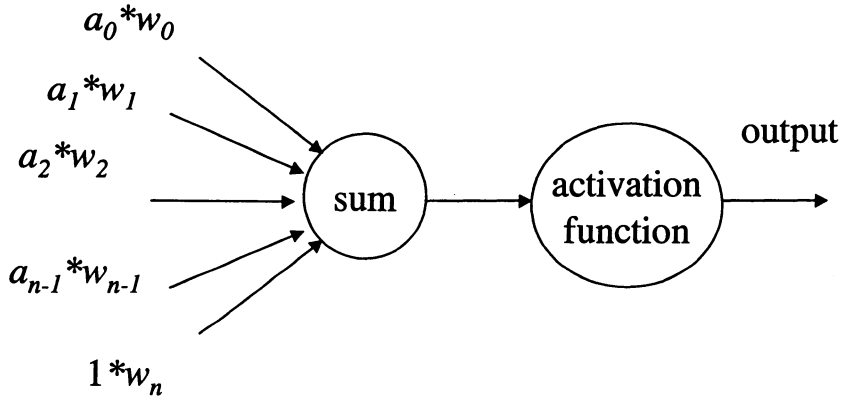


Figure 2-3 Back propagation neuron.

Figure 2-3 depicts the structure of a single node. The input values a_0, a_1, a_2, ... a_{n-1}, are the n inputs the node must process. Each value is multiplied by a weight, w_{ij} and then passed on to a summing element. The sum of the weighted inputs is applied to a transfer function that produces the output of the node. An $(n+1)^{th}$ input of fixed value, 1, is also weighted by a weight, w_n. This input is called the bias and allows the weighted input sum to be shifted before being passed through the transfer function.

The transfer function must be monotonically increasing. A common choice is the hyperbolic function shown in Figure 2-4. As it can be seen, while the absolute value of the weighted input sum is small, smaller than 0.5, the node behaves almost linearly. For non-linear behaviour, the node can appropriately increase the value of its weights.

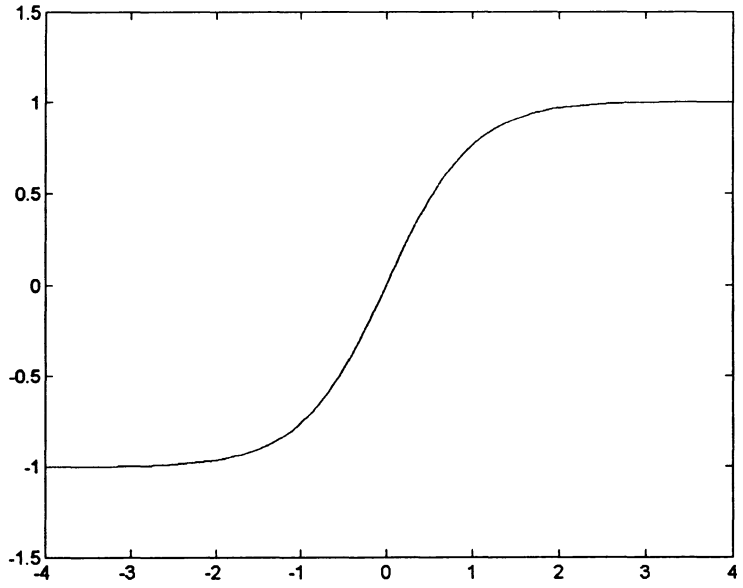


Figure 2-4 Hyperbolic tangent activation function.

During learning, the network is presented with a number of input and output vectors. Initially, the network's weights are randomly initialized with values that are typically

small. The result is that the network produces an output to an input pattern which is different from the desired target. The difference between the output and the target becomes the network's error. The learning process consists of applying an algorithm which decides the way the weights must be adjusted in order to reduce the total output error. The output error is propagated backwards, from the output nodes towards the inputs, hence the name, back propagation given to the learning procedure. One name for the error propagation, is the credit assignment problem: it is the assignment of "blame" to the different weights responsible for producing an output which contains an error. The details of the back propagation algorithm are described in many published works [33] and will not be repeated here. A few comments will be made:

1. The algorithm is iterative, therefore has the potential to be time consuming. There are many methods applied to speed it up.
2. The sensitivity of the error with respect to a weight depends on the derivative of the node output to its input. Therefore, if a node output is saturated by a large sum of its weighted inputs, the derivative of the output will be very small. The implication is that the weights that are feeding such a node will change very slowly. It is said that a saturated node learns very slowly. This is the reason for initializing a network's nodes with small values in order to allow them to learn.
3. In order to avoid saturating the hidden nodes, the input data set must be scaled appropriately. It is better to have input values that are too small than inputs that are too large. The network will learn quickly if it has to increase its input weights to compensate for small inputs. It may take longer for the network to be able to reduce the values of the input weights, once it is saturated.

Once the learning step is completed, and the network's weights are determined, the network can be used for recall. A new input vector will cause a feed forward computation to take place and an output to be computed.

2.2.2 Radial Basis Function Networks

Radial Basis Function (RBF) networks represent another major class of neural network models. The activation of an RBF hidden unit is determined by the distance between the input vector and a prototype vector associated to the hidden unit [33], [34], [35], [36], [37], [38].

Figure 2-5 shows the structure of the network. An input vector of dimension n is applied to the hidden layer composed of r nodes. Each node has a n dimensional prototype associated with it. It computes the Euclidian distance between its prototype and the input vector. The value of the distance is passed through a transfer function which has a maximum of one when the distance is zero and decays with the increase of the distance.

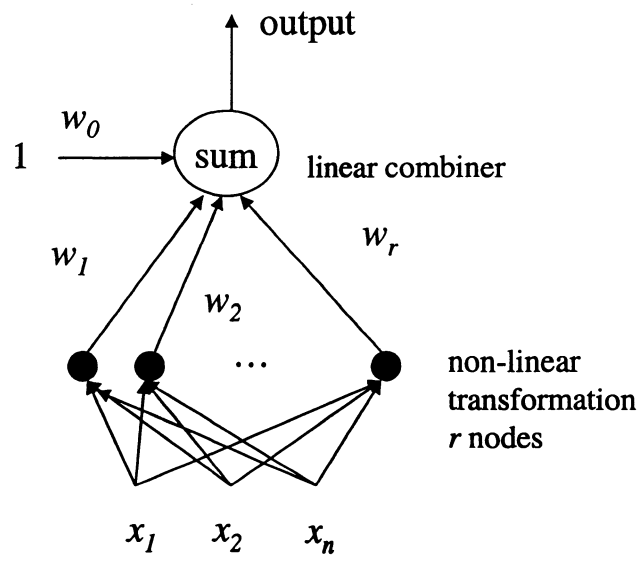


Figure 2-5 RBF network.

An example of such a transfer function is:

$$\phi_j(\underline{x}) = e^{-\frac{\|\underline{x}-\underline{c}_j\|^2}{\beta^2}} \qquad 1 \le j \le r \tag{2-8}$$

where $\underline{x}$ is the n dimensional input vector, and $\underline{c_j}$ is the prototype vector of dimension n associated with the node j while β is just a parameter that sets the decay rate of ϕ with the distance of the input into the node to its prototype.

The prototype nodes are chosen based on the input training data. Chen [38] shows how to chose these prototypes in an optimum sense. The linear combiner computes the weights w in such a way as to be able to approximate the data set.

The RBF network acts as an interpolator. The advantage it has over the BP is in the higher learning speed.

2.3 Fuzzy Logic Model

This chapter will briefly introduce the need for Fuzzy Logic followed by the description of the steps used to convert a set of inputs applied to a Fuzzy Logic block, into a number of outputs. Two implementation approaches are outlined: the Rule Based Model, and the Table Based Model.

2.3.1 Introduction to the Fuzzy Logic Paradigm

The term "fuzzy logic" has evolved from the work on the theory of fuzzy sets done by Lotfi Zadeh [39]. As Zadeh explained [40], his papers were motivated by the conviction that traditional methods of systems analysis were unsuited for dealing with systems in which relations between variables do not lend themselves to representations in terms of differential or difference equations, systems present in biology, sociology, and economics. Zadeh was concerned with the effect system complexity has on one's ability to explain system behaviour:

> "As the complexity of a system increases, our ability to make precise and yet significant statements about its behaviour diminishes until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics." (Lofti Zadeh, Outline of a New Approach to the Analysis of Complex Systems and Decision Processes [40])

In order to deal with these issues, Fuzzy Logic is built as an extension to Boolean logic. Variables, which in classical logic have values attached to, are now assigned labels, or fuzzy sets. Fuzzy sets are functions that map a value to a number between zero and one, indicating its actual degree of membership. A variable can be, and is, attached to several fuzzy sets. For instance, a temperature can be simultaneously high, medium, and low, with different degrees of membership. In the case of Boolean logic, a temperature would be considered high if its value is above a certain threshold, medium if its value is inside a range, and low if its value is lower than another threshold. Boolean logic requires for one and only one label to be associated with the temperature variable, and based on the label, an algorithm would perform its computations.

Fuzzy logic postpones the choices that must be made, as much as possible. An algorithm will perform its logic as if the temperature is high, and medium, and low, and then reconcile possibly conflicting conclusions, at the very last point of the inference.

The mapping from a specific variable value (crisp value) into a fuzzy set is called fuzzification. The computational algorithm typically consists of a number of "if" statements. The left hand side of the "then" in the statement, is called the antecedent. The right hand side of the "then" is called the consequent. Antecedents use fuzzy sets and the "AND" or "OR" conjunctions. There are a large number of ways "AND" and "OR" can be implemented in order to compute the value of the antecedent. Examples for the implementation of "AND" are the "minimum" operator and the "product" operator. "OR" is many times implemented using the "maximum" operator.

Once the antecedent value is determined, the implication will determine the value of the consequent. The implication method results in the shaping of the consequent, which is itself a fuzzy set. The implication can be supported using the "minimum" operator or "product" operator. The minimum operator would use the value obtained for the antecedent, and compute the minimum between this value and the fuzzy set of the consequent. Once the consequent is computed based on the implication, a weight, whose value is between 0 and 1, also multiplies the consequent. This models the fact that some rules may be considered more important than others in determining the output of the fuzzy logic process.

The implication is performed for each rule, or "if" statement. In order to combine different fuzzy sets that refer to the same output (eg. the value of the pressure is "high", the value is "medium", and the value is "low"), rules must be aggregated. This can be done in many ways. One such way, is by overlaying the output fuzzy sets on each other after they have been affected by implications, and computing the maximum value of the union. This approach is called the "min-max" inference ("min" used for implication, "max" is used for aggregation)

Finally, defuzzification converts the output, which is a function, into a real number. One way to achieve this, is by computing the center of gravity of the area below the function which represents the output fuzzy set.

The inference mechanism described above is called the Mamdani inference, a method which is intuitive, accepted, and well-suited to human input [41], [42], [43], [44], [45], [46].

An alternative approach to implementing a fuzzy output for each rule is the Takagi-Sugeno model. This model defines the consequent of each rule as a linear combination of the crisp inputs of the antecedent of the rule modulated by the membership function value for the crisp input:

Rule$_i$: If $\underline{x}$ is A_i then $y_i = \underline{a_i}^T \underline{x} + b_i$, $i=1,2,\ldots,K$

$\underline{x}$	...crisp (real) variables, input vector of antecedent for rule "i"
y_i	...crisp (real) variable
a_i, b_i,	...crisp (real) parameters
A_i	...linguistic terms (fuzzy sets)
K	...number of rules

(2-9)

The Takagi-Sugeno inference is computed as:

$$y = \frac{\sum_{i=1}^{K} \mu_{Ai}(x)\, y_i}{\sum_{i=1}^{K} \mu_{Ai}(x)} = \frac{\sum_{i=1}^{K} \mu_{Ai}(x)\left(\underline{a_i}^T \underline{x} + b_i\right)}{\sum_{i=1}^{K} \mu_{Ai}(x)} \qquad (2\text{-}10)$$

where μ_{Ai} is the implication value of the antecedent of rule i.

As can be seen from equation (2-10), the output of the Takagi-Sugeno model is a piece-wise linear function in the input $\underline{x}$. Therefore, this model is expected to have the following features:

- computationally efficient
- works well with linear techniques (PID control)
- works well with optimization and adaptive techniques
- guaranteed continuity of the output surface
- well-suited to mathematical analysis

There are several potential advantages of the fuzzy logic approach, compared to a Boolean implementation:

1. by converting the crisp inputs into fuzzy sets, the algorithm (the "if" rules) can concentrate on handling higher level methods rather than also having to deal with the details of the variable values.
2. rather than having to commit to categorizing an input to a certain part of the algorithm (for instance, once the temperature is considered "high" a certain part of the algorithm is functional, perhaps differently from the case if the temperature was "low"), the whole algorithm is exercised.

2.3.2 Fuzzy Logic Implementation Techniques. Rule Based and Table Based Models

Now that the fundamentals of Fuzzy Logic models have been introduced, a few implementation choices must be discussed. Many controllers are constructed using the methodology described above. At run time, inputs are fuzzified, and rule antecedents are computed one by one, followed by the computation of rule implication, aggregation, and finally output defuzzification. As can be expected, the above procedure is rather processor intensive. Its advantage is that it is memory efficient. An alternative approach is to encode the transfer function of the Fuzzy Logic controller in a lookup table. The inputs are digitized into intervals, and the output of the controller is computed for all input combinations, for all intervals. A potentially large table, of the order of $O(F^N)$, where F is the number of fuzzy sets for each input, and N is the number of inputs, is stored in a lookup memory. At run time, the controller may choose to linearly interpolate a number of table results, depending on the actual vector input value. The result is a much shorter computation time, compared to the rule based approach. This is the method of choice for demanding real-time applications. Another advantage of this implementation is that the fuzzy logic table can be computed off-line using a user-friendly fuzzy logic tool, which is not needed, and may not be available at run-time.

2.4 *Kalman Filtering*

A Kalman filter is a recursive, linear, optimal, real-time data processing algorithm used to estimate the states of a dynamic system in a noisy environment [47].

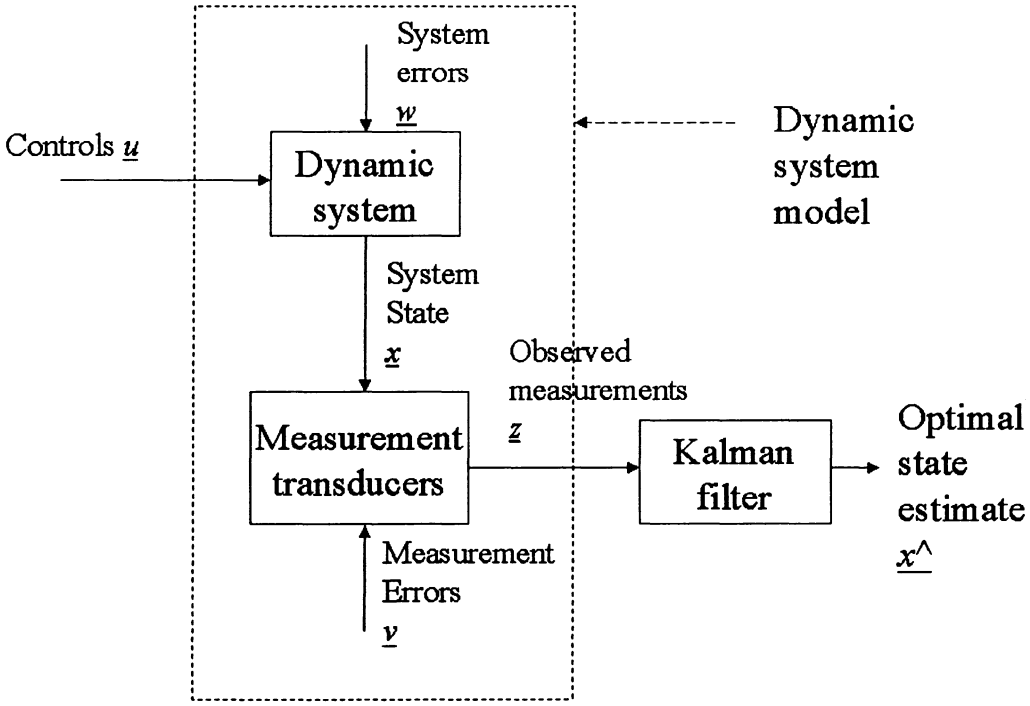


Figure 2-6 Kalman filter block diagram.

Figure 2-6 shows a dynamic system controlled by a known vector $\underline{u}$. The mathematical description of the model, the vector $\underline{u}$, and the observed measurements $\underline{z}$ are the only information used for estimation purposes. The quantity to be estimated is the system state $\underline{x}$.

The general form of the Kalman state estimation equation is [47]:

$$\hat{\underline{x}}(k+1) = \underline{A}(k) * \hat{\underline{x}}(k) + \underline{B}(k) * \underline{z}(k+1) \qquad (2\text{-}11)$$

where:

$\hat{\underline{x}}(k+1)$ is the Kalman filter estimate at latest time k+1

$\underline{A}(k)$ time varying matrix gains

$\underline{B}(k)$

$\underline{z}(k+1)$ measurement vector

The Kalman filter is:

- *Recursive*: based on equation (2-11), the new state can be estimated based on the previous state and the latest measurement
- *Linear*: property seen in equation (2-11)
- *Optimal*: matrices $\underline{A}(k)$ and $\underline{B}(k)$ are calculated at each cycle by the Kalman filter in such a way as to minimize the variance of the state estimation error.
- *Real-time*: ideally suited to real-time applications

In order to be able to use the Kalman design, three assumptions must be made with respect to the plant and measurement:

1. the system model is assumed to be linear
2. all noise sources are white
3. all noise sources are Gaussian.

Equation (2-11) contains the two time-varying matrices, $\underline{A(k)}$ and $\underline{B(k)}$ which refer to the expected system state transition as well as the measurement process. This equation can be broken down into the Kalman filter steps:

$$\underline{x}(k+1) = \Phi\,\underline{x}(k) + \Omega\underline{u}(k) + \underline{w}(k) \qquad (2\text{-}12)$$

$$\underline{z}(k) = \underline{H}\,\underline{x}(k) + \underline{v}(k) \qquad (2\text{-}13)$$

$$\hat{\underline{x}}(k) = \hat{\underline{x}}^{-}(k) + K(\underline{z}(k) - H\,\hat{\underline{x}}^{-}(k)) \qquad (2\text{-}14)$$

Equation (2-12) describes the system state evolution in time, under the effect of the controlling input $\underline{u}(k)$. It must be known (or estimated from data, as in the previous Linear Stochastic Model chapter). Equation (2-13) describes the measuring process. If the measured quantities are the system state components, then matrix $\underline{H}$ is simply a diagonal identity matrix with the dimension equal to the number of sensors. Vectors $\underline{w}$ and $\underline{v}$ represent noise in the state transition and measurement process. They are assumed to be independent of each other, white, and with normal probability distributions.

Equation (2.14) is the Kalman equation. $\underline{\hat{x}}^-$ is the system state computed by (2-12) and the "-" sign refers to the fact that it is computed for the time right before a new measurement $\underline{z}(k)$ is available. K is called the Kalman gain and is re-evaluated at each step. Its formula is given in all texts describing the Kalman filter [48].

The Kalman equation states that the new state estimate is a mixture between an a priori estimate (based on a model only, hence the term "a priori") and a difference between the latest measurement and an a priori forecast of the measurement (this difference is called the innovation). Depending on the noise level of the state transition and measurement processes, the Kalman gain puts more emphasis on the transition equation or on measurement, in order to minimize the variance of the residual computed as the difference between the estimated state and the true system state.

A number of papers [49], [50], [51] discuss the use of Kalman filters in FDS applications. The purpose of the filter is to model the process in order to detect a discrepancy between the measured process state and the estimated one.

It is interesting to note that this discrepancy is exactly the Kalman filter innovation. As Tylee notes in [52], a fundamental property of the Kalman filter is that if the physical system actually evolves according to the state and measurement equations (2-12) and (2-13), the filter will generate a innovation vector $\delta^I(k)$ that is zero-mean and white. In other words, the Kalman filter will adjust its gain in order to statistically track the process

state and produce a zero mean error. The result of this behaviour, is that once a fault occurs and the state of the system changes, the Kalman filter will start absorbing this change, to the point of producing estimates which again match the modeled system. The Kalman filter has just masked the fault and has treated it as if the process was the cause of the state shift, and not the fault.

If one uses the innovation to trigger an alarm block, depending on how quickly the Kalman filter will adapt to the fault, the innovation caused by the fault will be a spike which may not be obvious compared to the innovation values during normal filter operation.

We found that by taking the integral of the innovation, we can "lock in" the fact that a fault has taken place. Of course, by integrating a noisy innovation (residual), the standard deviation of the integral increases in time, resulting in an input into the alarm block which could increase or decrease in time, being driven strictly by noise. In order to eliminate this problem, the residuals to be integrated should be windowed. This is essentially what Sohlberg does in [53]. For the one-dimensional case, Sohlberg's approach results in a rectangular window by which the residuals are modulated. The length of the window is governed by the fail to trip (too narrow window) or false to trip (window too long results in too much noise being integrated) tradeoff.

In order to demonstrate the behaviour of the innovation signal in the presence of a fault, we have simulated the case of a scale which measures a weight of 173 lb and which develops a fault at time point 50. Figure 2-7 shows the scale measurement, the Kalman estimate (the measurement is the state to be estimated), as well as an indictor for the moment the fault occurs. Figure 2-8 shows the variation of the Kalman gain which settles to a value of 0.5, taking into account that the declared state modeling and measurement noises are equal in variance.

As it can be seen in Figure 2-7, the Kalman estimate quickly tracks the faulted measurement, thus masking the fault.

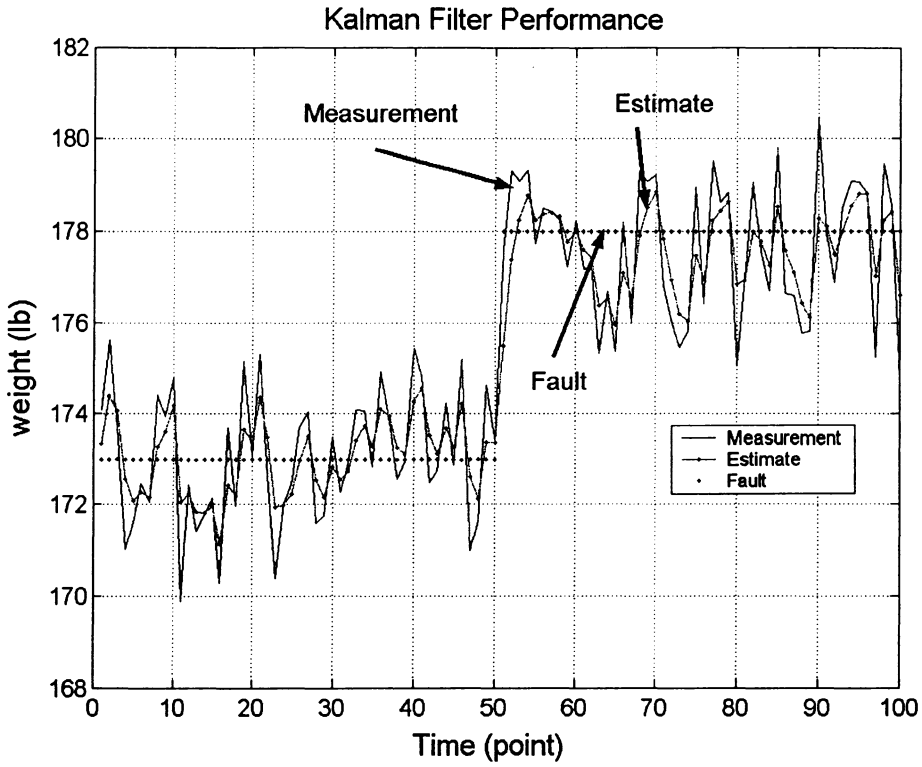


Figure 2-7 Measurement and Kalman estimate with a step fault.

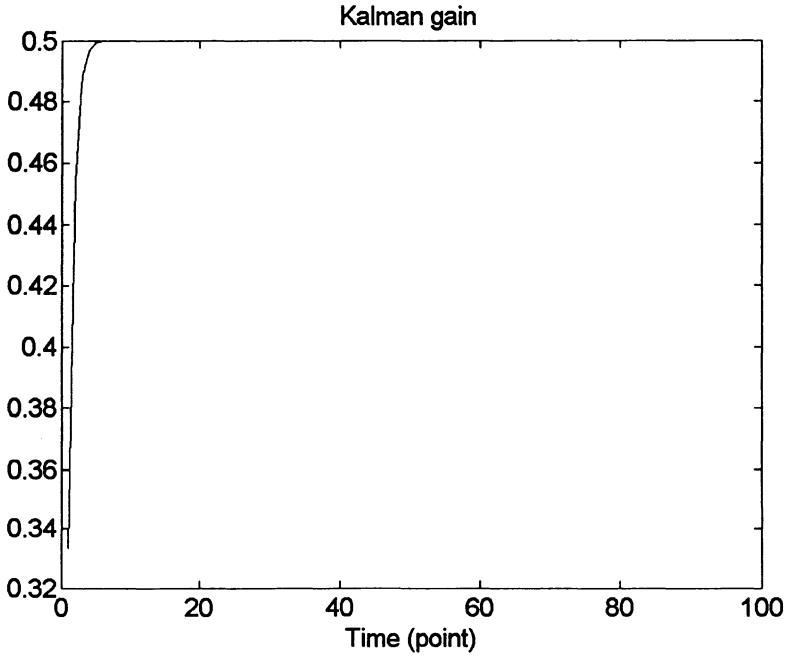


Figure 2-8 Kalman gain. Process noise equals measurement noise.

This can be also seen in Figure 2-9 where the value of the innovation during and after the fault is hardly different than the one before the fault.

Figure 2-10 shows our proposal: rather than using the innovation as a fault indicator, we use the innovation's integral. This quantity could easily be triggered on by an alarm block used by a Fault Detection System.

Figure 2-11 and Figure 2-12 show the behaviour of the innovation integral in the more realistic case of a fault which develops gradually.

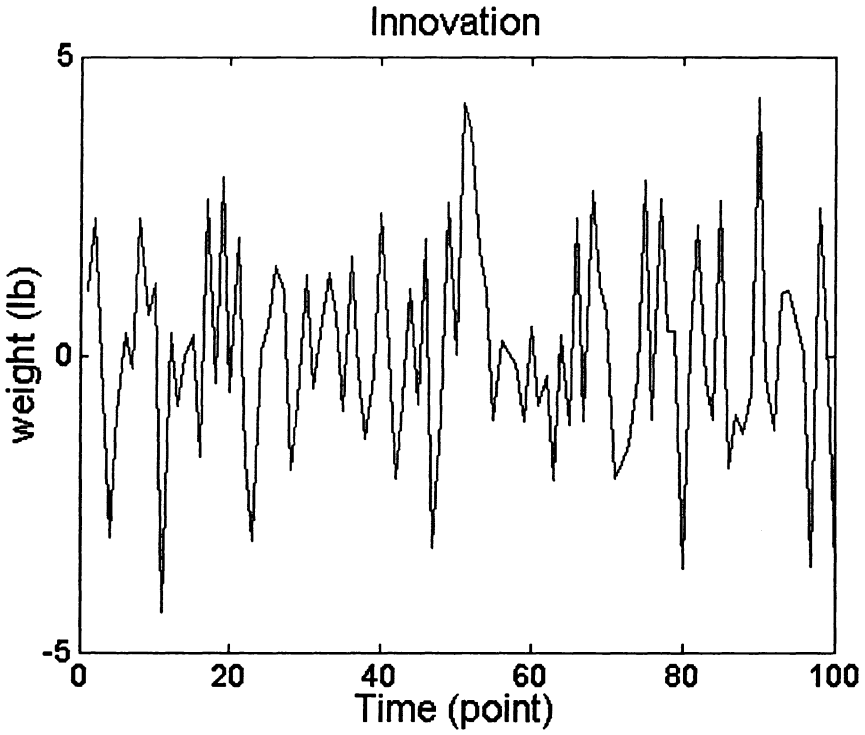


Figure 2-9 Innovation for step fault at time T=50.

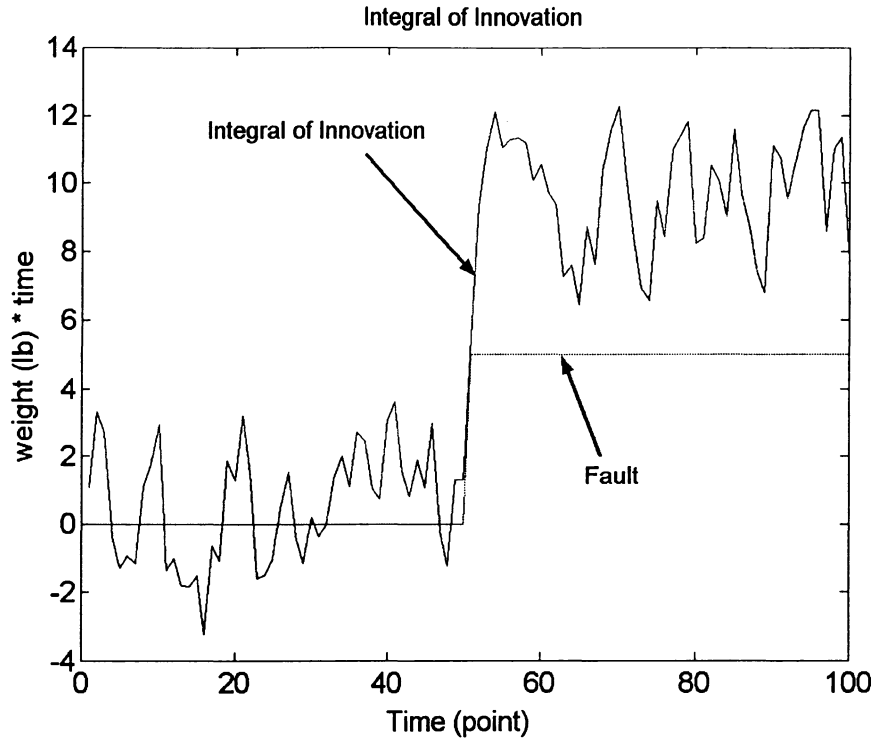


Figure 2-10 Integral of innovation for step fault.

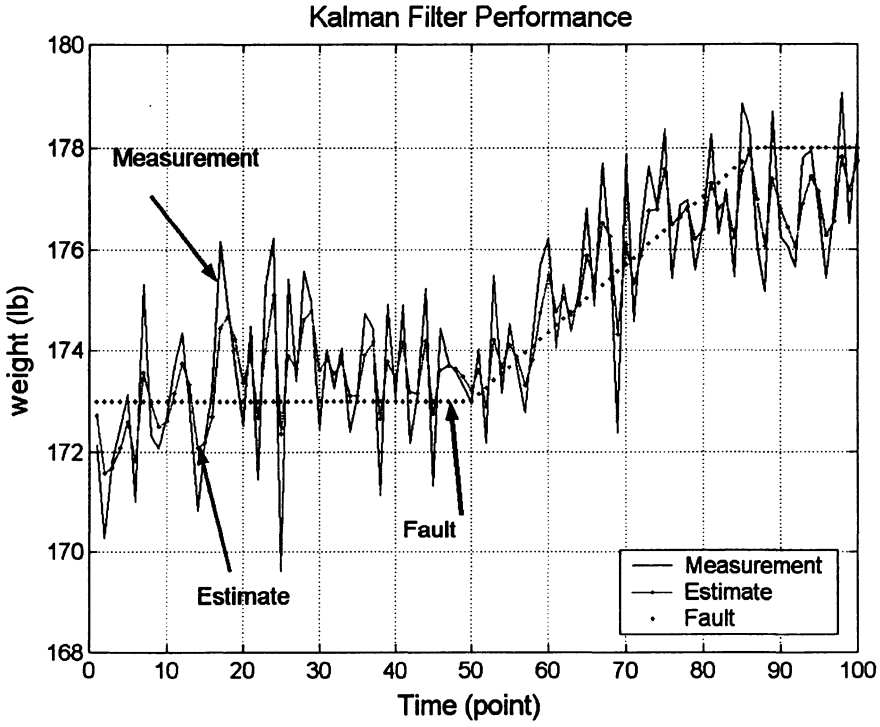


Figure 2-11 Measurement and Kalman estimate for ramp fault.

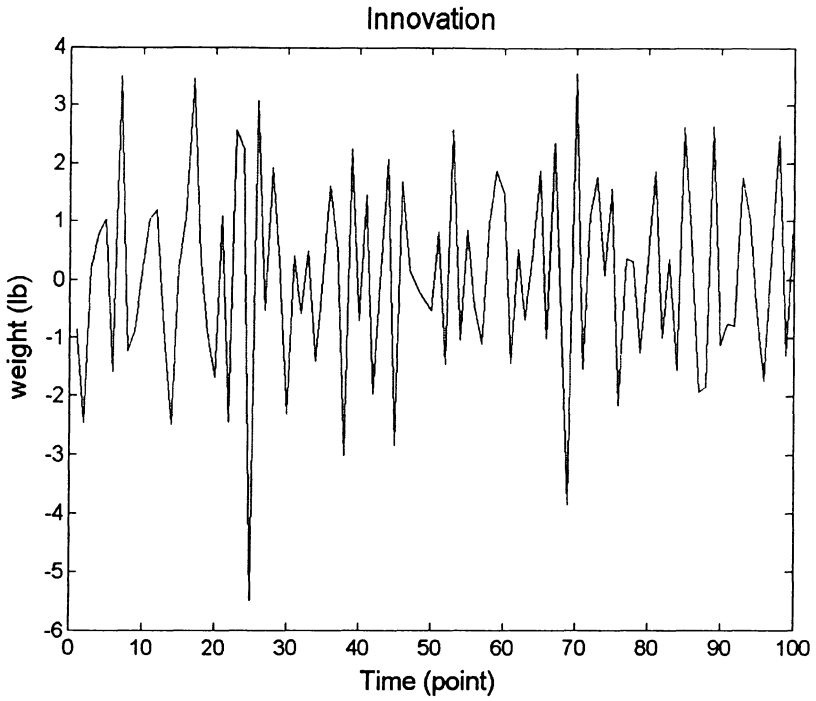


Figure 2-12 Innovation for ramp fault starting at T=50.

As expected, the innovation signal cannot mark the location of the fault. However, its integral value can, as shown in Figure 2-13.

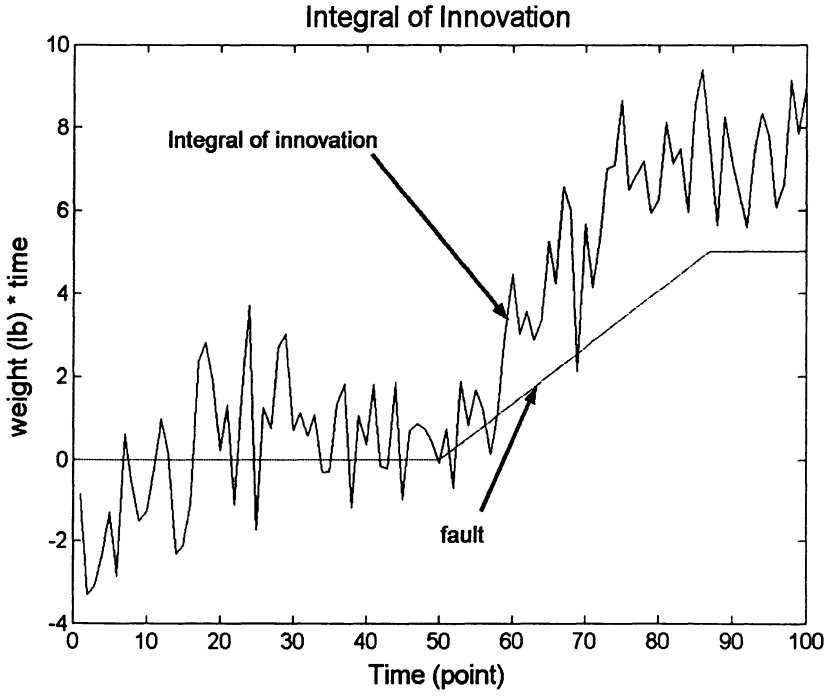


Figure 2-13 Integral of innovation for ramp fault starting at T=50.

2.5 *Partial Least Squares*

Partial Least Squares regression is a technique useful for building a linear model that can predict a set of dependent variables $\underline{Y}$ from a very large set of potentially correlated, independent variables $\underline{X}$ [54]. Matrix $\underline{X}$ contains the independent variables, where each column represents measurements from a specific sensor, while each row represents an observation, a set of measurements taken at one point in time. Similarly, matrix $\underline{Y}$, contains dependent variables, each variable being stored in a column, with each row representing a dependent observation.

When the number of predictors is large compared to the number of observations, $\underline{X}$ is likely to be singular and a regression approach, which could be used to relate $\underline{X}$ to the dependent $\underline{Y}$, is no longer feasible. One approach to deal with this problem is the use of Principal Component Analysis (PCA). PCA examines the matrix $\underline{X}$, and replaces the correlated independent variables with a smaller in number, set of uncorrelated variables. These variables are linear combinations of the original ones and are determined based on the covariance matrix of $\underline{X}$. The orthogonality of the principal components (the eigenvectors of the covariance matrix of $\underline{X}$) eliminates the colinearity problem, however the choice of principal components PCR performs, is sub-optimum from a modeling point of view. PCR examines $\underline{X}$, but not $\underline{Y}$.

In contrast, Partial Least Squares (PLS) finds components from $\underline{X}$ which are also relevant to $\underline{Y}$. The PLS regression searches for a set of directions, called latent vectors, that perform simultaneous decompositions of $\underline{X}$ and $\underline{Y}$ with the constraint that these components explain as much as possible of the covariance between $\underline{X}$ and $\underline{Y}$. The PLS algorithm is available in a number of publications in chemometrics applications [55], [56], [57], [58], [59], [60], [54], [61], and will not be repeated here. Two Fault Detection System applications presented in the following chapters make extensive use of PCA and PLS.

CHAPTER III

3 Classifier Design and Alarm Generation

Figure 1-1 indicates that a FDS must be able to perform two major computations: modeling and decision making. This chapter will briefly introduce classifiers, the algorithms responsible for deciding if the vector output of the Residual Generator block in Figure 1-1 indicates a healthy system, or if a failure alarm should be issued.

First, the distance-based classifiers are described, followed by the minimum intra-class distance classification. The maximum a-posteriori probability (Bayes classifier) algorithm is briefly described. This decision making process has the advantage of being optimal, in the sense of minimizing the probability of making a mistake.

Main ingredients for the Bayes classifier are the probability density functions of the classes the classifier must categorize the input data into. In our FDS case, these classes are: system OK class, and system FAILED class. We will introduce classical methods for density estimation, followed by a novel, Fuzzy Logic based method.

Finally, a new extension to the classical Bayes algorithm is introduced. This method allows real-time adaptive classification behaviour.

3.1 Minimum Euclidean Distance Based Classification

Let us consider the two dimensional space shown in Figure 3-1. A number of vectors cluster around a vector $\underline{Z1}$. Another set of vectors are clustered around another vector, $\underline{Z2}$. A new vector, $\underline{x}$, needs to be classified. The question to be answered is: does $\underline{x}$ belong to class 1, represented by $\underline{Z1}$, or to class 2 represented by $\underline{Z2}$?

In the Failure Detection System context, class 1 could be the class of vectors representing a system that functions correctly, while class 2 could be the representation of a failed system. The two dimensions of the vectors in Figure 3-1, x1 and x2 could represent the difference between two temperatures measured at different points in the system and their values modeled by the modeling block in Figure 1-1. This vectorial difference represents a residual vector. Each vector of coordinates (x1,x2) represents an actual residual at one moment in time. Therefore, if the system functions properly, all measurements will cluster around vector $\underline{Z1}$ of coordinates (0,0). If the system fails, the new measurements will cluster around $\underline{Z2}$, the failure residual vector. The components of the vector x, are called the vector's features.

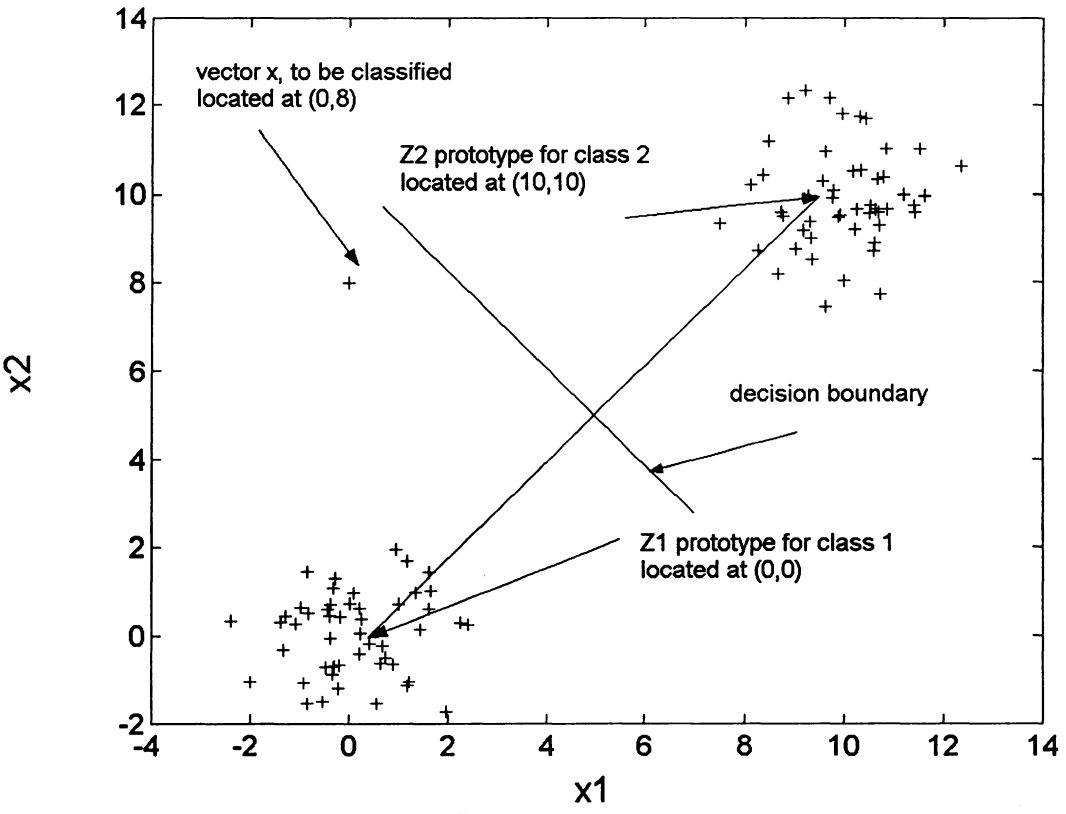


Figure 3-1 Two-class problem, new vector *X* to be classified.

If a new residual vector $\underline{x}$ is computed we wish to decide if $\underline{x}$ is indicative of a failed system. The Minimum Distance classifier groups the new vector with the class whose prototype is closest to $\underline{x}$. If $\underline{x}$ is closest to $\underline{Z1}$, then $\underline{x}$ belongs to class 1.

Let the Euclidean distance between $\underline{x}$ and $\underline{Z_k}$ be defined as:

$$d_E(\underline{x},\underline{Z_k}) = \left\{ \sum_{i=1}^{n} \left(x_i - x_{ki} \right)^2 \right\}^{\frac{1}{2}} \tag{3-1}$$

$$d_E(\underline{x},\underline{Z_k}) = \left\{ \left(\underline{x} - \underline{Z_k} \right)^T \left(\underline{x} - \underline{Z_k} \right) \right\}^{\frac{1}{2}} \tag{3-2}$$

The decision rule based on this metric decides that $\underline{x}$ belongs to class k if and only if:

$$d_E(\underline{x},\underline{Z_k}) \leq d_E(\underline{x},\underline{Z_l}) \qquad \text{for all } l \neq k \tag{3-3}$$

For a two class problem, the points whose distances are equal to both classes, form the decision boundary. For the Minimum Euclidean Distance (MED) classifier, the decision boundary is a hyperplane orthogonal to the line that connects the two class prototypes, and intersects the line half way between the two classes. Figure 3-1 shows the decision boundary. The advantage of the Minimum Euclidean Distance classifier is its simplicity. The only information one has to have in order to be able to use it, is the location of the class prototypes. Usually, these prototypes are computed as the statistical mean of the available samples known to belong to the class.

3.2 Minimum Intra-Class Distance Based Classification

A fundamental problem with the Euclidean distance method, is shown in Figure 3-2. According to the MED rule, the new vector $\underline{x}$ belongs to class 1. However, while it is not very close to the prototype of class 2, $\underline{Z2}$, it is close to a number of vectors known to

belong to class 2, one of which being located at (3.5, 7.2). There are two reasons for this phenomenon:

1. the variance of feature x1 is larger than the variance of feature x2. This is a common occurrence in practical applications caused by different units and gains attributed to the features presented to the classifier
2. the two features are correlated. Feature correlation will be a major topic in one of the case studies described in Chapter 4.

Intuitively, one feels that a different classification rule from the MED, should be used.

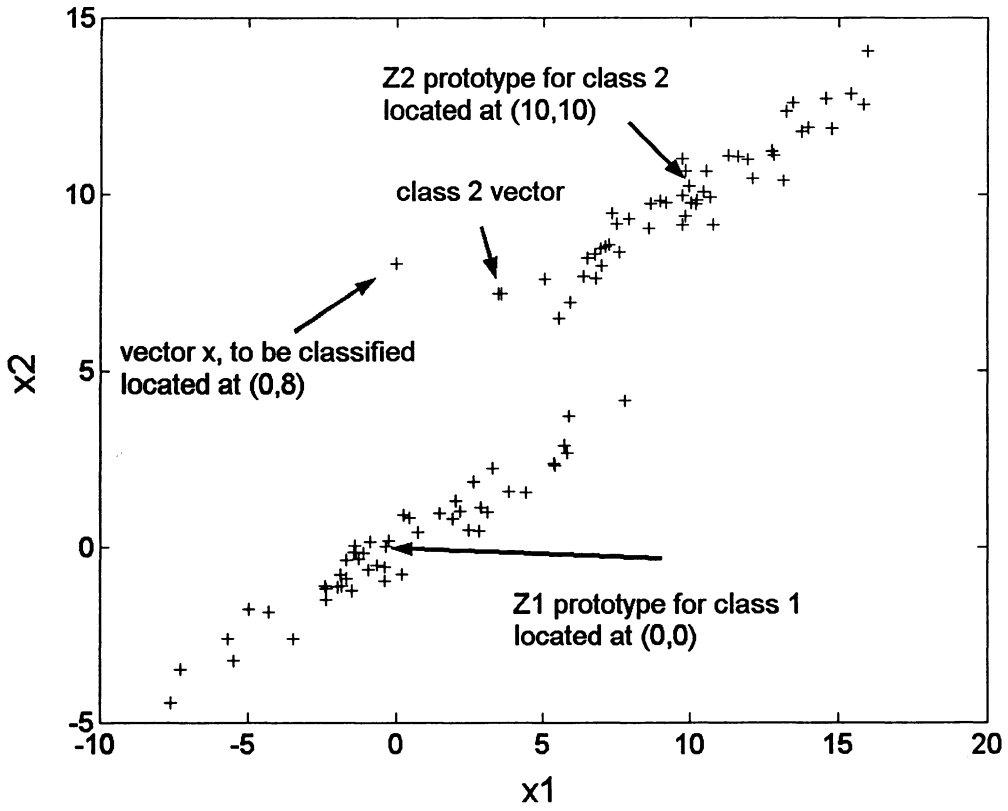


Figure 3-2 Two-class problem. Different feature variance and feature correlation.

An alternate approach to deriving a suitable metric for pattern classification is based on the premise that the metric should indicate that members of the same class are as similar

as possible. This can be achieved if the distance of the members within a class is minimized [62].

The new distance measure is derived as:

$$d^2\left(\underline{x},\{\underline{x}_k\}\right)=\left(\underline{x}-\overline{\underline{x}}\right)^T S^{-1}\left(\underline{x}-\overline{\underline{x}}\right)$$

where $\overline{\underline{x}}$ is the class mean, and S is the class covariance matrix (3-4)

$$\overline{\underline{x}}=\underline{m}=\frac{1}{N}\sum_{k=1}^{N}\underline{x}_k$$

This distance measure is called the Mahalanobis distance measure.

Therefore, given a two class classification problem, where the classes A and B have means $\underline{m}_A$ and $\underline{m}_B$, and covariance matrices S_A and S_B, a vector $\underline{x}$ belongs to class A if:

$$\left(\underline{x}-\underline{m}_A\right)^T S_A^{-1}\left(\underline{x}-\underline{m}_A\right)<\left(\underline{x}-\underline{m}_B\right)^T S_B^{-1}\left(\underline{x}-\underline{m}_B\right) \quad (3\text{-}5)$$

Note that unlike the Euclidean distance which was defined for any two vectors, the new distance is defined between a vector and a class. Therefore, for any given vector, each class has its own intra class distance metric determined by its own covariance matrix.

It is shown by Kittler [62] that the metric derived in equation (3-4) that minimizes the intra-class distance, also deals with the issues pointed out in Figure 3-2, namely the correlations between features and the different class variances. Therefore, if the class means and covariance matrices are known, or can be estimated from data, the Mahalanobis distance defined in equation (3-4) is a better choice compared to the simple Euclidian distance. For the case of uncorrelated features with the two classes having

different means and equal covariance matrices (diagonal), the Mahalanobis distance reduces to the Eucledian distance.

3.3 Maximum A Posteriori Probability Classifier

We have briefly introduced the Minimum Eucledian Distance (MED) and the Minimum Intra-Class Distance (MICD) classifiers. The MED classifier is appropriate for the simple cases having uncorrelated feature vectors, classes having different means and equal variances. If the above conditions are not met, the MICD is a better choice.

If more information about the probabilistic behaviour of the class is available (or can be estimated from data), an alternate approach to classification is possible. Assuming that the class conditional probability density functions are known, and that the probability of occurrence of the classes are also known or can be estimated, the Bayes classifier, also known as the Maximum A-posteriori Probability (MAP) classifier provides optimum classification [62], [63], [49], [50].

The following section will develop the Bayes decision surface, or the optimum threshold strategy. Such a strategy will either minimize the probability of making a mistake (false alarm or fail to alarm), or will minimize the cost of a mistake. The minimum requirements of a Bayes Classifier are the description of the probability density functions (pdf) of the residual values for the no-fault and faulty classes. In practical applications, deviation values computed from a no-fault system are readily available. There is relatively little difficulty in estimating the pdf for the no-fault case. Estimating the pdf of a faulted system is challenging. Normally, faulted system data are very sparse or not available. Moreover, the failure degree may vary resulting in a family of pdfs that need to be estimated.

Practical FDS avoid this issue by considering only the pdf of the no-fault system and setting the alarm threshold at a value large enough that would result in a low false alarm

probability. This approach does not consider the fail to alarm probability and is intrinsically sub-optimal. Once we have introduced the Bayes decision strategy, we will focus on several ways to estimate the required probability density functions. Two new approaches that use Fuzzy Logic controllers will be demonstrated.

3.3.1 Introduction to Bayes Classification

The Bayes classifier is based on the Bayes theorem published in 1763 [51]:

$$\text{Prob}(P_i|D)=\frac{\text{Prob}(D|P_i)*\text{Prob}(P_i)}{\text{Prob}(D|P_1)*\text{Prob}(P_1)+...+\text{Prob}(D|P_k)*\text{Prob}(P_k)} \tag{3-6}$$

where D, $P_{1, ...,} P_k$ are events, $P_{1, ...,} P_k$ are exclusive and exhaustive and Prob(P_i|D) is the conditional probability. In the special case of a two class problem where one class represents an OK system while the other one is a FAILED system and D represents an observed quantity (measured or computed), equation (3-6) becomes:

$$\text{Prob}(OK|D)=\frac{\text{Prob}(D|OK)*\text{Prob}(OK)}{\text{Prob}(D|OK)*\text{Prob}(OK)+\text{Prob}(D|FAILED)*\text{Prob}(FAILED)} \tag{3-7}$$

The Prob(OK|D) represents the probability that the system is OK given that we are observing the datum D. It is called the a posteriori probability. Prob(D|OK) represents the probability that an OK system generates datum D. It is called the likelihood probability. Prob(OK) is the probability that the system is OK before any datum D is observed. It is called the prior or a priori probability. Similarly:

$$\text{Prob}(FAILED|D)=\frac{\text{Prob}(D|FAILED)*\text{Prob}(FAILED)}{\text{Prob}(D|OK)*\text{Prob}(OK)+\text{Prob}(D|FAILED)*\text{Prob}(FAILED)} \tag{3-8}$$

A Failure Detection System needs to classify the state of a device into OK or FAILED given an observation D. If the FDS uses a Bayes classifier, it will assume that the device is OK if:

$$\text{Prob}(OK \mid D) \geq \text{Prob}(FAILED \mid D) \tag{3-9}$$

In other words, the classifier will select the most likely state given the observed data. In fact, for the classifier to make its decision and select the OK state, only the numerators of equations (3-8) and (3-9) need to be computed and compared:

$$\text{Prob}(D \mid OK) * \text{Prob}(OK) \geq \text{Prob}(D \mid FAILED) * \text{Prob}(FAILED) \tag{3-10}$$

This decision strategy has the advantage that it minimizes the probability of a mis-classification (a fail to alarm or a false alarm) [50]. This is why a Bayes classifier is optimum.

If rather than dealing with a discrete data event D, the observed datum is part of a continuum, some of the probabilities in equations (3-7) and (3-8) become conditional probability density functions:

$$\text{Prob}(OK \mid D) = \frac{pdf(D \mid OK) * \text{Prob}(OK)}{pdf(D \mid OK) * \text{Prob}(OK) + pdf(D \mid FAILED) * \text{Prob}(FAILED)} \tag{3-11}$$

$$\text{Prob}(FAILED \mid D) = \frac{pdf(D \mid FAILED) * \text{Prob}(FAILED)}{pdf(D \mid OK) * \text{Prob}(OK) + pdf(D \mid FAILED) * \text{Prob}(FAILED)} \tag{3-12}$$

and the decision to consider the device OK or FAILED is made if:

$$pdf(D \mid OK) * \text{Prob}(OK) \underset{FAILED}{\overset{OK}{\gtrless}} pdf(D \mid FAILED) * \text{Prob}(FAILED) \tag{3-13}$$

In order to evaluate inequality (3-13), the two density functions and two priors must be estimated. Taking into account the fact that the OK and FAILED system modes are exclusive and exhaustive, one can write:

$$\text{Prob}(FAILED) = 1 - \text{Prob}(OK) \tag{3-14}$$

Therefore one has to only know an estimate for Prob(OK) and the other prior is computed. The probability density functions of the two classes must be obtained from data obtained while observing the device. Most of the time, the device is in an OK state, therefore several estimation techniques can be used to estimate pdf(D|OK). Little or no data are available to describe the FAILED state, therefore the estimation of the pdf in this case is challenging. The following section will show how the two pdfs can be obtained. Field data introduced in Chapter 4.3 will be used. These field data refer to temperature measurements on the stator bars of an electric power generator. As it was discussed in Chapter 1, for each temperature measurement, a residual is computed by subtracting an estimate of the temperature at one particular point from its measured value. It is this residual that is being applied to the classifier, as an input.

3.3.2 Probability Density Estimation Using a Mixture of Gaussians

As we have discussed in the previous section, the estimation of the two class probability density functions is key to being able to implement the Bayes classifier. Since observed data are normally available for the OK state, we will start first with the estimation of this pdf.

There are two approaches to the estimation of such a function. If the pdf can be assumed to be of a certain type, it can be characterized by a set of parameters which can be estimated. This approach is called parametric estimation. If sufficient knowledge is available, it is advisable to try to fit a density function of prescribed form to the data [63].

This approach is used for a case study presented in Chapter 4.3, the electric generator overheating problem, and the pdf(D|OK) is estimated as a normal distribution. A Gaussian of zero mean and an undetermined standard deviation is fitted to the values of the disagreement for a particular temperature sensor. The resulting standard deviation estimate is caused by measurement noise, process modeling errors, and the normality assumption.

Figure 3-3 shows the resulting pdf(D|OK).

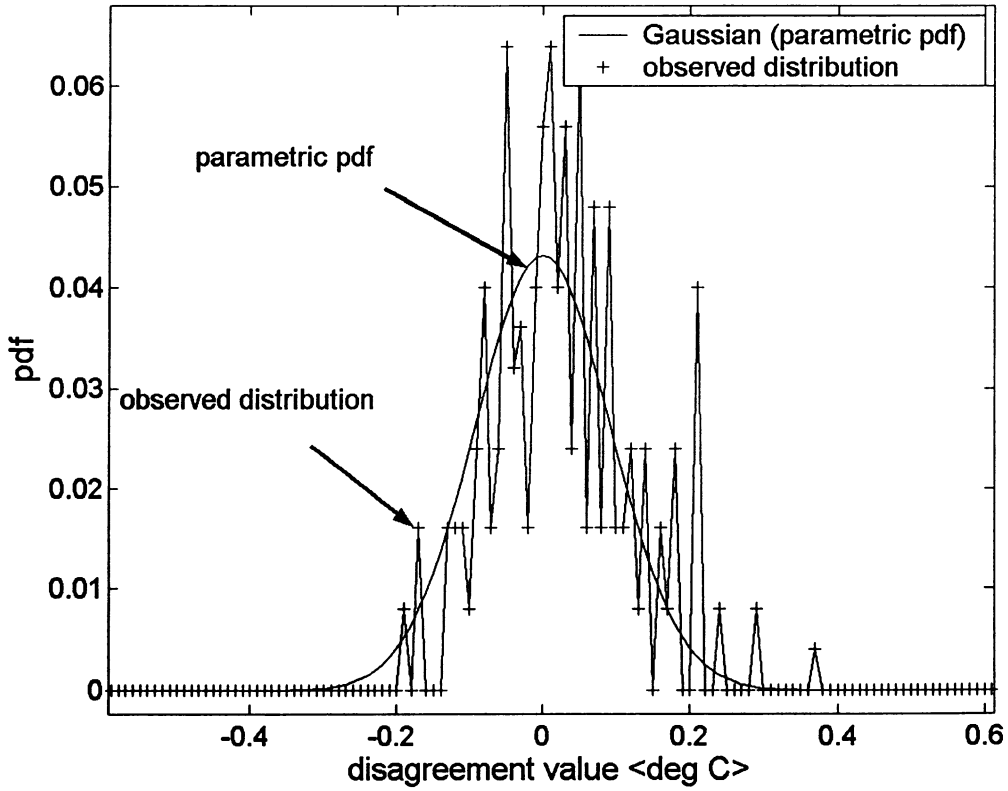


Figure 3-3 Parametric estimation of pdf. Measured histogram vs. Gaussian pdf.

While the parametric approaches yield impressive results, nonparametric techniques [50], [64] free of a priori assumptions (e.g. normal distribution assumed) are more general, therefore more adaptable. Histograms, splines, and mixture of densities are a few examples of nonparametric techniques [65]. Among the different nonparametric methods of pdf estimation, considerable interest has been shown in mixture density models [64], [66]. A mixture density is composed of a number of components, each being defined by its own set of parameters. The mixture model combines most of the flexibility of the

nonparametric models with the robustness of the parametric approach. Since in our generator stator overheating problem no FAILED class data is available, and since the generator can experience many degrees of overheating, the pdf for this class is constructed as a mixture density, the kernel used in the mixture being pdf(OK).

In the Overheating Detection System, we expect the pdf of a certain overheating value to be similar to the normal system pdf, but translated towards a positive disagreement mean value:

$$pdf(D \mid FAILED) = K * pdf(D - O_1 \mid OK) \tag{3-15}$$

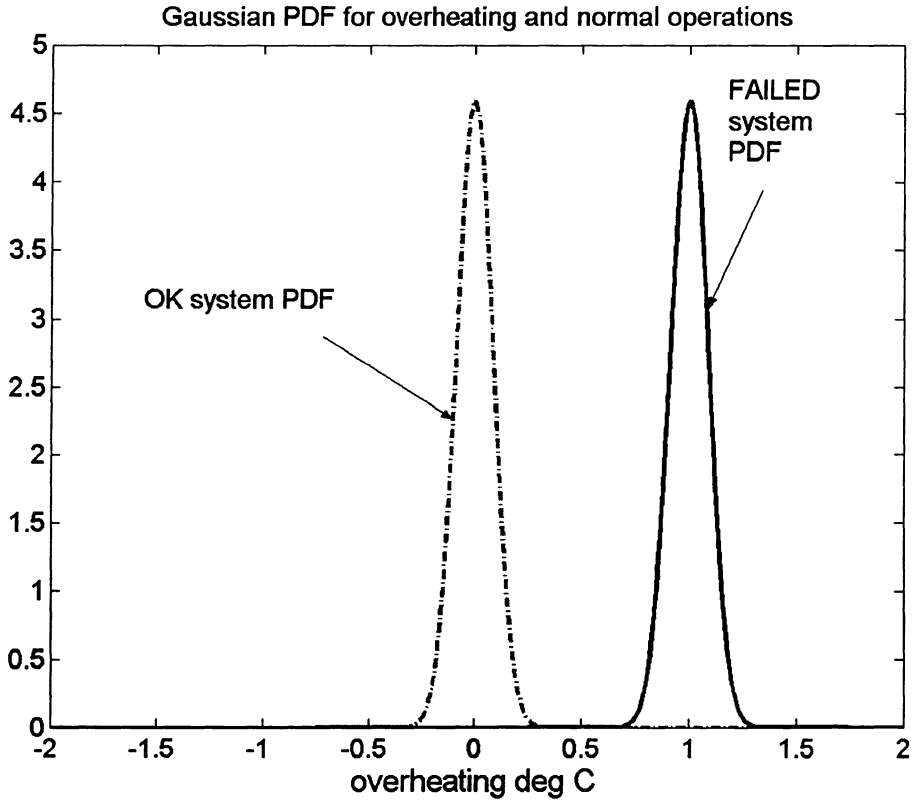


Figure 3-4 pdf estimate of FAILED system. One failure mode.

Figure 3-4 shows the pdf of the "OK" class as well as a "FAILED Overheating value O_1" class. In this case, the overheating value O_1 is 1°C.

If the system could fail in only one overheating value mode, the two Gaussians could be used by a Bayesian classifier to achieve optimum results. However, if two overheating values are predominant, and if the two failure modes are independent then a cumulative failure pdf can be obtained by summing the individual pdfs and applying an appropriate scaling:

$$pdf(D \mid FAILED) = W_1 * pdf(D - O_1 \mid OK) + W_2 * pdf(D - O_2 \mid OK) \qquad (3\text{-}16)$$

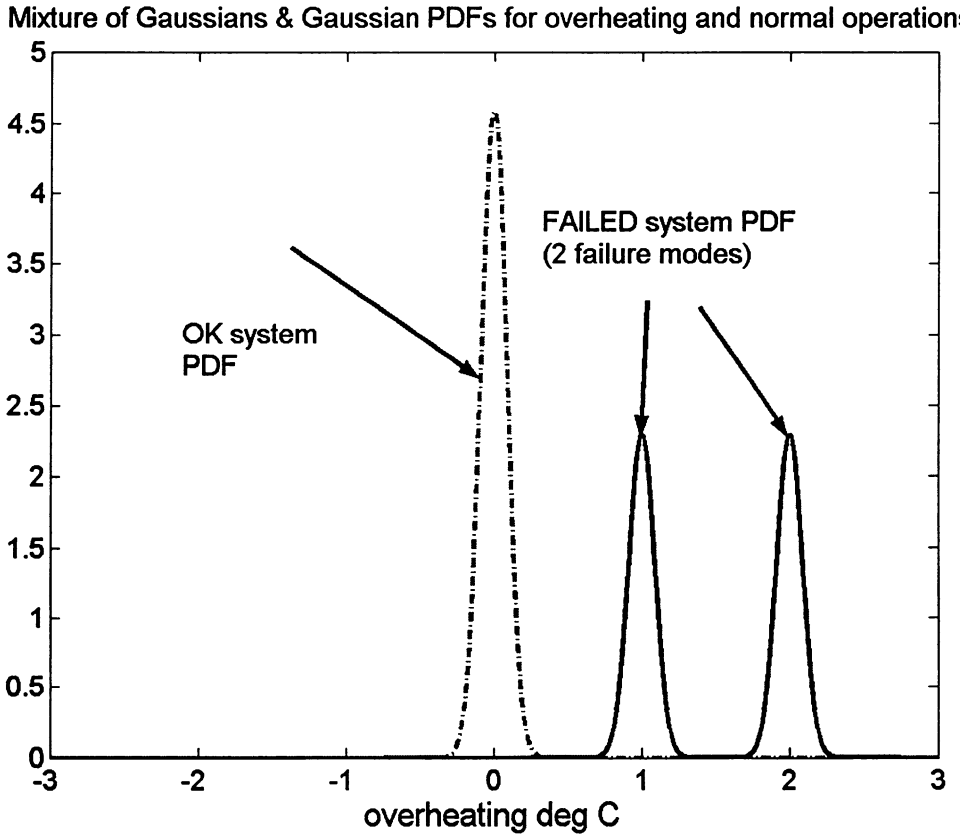


Figure 3-5 pdf estimate of FAILED system. Two failure modes.

Figure 3-5 shows, both the pdfs of the "System OK" class as well as a "System Failed Overheating value O_1 and O_2" class. In this case, the overheating values are 1°C for O_1 and 2°C for O_2.

If all overheating values from zero to infinity are possible, and if the probability density function of these modes is w(t), the previous sum of individual pdfs becomes a weighted sum of an infinite number of pdfs, which is expressed by a convolution integral:

$$pdf(D \mid FAILED) = K * \int_{-\infty}^{+\infty} w(x)\, pdf(D - x \mid OK)\, dx \tag{3-17}$$

K scales pdf(D|FAILED) in order to meet:

$$\int_{-\infty}^{+\infty} pdf(D \mid FAILED)\, dD = 1$$

If we assume that all failure modes are equally likely and that the pdf of failure outside the range [α, β] is zero, equation (3-17) becomes:

$$pdf(D \mid FAILED) = K * \int_{\alpha}^{\beta} pdf(D - x \mid OK)\, dx \tag{3-18}$$

If we further assume that pdf(D|OK) is a Gaussian, then the two pdfs become:

$$pdf(t \mid OK) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{t^2}{2\sigma}} \tag{3-19}$$

$$pdf(t \mid FAILED) = \frac{1}{2(\beta - \alpha)} \left[Erf\left(\frac{t-\alpha}{\sqrt{2\sigma}}\right) - Erf\left(\frac{t-\beta}{\sqrt{2\sigma}}\right) \right] \tag{3-20}$$

$$where: Erf(x) = \int_{0}^{x} e^{-t^2}\, dt \tag{3-21}$$

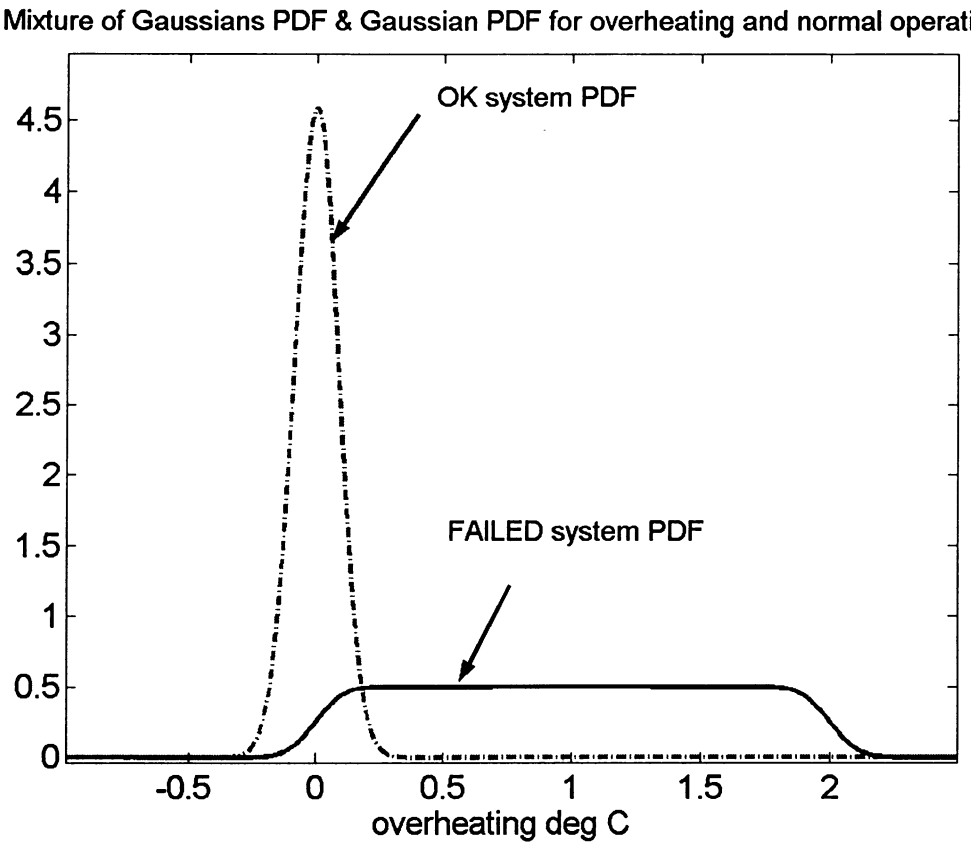


Figure 3-6 pdf(D|OK) and pdf(D|FAILED).

Figure 3-6 shows both pdf(D|OK) and pdf(D|FAILED). In equation (3-13), the Bayesian classifier scales the pdfs by their respective prior values and uses the intersection of the two resulting curves as a threshold value to be used for classification.

3.3.3 Probability Density Estimation Using a Fuzzy Logic Generator

This section will build on the fuzzy logic concepts introduced in Chapter 2.3 and demonstrate two Fuzzy Logic approaches that result in the estimation of a faulted system probability density function. The first implementation is based on a Mamdani inference, while the second one uses a Sugeno engine.

Chapter 3.3.2 has derived the probability density function of a failed system based on the pdf of the OK system and has shown in equation (3-17) that a convolution is performed between the pdf and a weighting function. Furthermore, the pdf of the failed system was computed for the special case when the pdf of the OK system is a Gaussian and the weighting function is a rectangular box.

$$pdf(D \mid FAILED) = K * \int_{-\infty}^{+\infty} w(x)\, pdf(D - x \mid OK)\, dx$$

$$w(x) = \begin{cases} 1 & \text{for } \delta_1 \le x \le \delta_2 \\ 0 & \text{otherwise} \end{cases}$$

Obviously, the above w(x) implies that all failure modes between δ_1 and δ_2 are equally likely and that there are no possible failure modes for residuals smaller than δ_1 or larger than δ_2. These assumptions, while resulting in useful results, may be too restrictive.

We are proposing now a Mamdani fuzzy logic engine that estimates the probability density function in a general case. The engine is a Single Input Single Output model. The input is the residual value, while the output is the pdf value. The membership functions shown in Figure 3-7 are Gaussians covering the range of temperature residuals of interest, in this case from 0°C to 8°C. The set rule is:

1. If the residual is Low then pdf is Fuzzy_0.0 with weight 1
2. If the residual is MediumLow then pdf is Fuzzy_0.1 with weight 1
3. If the residual is Medium then pdf is Fuzzy_0.25 with weight 1
4. If the residual is MediumHigh then pdf is Fuzzy_0.5 with weight 1
5. If the residual is High then pdf is Fuzzy_0.0 with weight 1

The weight for each rule is 1 implying that all rules are equally important in determining the value of the pdf.

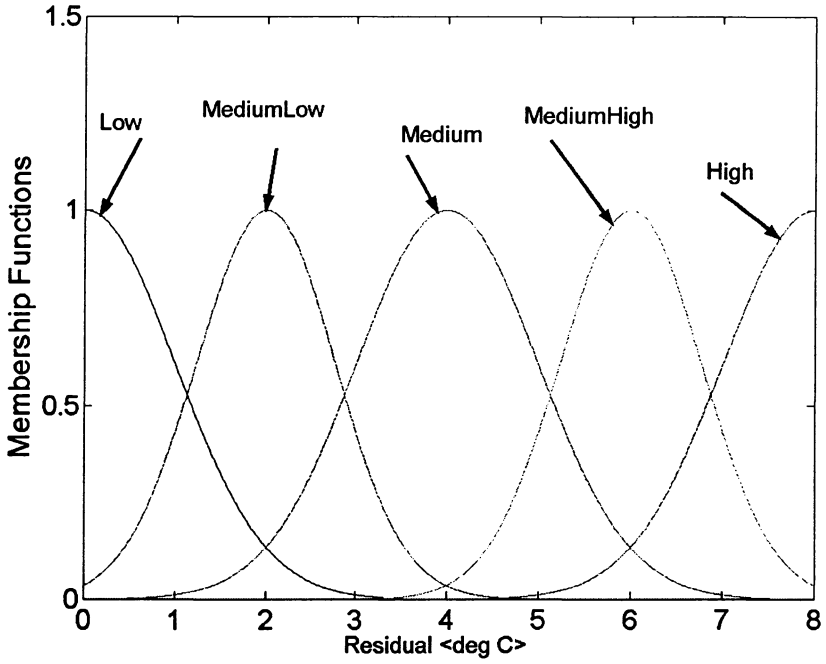


Figure 3-7 Input membership functions to be used with Mamdani engine.

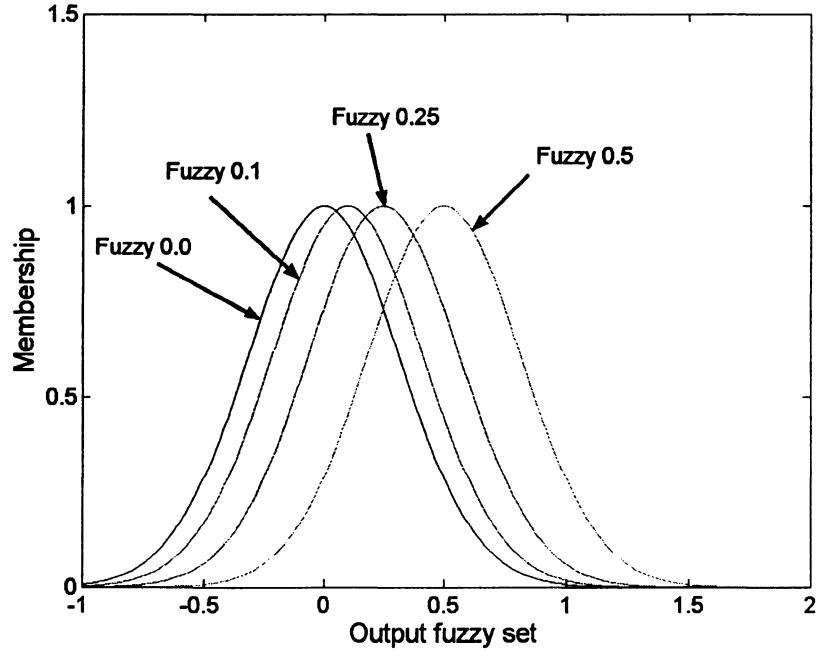


Figure 3-8 Output membership functions to be used with Mamdani engine.

The output fuzzy sets that were used are shown in Figure 3-8. Min-max inference was used.

A comment is now in order regarding the output membership functions shown in Figure 3-8. The range on which these functions are defined, or the universe of discourse, is [–1, 2]. The Min operator is applied to each function during rule inference, followed by the Max operator being applied during the rule aggregation. Finally, deffuzification is performed using the center of gravity method. Therefore, examining the universe of discourse, the maximum range of the crisp output that could result from this fuzzy engine is [-1, 2]. However, in our application, this output represents, a probability density function whose value must be greater than or equal to zero. The integral of the function over the whole interval must be equal to 1. The conclusion of this analysis is that the universe of discourse for the output membership functions may contain regions which do not have a physical meaning for the problem at hand. In our case, such an interval is [-1, 0)

Figure 3-9 shows the output produced by the Mamdani Fuzzy Logic inference. As was desired, the pdf increases gradually with the increase of the residual. This effect was achieved by careful positioning of the output fuzzy sets. The same effect could have been achieved by hand crafting the weighting function $w(x)$ in equation (3-17) in the previous section. The advantage of using the Fuzzy Logic approach lies in the fact that the output fuzzy set positioning may be based on problem domain expert knowledge. Based on this knowledge the Fuzzy Logic engine internally computes $w(x)$ and also performs the convolution, all in one step. This knowledge may be easier to acquire compared to knowledge regarding the particular shape for $w(x)$.

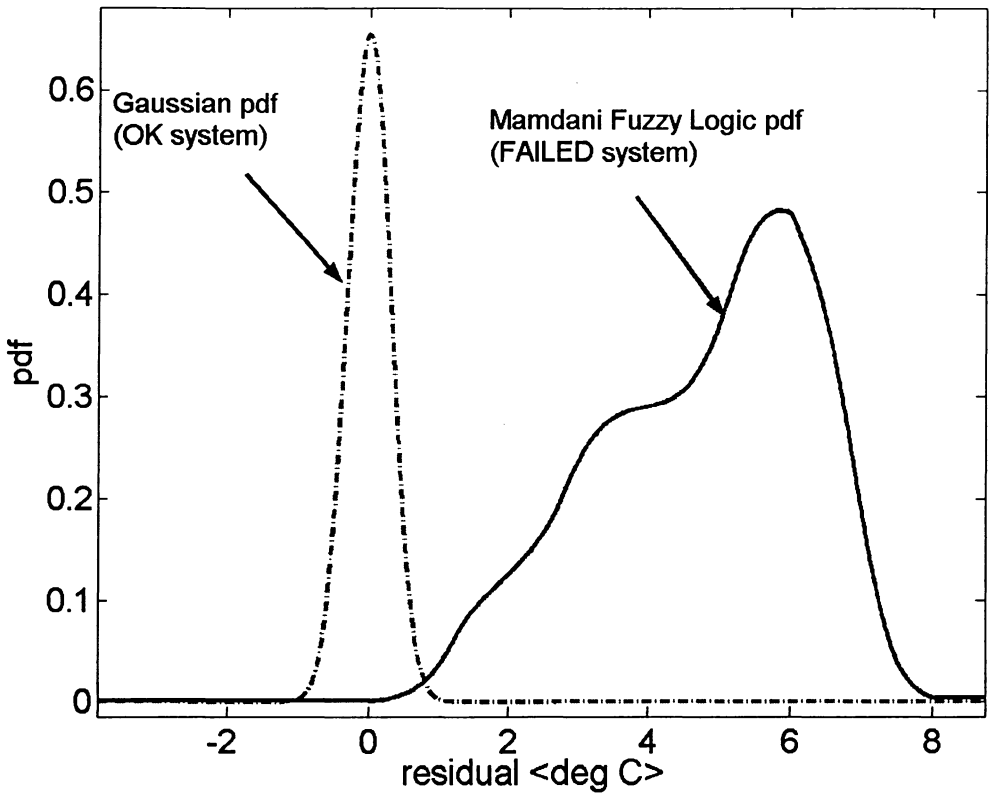


Figure 3-9 pdf of OK and FAILED system obtained with Mamdani inference.

An alternative approach of deriving the pdf_{failed} using the Mamdani inference, is to use a fuzzy logic engine employing the Takagi-Sugeno method of inference.

Just as in the previous case, we could have fed the engine with one input, the residual value. In order to provide the Fuzzy Logic design with invariance relative to input changes in scale and offset, we chose to compute five quantities which are the distance between the residual and five fixed values, and then to divide the differences by a fixed quantity. This allows us to design the Fuzzy Logic engine once, and rescale its behaviour by simply changing the position of the five fixed values and of the scaling factor.

In our example, the fixed values are: 1, 3, 4.5, 6, and 8. The goal is to design rules that produce a pdf value based on the fact that inputs may be close to the values: 1, 3, 5, 6, and 8.

The rules for the Sugeno engine are:

1. If input1 is Small then output is large with weight 0.3
2. If input2 is Small then output is large with weight 0.4
3. If input3 is Small then output is large with weight 0.5
4. If input4 is Small then output is large with weight 0.6
5. If input5 is Small then output is large with weight 1.0
6. If input1 is large AND input2 is large AND input3 is large AND input4 is large AND input5 is large then output is small with weight 1.0

For the output fuzzy sets, since an order 1 Sugeno inference is used, only one constant must be determined for each set (in equation (2-9), Chapter 2.3.1 all a_i are equal to zero, only b_i must be chosen). In our case, b_{large} is equal to 1, while b_{small} is equal to 0. If the residual value results in the input1 to be small, the output of the engine will be equal to 0.3. If input2 is small, then the output is 0.4, and so on. If the residual value results in all inputs to be large, then the output of the engine is 0. Unlike the Mamdani engine that modulates its output, based on the output fuzzy sets, our Sugeno engine performs the same function by using its rule weights as parameters.

Figure 3-10 shows the resulting probability density function. As expected, the pdf of the failed system is similar to the result obtained in Figure 3-9.

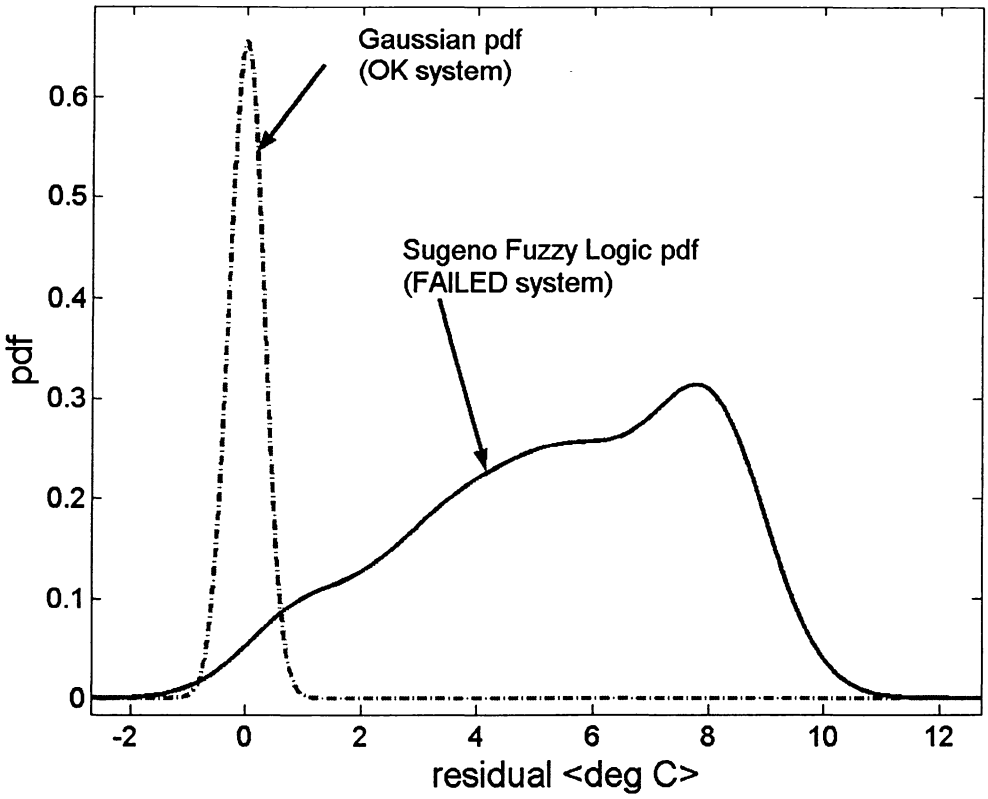


Figure 3-10 pdf of OK and FAILED system obtained using Sugeno inference.

3.3.4 On-Line Adaptive Bayes Classification

Chapter 3.3.1 showed how the Bayes classifier can make an optimum decision for a two class classification problem. In the case of classifiers used for Failure Detection Systems, the input in the classifier is a computed residual value and the two classes are the class of residuals that belongs to the system OK class, and the ones that belong to the system FAILED class. The chapter also showed how, by observing the values of the residuals during the normal operation of the system, one can estimate the probability density function pdf(D|OK) and the pdf(D|FAILED). By having an educated guess of the prior, P(OK), one can use equation (3-14) to compute P(FAIL). These four elements allow the application of the Bayes classifier every time a new disagreement value is

observed. The classifier can estimate the most likely system state, OK or FAILED. Every time a new value becomes available, the system evaluates inequality (3-13) and makes its decision. An equivalent technique to this evaluation is to compute once the intersection of the two scaled pdfs and use this intersection as a decision threshold for new disagreement values. From equation (3-13), setting the datum D to the value of the residual threshold, the inequality becomes an equality:

$$pdf(Th \mid OK) * \text{Prob}(OK) = pdf(Th \mid FAILED) * \text{Prob}(FAILED) \tag{3-22}$$

Equation (3-22) must be solved for the value of Th which is the decision threshold:

$$Th \begin{array}{c} OK \\ \geq \\ \leq \\ FAILED \end{array} D \tag{3-23}$$

If the value of D is larger than the threshold Th, the system has FAILED, otherwise the system is OK.

It is apparent that the threshold value is constant, being independent of new data. If before any data were available, the value of the prior P(OK) was assumed to be 0.5, is it reasonable to assume that after perhaps a large number of measurements that have resulted in small value residuals, the prior P(OK) should still be set at 0.5? Intuitively, one feels that the prior value should also change once data become available. If these data reinforce the belief that the system is OK, the prior value should increase, otherwise its value should be lowered. The result would be a system that compares the disagreement against an adaptive threshold.

Starting with equation (3-11), the Bayes theorem, and dividing both the numerator and the denumerator by the numerator value:

$$\text{Prob}(OK \mid D) = \frac{1}{1 + \frac{pdf(D \mid FAILED)}{pdf(D \mid OK)} * \frac{\text{Prob}(FAILED)}{\text{Prob}(OK)}} \tag{3-24}$$

which, when combined with equation (14) results in:

$$\begin{aligned} &\text{Prob}(OK_i \mid D_i) = \frac{1}{1 + x_i * a_i} \\ &where: a_i = \frac{1}{\text{Prob}(OK_i)} - 1 \\ &and: x_i = \frac{pdf(D_i \mid FAILED)}{pdf(D_i \mid OK)} \end{aligned} \tag{3-25}$$

The subscript "i" was used to indicate that the computation is done for a disagreement at step "i". The purpose of computation (3-25) is to estimate the probability that the system is OK. Once this is done, and before a new disturbance value at step "i+1" is used, the prior value is updated:

$$\text{Prob}(OK_{i+1}) = \text{Prob}(OK_i \mid D_i) \tag{3-26}$$

In other words, the a posteriori probability at step "i" becomes the prior at step "i+1".

In summary, the method we propose [67], for the adaptive Bayes classifier, is the following:

1. Estimate the conditional probabilities pdf(D|OK) and pdf(D|FAILED) using one of the methods outlined in the previous chapters (mixture of Gaussians, Fuzzy Logic, etc.)
2. Start with a "gut feel" value of the prior Prob(OK)
3. Use equation (25) to compute the a posteriori probability at step 1 (step "i"). We have performed now the classification at step "i"
4. Use equation (26) to propagate the a priori probability value.

5. Go to step 3

It will be shown in a case study in Chapter 4.2.6.1.5, that the re-estimation of the priors at every iteration results in the use of higher threshold values when small value residuals are computed, while large disagreement values result in small thresholds. In other words, if small disagreement values are seen by the system, the system has more confidence that no failure takes place and increases its threshold being more tolerant to spurious noise. If however, large disagreement values are experienced, the system lowers the threshold value making a classification to "FAILED" more likely. It will be seen that the proposed system is an improvement over a classical Bayesian implementation (fixed threshold) and a large improvement over a fixed, arbitrary value threshold classifier (which is the result of a false alarm minimization constraint with no regard for a fail to alarm consideration).

CHAPTER IV

4 Case Studies

This chapter will present three real-life problems and propose solutions within the Fault Detection System framework. These applications represent classes of FDS:

1. The Static Security Assessment in Electric Power System represents the class of FDS having an extremely limited data set available for learning. The internal modeling blocks must perform a non-linear mapping. Some of the inputs are highly correlated, resulting in a potential near singularity, unless de-correlation methods are used.

2. The Oil Leak Detection in Underground Power Cables represents the class of FDS having an abundant amount of learning data acquired while the system is functioning correctly (no oil leaks take place). However, no data from a failed system are available (no leak data). This situation is typical of the majority of FDS applications, where one does not have the luxury of witnessing a failed system in operation. Another complication that must be addressed involves the strongly non-linear phenomena, which overlap the process the FDS is trying to model (oil pressure is changed by valve and pump operations, distorting the oil temperature/pressure relationship to be modeled). Finally, the issue of data corruption and data loss will be addressed. A relatively small amount of data loss present in the data set might render the whole set unusable. We will present an approach that was developed to solve this general problem.

3. The Detection of Flow Restrictions in Water-Cooled Generator Windings problem represents the class of FDS applications which have to deal with a large number of highly correlated input quantities that also have a rather small

variability (the electric power generator works very closely to its set point, most of the time). These facts can result in a poorly numerically conditioned problem, unless data pre-conditioning is performed. Solutions to this application have been published and make use of non-linear modeling blocks. We will present an approach which needs only linear models (non-linear modeling ability does not add significant accuracy), and does not have to linearise the non-linear physical model around an operating point (as it is usually done, reducing the problem to a small signal application). Similar to the Oil Leak Detector, no fault data are available for learning.

4.1 The Static Security Assessment in Electric Power Systems

Voltage stability is a problem in power systems which are heavily loaded, faulted, or have a shortage of reactive power [18]. The problem of voltage stability concerns the whole power system, although it usually has a large involvement in one critical area.

This chapter first introduces the notion of maximum power transfer. It then shows two cases of voltage collapse experienced by a loaded system. A possible solution that reverses the voltage collapse trend is shown. The power flow equations of the power system are then introduced. These equations allow the calculation of the steady state bus voltages. Because the equations are non-linear, several iterative methods used to compute the solutions are mentioned. While these approaches are adequate for off-line studies, their computational load prevents them from being applied in an on-line setting. In order to achieve a high computational speed, we propose a black-box approach that utilizes a number of Neural Networks as models. We show that a very limited learning data set results in a poor recall performance of the Neural Networks. We introduce a new method that conditions the data set, thus solving the sparse data set issue. This method is first applied to a very small modeling example, followed by its application to the New England IEEE 39 bus system. Finally, we show the performance of the modeling approach when various parameters of the Neural Networks are varied.

4.1.1 State of the Art

The Power-Voltage Curves

The concept of voltage stability can be exemplified by the analysis of the circuit in Figure 4-1. The generator *G* produces active and reactive power transferred via a

transmission line of reactance X to the load having the series resistance R_L and reactance X_L. The load uses the active power P and reactive power Q.

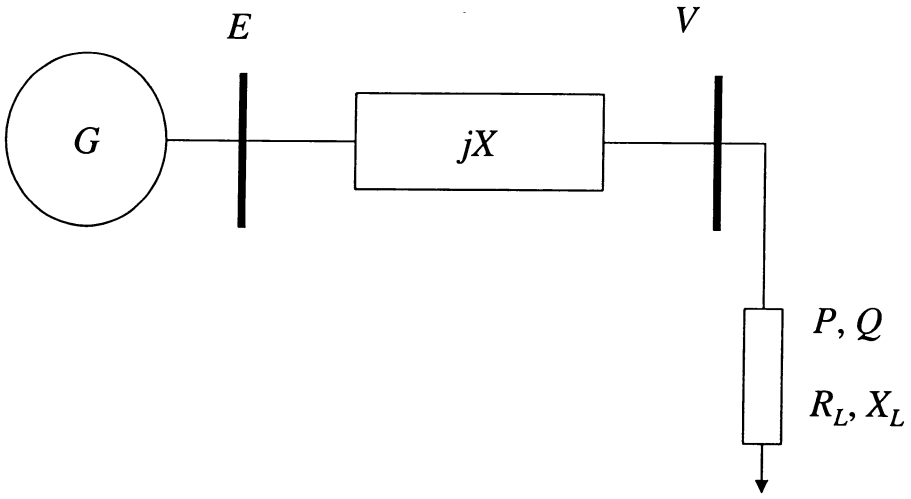


Figure 4-1 Two bus system.

We would like to calculate the value of the load voltage V, as a function of E, X, P, and Q.

$$\vec{V} = E\frac{R_L + jX_L}{R_L + j(X + X_L)} \tag{4-1}$$

$$\vec{I} = \frac{E}{R_L + j(X + X_L)} \tag{4-2}$$

$$\vec{S} = \vec{V} \cdot \vec{I}^{*} \tag{4-3}$$

Voltage E is used as a reference. The load voltage V and current I are used as phasors, to account for the shift in phase between them and E.

Substituting (4-2) into (4-3) and applying the complex conjugate operator:

$$\vec{S} = E^2\frac{R_L + jX_L}{R_L^2 + (X + X_L)^2} \tag{4-4}$$

By definition, the load active power is the real part of the total power S, while the reactive power is the imaginary part of S:

$$P = E^2 \frac{R_L}{R_L^2 + (X + X_L)^2} \tag{4-5}$$

$$Q = E^2 \frac{X_L}{R_L^2 + (X + X_L)^2} \tag{4-6}$$

Using (4-1) to compute the magnitude of V:

$$V^2 = E^2 \frac{R_L^2 + X_L^2}{R_L^2 + (X + X_L)^2} \tag{4-7}$$

Our task is to eliminate R_L and X_L from (4-5), (4-6), and (4-7). Dividing equation (4-5) by (4-6):

$$\frac{P}{Q} = \frac{R_L}{X_L} \tag{4-8}$$

and R_L is:

$$R_L = \frac{P}{Q} X_L \tag{4-9}$$

Equations (4-6) and (4-7) become:

$$Q = E^2 \frac{X_L}{\left(\frac{P}{Q}\right)^2 X_L^2 + (X + X_L)^2} \tag{4-10}$$

$$V^2 = \frac{P^2}{Q} X_L + Q X_L \tag{4-11}$$

Combining (4-10) and (4-11) by eliminating X_L results in:

$$Q = E^2 \frac{\frac{V^2 Q}{P^2+Q^2}}{\left(\frac{P}{Q}\right)^2 \frac{V^4 Q^2}{(P^2+Q^2)^2} + (X + \frac{V^2 Q}{P^2+Q^2})^2} \tag{4-12}$$

Equation (4-12) is a quadratic in V^2 and the solutions are:

$$V = \sqrt{\frac{(E^2 - 2QX) \pm \sqrt{E^4 - 2QXE^2 - 2P^2X^2}}{2}} \tag{4-13}$$

If we consider the load power factor:

$$\tan(\theta) = \frac{Q}{P} \tag{4-14}$$

and we vary P for a fixed value of the power factor, we obtain the family of curves shown in Figure 4-2.

To compute these curves, the line impedance, X, was set to 100Ω. Both V and P were normalized. *Pmax* is the maximum active power that can be transferred to a unity power factor load. The nose point on each curve indicates the maximum active power that can be transferred to the load through the transmission line. For each curve, the voltage corresponding to the maximum power transfer is called the critical voltage. The reason for this name will become apparent shortly. As can be seen, for a desired active power transfer, the operating point is closer to the maximum power transfer, if the power factor is larger, compared to a smaller power factor. A negative power factor can result in an actual increase of the load voltage with an increase of the load.

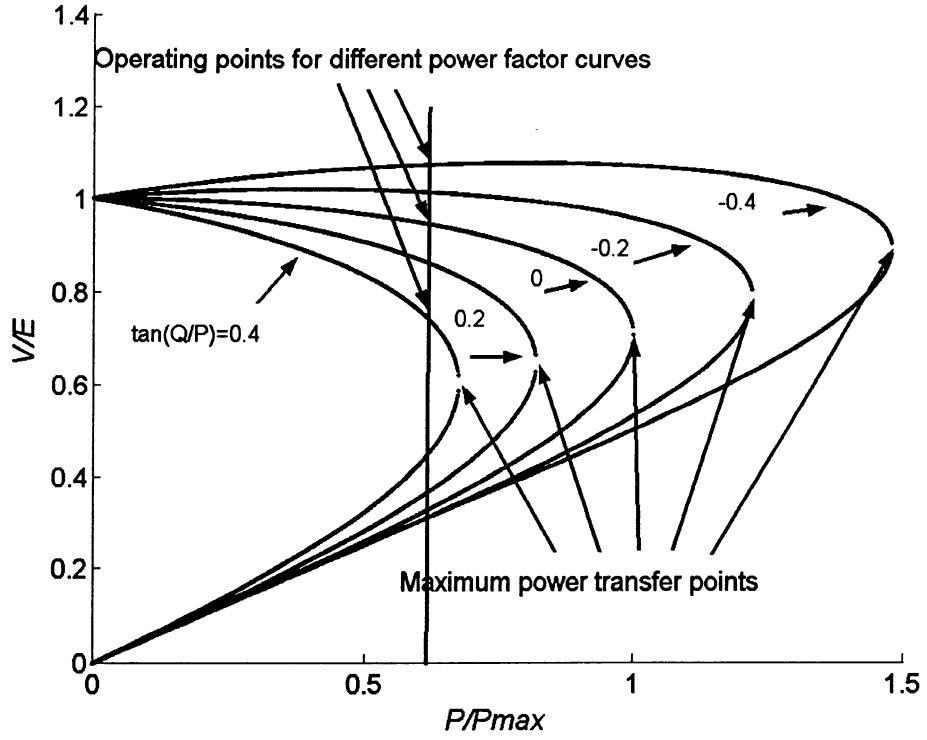


Figure 4-2 PV characteristics for different power factors.

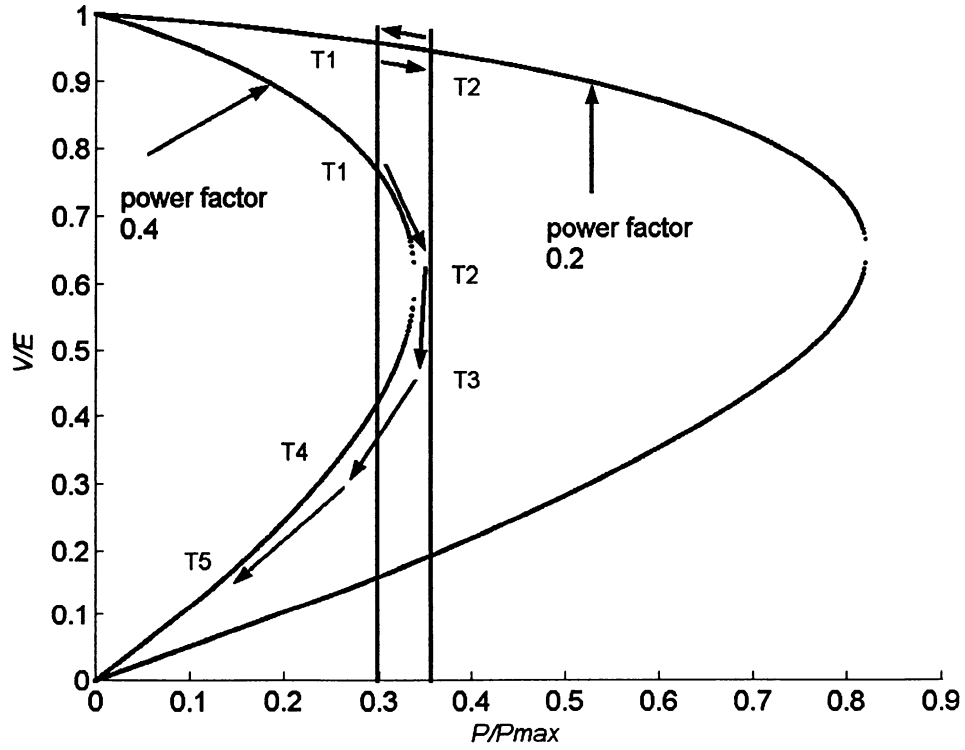


Figure 4-3 Temporary motor power demand increase may result in voltage collapse.

Let us consider a constant power load. An example of such a load is an induction motor. Figure 4-3 shows two examples: a motor having a power factor of 0.2 and one with a power factor of 0.4. We assume that initially, both motors match a mechanical power demand *P/Pmax* of 0.3. If the mechanical power demand is increased to 0.35, the motor having power factor 0.2 finds a new position of equilibrium. Once the demand becomes 0.3, the operating point returns to the same position as before. However, the motor having power factor 0.4 behaves differently. A power demand of 0.35 is beyond the maximum power transfer of that curve. Therefore, the power transferred does not match the power demand, and the motor starts stalling. This reaction, in turn, results in a decrease of the motor's impedance, which results in a lowering of load voltage, which, in turn, further lowers the transferred active power. As it can be seen in Figure 4-3, even if the power demand is restored to 0.3, the voltage collapses and the motor stalls. For this reason, the voltage corresponding to the maximum power transfer of a curve, is called the critical voltage.

The example above showed that a temporary increase in the power demand of a constant power load can result in a voltage collapse. The following example will show that a contingency affecting the power delivery capability of the system can also result in a voltage collapse.

Let us consider an aggregated constant power load (large industrial motors) having a power factor of 0.2 supplied by a line having an impedance of 100Ω. It can be seen that the operating point is well above the critical voltage point and the system is stable (Figure 4-4). Let us assume that the line trips and that power is delivered via an alternate path having a larger impedance of 200Ω. The new PV curve shifts, having this time the maximum power transfer capability less than the power demand. Just like in the previous example, the motors start stalling, thus decreasing their impedance, resulting in a voltage collapse. This situation can be avoided, if the power factor is decreased before a complete voltage collapse. A capacitor can be used as illustrated in Figure 4-5.

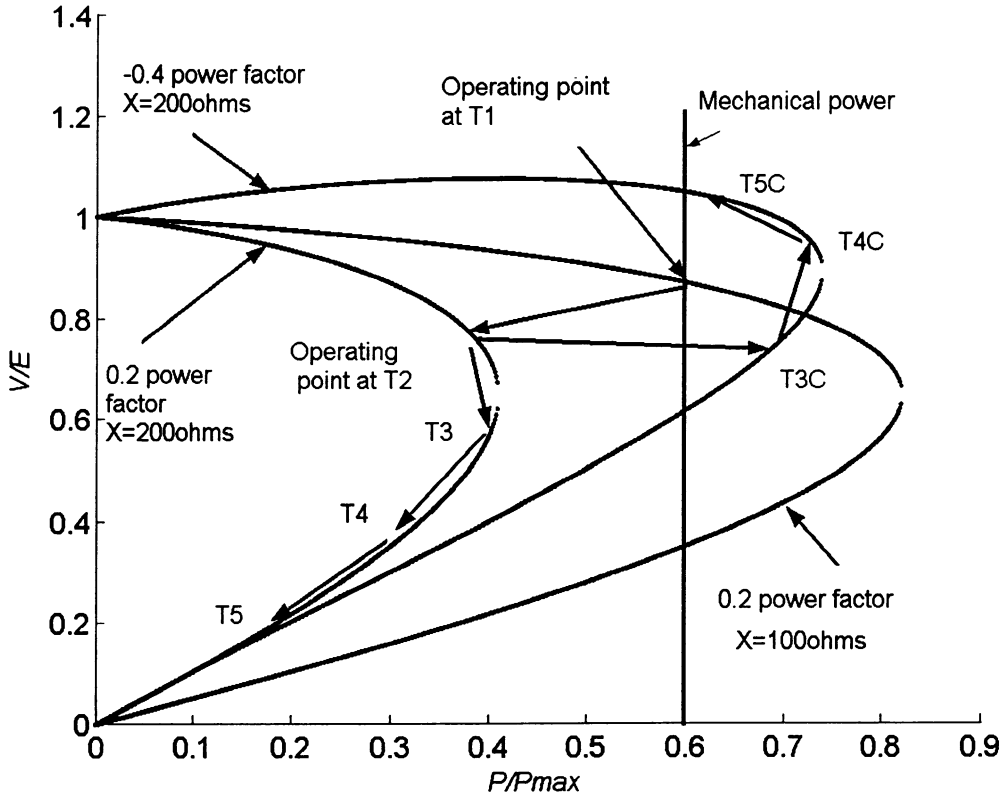


Figure 4-4 Contingency resulting in voltage collapse.

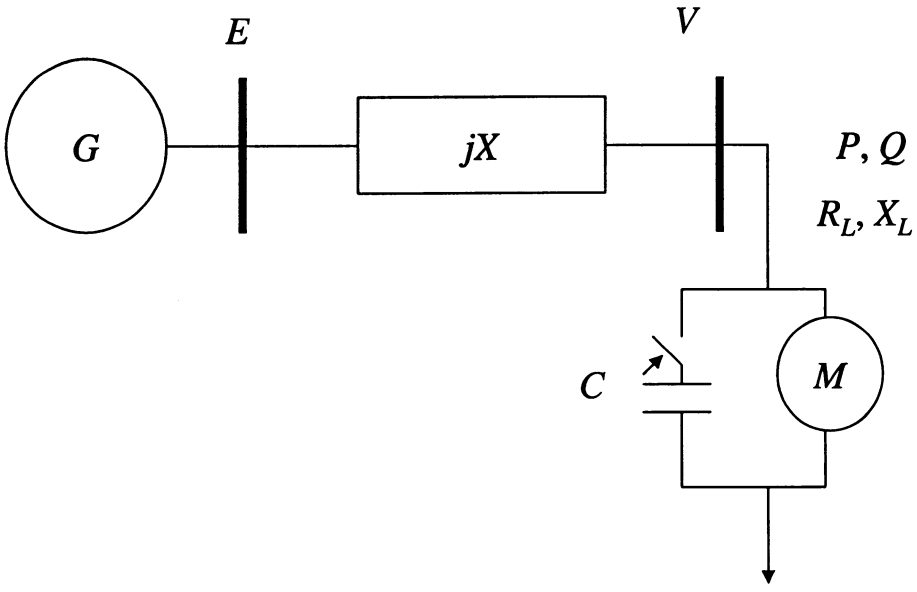


Figure 4-5 Capacitor compensator.

It can be noted that the system is stable once the capacitor is switched in, in spite of the fact that the voltage dipped below the minimum critical voltage at time T3C. The delivered power is larger than the power demand, allowing the motors to accelerate. This increase in rotational speed results in an impedance increase allowing the operating point to move to the power demand value.

Transmission Power Flow Analysis

Now that we have exemplified the effects of an unsuitable power transfer capability / load power factor pair, we shall examine ways of determining whether or not a system is close, at any bus, to the critical voltage point. The load-flow, or power-flow, analysis involves the calculation of power flows and voltages for a specific system configuration, certain loads and generation dispatch. The network equations can be written in terms of the node admittance matrix [18]:

$$\begin{bmatrix} \bar{I}_1 \\ \bar{I}_2 \\ \vdots \\ \bar{I}_n \end{bmatrix} = \begin{pmatrix} Y_{11} & \cdots & Y_{1n} \\ \vdots & \ddots & \vdots \\ Y_{n1} & \cdots & Y_{nn} \end{pmatrix} \begin{bmatrix} \bar{V}_1 \\ \bar{V}_2 \\ \vdots \\ \bar{V}_n \end{bmatrix} \tag{4-15}$$

where:

n is the total number of nodes

Y_{ii} is the self admittance of node i (sum of all admittances terminating at node i)

Y_{ij} is the mutual admittance between nodes i and j (negative of the sum of all admittances between nodes i and j)

$\bar{V}_i$ is the phasor voltage to ground at node i

$\bar{I}_i$ is the phasor current flowing into the network at node i

The system of equations (4-15) would be linear if the currents were known. In practice, the currents injected at each node are not known. Their values can be specified in terms of active, reactive, powers injected at nodes, as well as the node voltage:

$$\bar{I}_k = \frac{P_k - jQ_k}{\bar{V}^*_k} \tag{4-16}$$

Substituting (4-16) into (4-15) and writing the equation at bus k:

$$\frac{P_k - jQ_k}{\bar{V}^*_k} = Y_{kk}\bar{V}_k + \sum_{\substack{i=1 \\ i \neq k}}^{n} Y_{kk}\bar{V}_k \tag{4-17}$$

and the voltage V_k is computed as:

$$\bar{V}_k = \frac{P_k - jQ_k}{Y_{kk}\bar{V}^*_k} - \frac{1}{Y_{kk}} \sum_{\substack{i=1 \\ i \neq k}}^{n} Y_{kk}\bar{V}_k \tag{4-18}$$

Equation (4-18) is at the basis of an iterative algorithm called the Gauss-Seidel method. Each bus voltage is computed based on an informed guess of the other voltages in the system. In general, this iterative approach has slow convergence [18] due to the weak diagonal dominance of the node admittance matrix. One factor used to accelerate the convergence rate is an overly corrected solution based on:

$$\bar{V}^{new}{}_{k\_accelerated} = \bar{V}^{old}{}_k + c(\bar{V}^{new}{}_k - \bar{V}^{old}{}_k) \tag{4-19}$$

where c is the acceleration factor and has a value greater than unity.

Another approach used to solve the non-linear system (4-18), is the Newton-Raphson (N-R) method.

Let us consider a system of non-linear equations:

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n) = b_1 \\ &f_2(x_1, x_2, \ldots, x_n) = b_2 \\ &\vdots \\ &f_n(x_1, x_2, \ldots, x_n) = b_n \end{aligned} \tag{4-20}$$

We can replace the unknowns x_i with a set of initial estimates combined with a set of corrections to these estimates, in order to satisfy (20):

$$\begin{aligned} &f_1(x^0_1 + \Delta x_1, x^0_2 + \Delta x_2, \ldots, x^0_n + \Delta x_n) = b_1 \\ &f_2(x^0_1 + \Delta x_1, x^0_2 + \Delta x_2, \ldots, x^0_n + \Delta x_n) = b_2 \\ &\vdots \\ &f_n(x^0_1 + \Delta x_1, x^0_2 + \Delta x_2, \ldots, x^0_n + \Delta x_n) = b_n \end{aligned} \tag{4-21}$$

Expanding each function f_i in a Taylor expression around x^0_i , keeping only the linear terms in Δx_i , and solving for Δx_i results in:

$$\begin{pmatrix} b_1 - f_1(x^0_1, x^0_2, \ldots, x^0_n) \\ b_2 - f_2(x^0_1, x^0_2, \ldots, x^0_n) \\ \vdots \\ b_n - f_n(x^0_1, x^0_2, \ldots, x^0_n) \end{pmatrix} = \begin{pmatrix} \left(\frac{\partial f_1}{\partial x_1}\right)_0 \left(\frac{\partial f_1}{\partial x_2}\right)_0 & \cdots & \left(\frac{\partial f_1}{\partial x_n}\right)_0 \\ \vdots & \ddots & \vdots \\ \left(\frac{\partial f_n}{\partial x_1}\right)_0 \left(\frac{\partial f_n}{\partial x_2}\right)_0 & \cdots & \left(\frac{\partial f_n}{\partial x_n}\right)_0 \end{pmatrix} \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{pmatrix} \tag{4-22}$$

or:

$$\Delta f = J * \Delta x \tag{4-23}$$

Matrix J is called the Jacobian matrix. If the initial estimates of x_i were exact, then Δf would be zero resulting in vector Δx being also zero. The Taylor expansion has resulted in the linear set equation (4-23).

In order to apply the N-R method to the power-flow problem, we write the total power injected at a bus as a function of the real and reactive powers:

$$\overline{S}_k = P_k + jQ_k = \overline{V}_k \overline{I}_k^* \tag{4-24}$$

Using (4-15) to eliminate the currents I_k from (4-24) results in:

$$P_k + jQ_k = \overline{V}_k \sum_{m=1}^{n} (G_{km} - jB_{km}) \overline{V}_m^* \tag{4-25}$$

Isolating the real and imaginary parts:

$$P_k = \overline{V}_k \sum_{m=1}^{n} (G_{km} V_m \cos\theta_{km} + Bk_m V_m \sin\theta_{km})$$

$$Q_k = \overline{V}_k \sum_{m=1}^{n} (G_{km} V_m \sin\theta_{km} - Bk_m V_m \cos\theta_{km})$$

(4-26)

where θ_{km} is the angle difference between the voltages at buses k and m. Using (4-22) and substituting for b_i the specified active and reactive powers, P_i^{sp} and Q_i^{sp} , at all buses, results in:

$$\begin{pmatrix} P^{sp}{}_1 - P_1(\theta^0{}_1, \theta^0{}_2, \ldots, \theta^0{}_n, V^0{}_1, V^0{}_2, \ldots, V^0{}_n) \\ \vdots \\ P^{sp}{}_n - P_n(\theta^0{}_1, \theta^0{}_2, \ldots, \theta^0{}_n, V^0{}_1, V^0{}_2, \ldots, V^0{}_n) \\ Q^{sp}{}_1 - Q_1(\theta^0{}_1, \theta^0{}_2, \ldots, \theta^0{}_n, V^0{}_1, V^0{}_2, \ldots, V^0{}_n) \\ \vdots \\ Q^{sp}{}_n - Q_n(\theta^0{}_1, \theta^0{}_2, \ldots, \theta^0{}_n, V^0{}_1, V^0{}_2, \ldots, V^0{}_n) \end{pmatrix} = \begin{pmatrix} \frac{\partial P_1}{\partial \theta_1} \cdots \frac{\partial P_1}{\partial \theta_n} & \frac{\partial P_1}{\partial V_1} \cdots \frac{\partial P_1}{\partial V_n} \\ \vdots & \vdots \\ \frac{\partial P_n}{\partial \theta_1} \cdots \frac{\partial P_n}{\partial \theta_n} & \frac{\partial P_n}{\partial V_1} \cdots \frac{\partial P_n}{\partial V_n} \\ \frac{\partial Q_1}{\partial \theta_1} \cdots \frac{\partial Q_1}{\partial \theta_n} & \frac{\partial Q_1}{\partial V_1} \cdots \frac{\partial Q_1}{\partial V_n} \\ \vdots & \vdots \\ \frac{\partial Q_n}{\partial \theta_1} \cdots \frac{\partial Q_n}{\partial \theta_n} & \frac{\partial Q_n}{\partial V_1} \cdots \frac{\partial Q_n}{\partial V_n} \end{pmatrix} \begin{pmatrix} \Delta\theta_1 \\ \\ \Delta\theta_n \\ \Delta V_1 \\ \vdots \\ \Delta V_n \end{pmatrix} \tag{4-27}$$

or:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} \frac{\partial P}{\partial \theta} & \frac{\partial P}{\partial V} \\ \frac{\partial Q}{\partial \theta} & \frac{\partial Q}{\partial V} \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix} \tag{4-26}$$

where the four elements of the Jacobian are submatrices.

As has been previously mentioned, the Gauss-Sidel method is reliable, however it is relatively slow for larger systems. The Newton-Raphson method has a reasonable convergence rate, being well suited to large systems, however it relies on an appropriate starting solution [18]. A good combination, is to start the computation using the Gauss-Sidel method and then to switch to N-R when one is close to the solution.

To accelerate the N-R method, Fast Decoupled Load-Flow (FDLF) methods are used. These techniques exploit the weak physical coupling between active powers P and voltages V, and also, between the reactive powers Q and voltage angles θ.
Equation (4-26) is reduced to:

$$\Delta P = \frac{\partial P}{\partial \theta} \Delta\theta$$
$$\Delta Q = \frac{\partial Q}{\partial V} \Delta V \tag{4-27}$$

or:

$$\Delta P = H\Delta\theta$$
$$\Delta Q = L\Delta V \tag{4-28}$$

where:

$$H_{km} = \frac{\partial P_k}{\partial \theta_m} = V_k V_m (G_{km}\sin\theta_{km} - B_{km}\cos\theta_{km}) \qquad \text{for m} \neq \text{k}$$
$$H_{kk} = \frac{\partial P_k}{\partial \theta_k} = -B_{kk}V_k^{\,2} - Q_k$$
$$L_{km} = \frac{\partial Q_k}{\partial V_m} = V_k (G_{km}\sin\theta_{km} - B_{km}\cos\theta_{km}) = H_{km}/V_m \qquad \text{for m} \neq \text{k} \tag{4-29}$$
$$L_{kk} = \frac{\partial Q_k}{\partial V_k} = -B_{kk}V_k - Q_k/V_k$$

The N-R method requires the Jacobian matrix to be re-evaluated after each iteration. Similarly, matrices H and L have to be re-evalutated and re-triangularized at each iteration.

In order to further speed up the load flow algorithm, simplifications can be made [18]:

$$\cos\theta_{km} \approx 1$$
$$G_{km}\sin\theta_{km} \ll B_{km} \tag{4-30}$$
$$Q_k \ll B_{km}V^2{}_k$$

With these simplifications, system (28) becomes [18]:

$$\Delta P/V = B'\Delta\theta$$
$$\Delta Q/V = B''\Delta V \tag{4-31}$$

The advantage of this formulation is in the fact that matrices B' and B'' are real and sparse, and that they contain only network admittances that are constant; they have to be triangularized only once at the beginning of the load flow computation. Being an approximation of the N-R method, the FDLF requires a larger number of iterations to converge to a solution. However due to its lower computational requirements (no need

for Jacobian re-evaluation), this method is faster. It should be kept in mind that one approximation made in the FDLF is that of small angles across lines (first equation in (4-30)). Therefore, for system conditions with very large angles, the full N-R algorithm may be needed.

As was previously discussed, being able to do an On-Line Static Security Assessment is very useful. In order to achieve this goal, a number of developments have been published in the literature. The limitation imposed by the finite number of off-line load flow studies, are mitigated to a certain extent in [68]. A feedforward NN is trained with the backpropagation algorithm in order to perform an interpolation thus approximating unstudied cases. A Neural Network is trained in [69] to compute a voltage security index, based on off-line studies. [70] computes a severity index for contingency screening using a NN also trained on off-line studies. A contingency screening and ranking was performed in [71], [72] using a feedforward NN trained on off-line studies.

A number of papers [73], [74], [75], [76], [77] have used the Kohonen neural network to identify similarities of system states in order to assist in contingency analysis. Unlike the feedforward NN trained with backpropagation, the Kohonen network uses an unsupervised learning scheme; no teacher is required. The Kohonen network has the property of automatically clustering the input data vectors it receives during the learning process. Naturally, the result of its clustering process is fully dependent on the specifics of the input vectors the networks does its learning on. It is the responsibility of the designer of the algorithm using the Kohonen NN to ensure that the clustering the network spontaneously performs, is the clustering the user needs done in the application. This constitutes, in our opinion, the weakness of the Kohonen based implementations. The unsupervised network will perform the function it wants to perform. It may not be a trivial task to verify that the task performed by the network is the task needed by the application, in our case the static security assessment.

A number of papers [78], [79], [80], [81], [82], address the issue of data preprocessing, before data are presented to the Neural Network for learning and recall. Some designers

view data preprocessing as a way to improve the NN learning speed. It is our opinion that data preprocessing is vital in a real-world implementation of the on-line static security assessment, where field learning data are by far not as abundant as simulated data, the result of which can be a NN that learns very well by memorizing and therefore does not generalize. The mentioned references use Principal Component Analysis (also called Karhunen Loe've expansion) as the preprocessor of choice. While this approach may not be optimum, it is working reasonably well. Sidhu [83] presents a rather unusual approach that uses the Fast Fourier Transform (FFT) and applies it to the power system state vector (active and reactive powers). The paper treats the state vector components as a digital sequence signal and computes its FFT. Of course, by reordering the components in the sequence, the FFT will be different. We feel that:

- The order of the components is determined by the numbering of the power system buses. However, this arbitrarily decided order unduly influences the preprocessor's performance.
- There is no physical meaning to the ordering of the state vector components. The FFT, on the other hand, attaches significance to the order in which the digital signal samples are presented.

We feel that the reason the FFT approach works in [83], is due to the relatively high correlation between the state vector's components. This means that there is a rather strong "DC" component in the "signal" (which is the average of all samples). Of course, when the FFT computes the DC value, the order of the signal samples is irrelevant, since the average has no time (independent variable) dependency. We feel that the FFT may not do just as well de-correlating the "signal" components present in its "harmonics".

The following section will present a novel preprocessor designed to mitigate the scarcity of field data as well as reduce the input correlations.

4.1.2 Proposed Artificial Intelligence Based Development

There are several challenges a Neural Network based system designer must face when addressing the static security assessment problem. The first one is the NN structure. Most published work [70], [71], [83], [84] use one NN and compute the post contingency values for all contingencies that are considered. Some authors [82] use a separate NN for each contingency. We will contrast these two approaches and offer a combined solution.

Another challenge is the scarcity of real fault data on which the system needs to learn. The system must be able to learn to compute post contingency bus voltages based on real-time observed system conditions and not just on off-line simulations. It would be unlikely that the effect of a contingency could be observed under many operating conditions, a requirement for robust learning. This section will present a novel data conditioning method which will simultaneously deal with both, the lack of sufficient learning data and with the Neural Network based system structure [85].

4.1.2.1 Contingency Clustering Technique

Figure 4-6 shows a possible NN-based static security assessment engine. The model is presented with the pre-contingency bus voltages and generator and load active and reactive powers. Its task is to compute the post-contingency bus voltages. One Neural Network is responsible for learning the effect of a specific contingency. Jeyasurya uses this approach in [82]. Due to the large number of NNs, the number of weights to be determined is very large. Therefore, a large training set is needed.

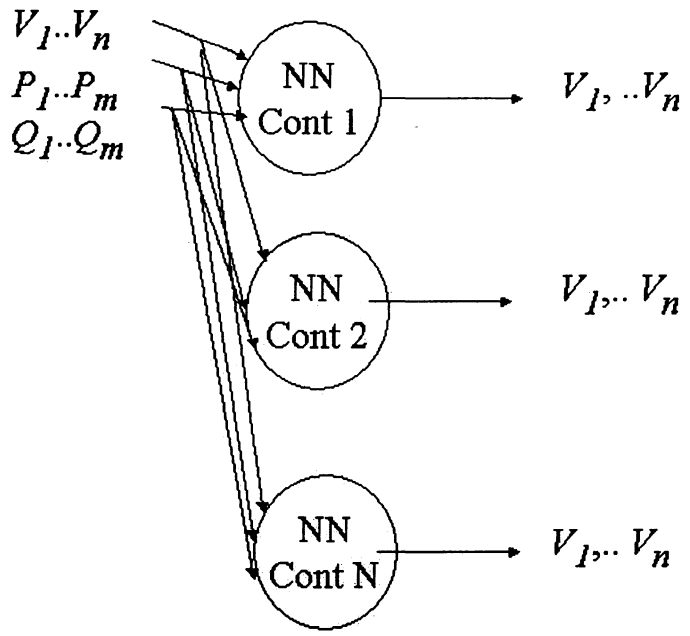


Figure 4-6 Post-contingency NN model. One NN for each contingency.

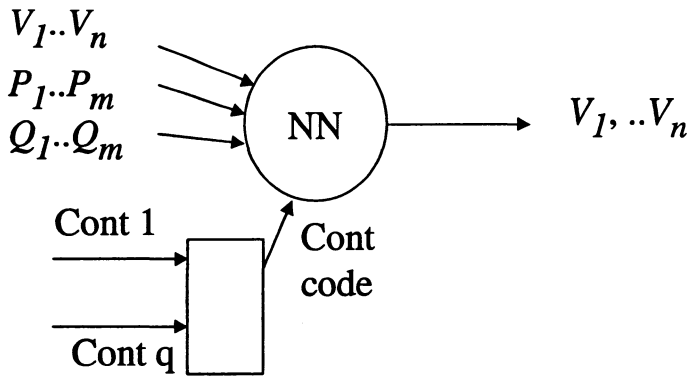


Figure 4-7 Post-contingency NN model. One NN models all contingencies.

To deal with this issue, Figure 4-7 depicts an alternative approach. As before, the pre-contingency bus voltages, and the active and reactive powers are presented to the, now one only NN. Moreover, a contingency code has to inform the NN which contingency is being applied. This code is computed by the contingency encoder shown in Figure 4-7. Most authors simply use the contingency number as a contingency code. One could assign one Neural Network (NN) for all contingencies and encode the contingency name

as an input into the Neural Network. The first disadvantage of this method is that the NN would have to combine the analog inputs presented to it (pre-contingency bus voltages, active and reactive powers) with contingency names. The contingency names must be translated into real numbers, and the NN will have to perform mathematical calculations with them (for instance, it may have to compute the vectorial distance between two input vectors or it may have to compute the post contingency bus voltages based on math applied to the numerical name of a contingency). The way the translation from a contingency name to a real number will be crucial to the success of the NN implementation and will affect the output accuracy; however this translation is artificial.

The second disadvantage of the method is that it forces the NN to simultaneously accommodate contingencies that may have drastically different effects on the power system. This results in a lower performance of the NN.

Because of this fact, it would be desirable to somehow group contingencies that have similar effects on the bus voltages. If this can be done, the number of models needed to compute the effects of the different contingencies is lowered. Also, the model that represents a particular set of contingencies is expected to be more robust since it is based not only on a small number of training cases a particular contingency was involved, but on a larger data set supplied by all equivalent contingencies.

The contingency grouping can be done either manually by a power system expert, or automatically by non supervised clustering. The method that we will present is an automatic, unsupervised algorithm of contingency grouping, followed by the building of a set of Neural Networks capable of computing the post contingency bus voltages. Each Neural Network is responsible for modeling a group of equivalent contingencies. Our approach is to use the Partial Least Squares (PLS) [86], [55], [56], [57] method to build models, one model for each contingency. Each PLS model will be fed with the pre-contingency bus voltages, and active and reactive powers, and will be trained with the post-contingency bus voltages. The disadvantage of PLS is that being a linear method, when applied to a non-linear problem it has a lower recall accuracy compared to a non-

linear model. The advantage of the method is that it is very well behaved, being able to compute its parameters using as few as two vectors as a learning set (regardless of the input vector dimension). In contrast with PLS, a NN having too few training vectors would memorize during learning and exhibit a poor recall ability. The next step is to check how well a PLS model representing a contingency can compute bus voltages when used under different contingencies. If two models produce similar results, the contingencies they represent are considered similar and the field data sets each one has, are combined. We have now a contingency group. The process is again repeated. The process will keep performing iterations resulting in further contingency grouping. In the limit, the method will stop when all contingencies end up in one group, implying that all contingencies are equivalent. Of course, this is rather unlikely, therefore a contingency termination criterion is used.

The second step is the training of Neural Networks models. The topology used is the Feedforward Back Propagation Network. Just like the PLS models above, the Neural Networks are first trained using data from a single contingency. Therefore we are going to compute as many NNs as there are contingencies. Due to the small number of training cases, we will show that each NN learns extremely well, however it will have a very poor recall ability (the NNs memorize). The next pass will train as many NNs as there are contingency groups (the groups formed by PLS in the previous step, iteration 2). It is expected that the NNs which have access to learning data that have resulted from a group of contingencies, would have a better recall performance. We will show that the system of Neural Networks reaches an optimum when the number of groups is smaller than the number of contingencies but larger than 1. Too many groups result in NNs memorizing (not enough data for learning). Too few groups force the NNs to predict bus voltages under conflicting, non-equivalent, contingencies. The NNs will compromise.

The Static Security Assessment problem is a Multiple Input Multiple Output (MIMO) decision making problem. Both inputs and outputs are high dimensionality vectors, meaning that it is impossible to plot the output as a function of the input and show the performance of the NNs during recall. Because of this fact we will first show how our

approach can deal with a similar problem which is Single Input Single Output (SISO). This approach will also underline the fact that the presented method is general and can be applied to any application where an input/output relationship must be established and the input/output data can be grouped into somewhat equivalent regions. A separate model will be assigned to each region.

Following the SISO example, the method will be applied to the 10-machine, 39-bus New England Power System model.

4.1.2.2 Single Input Single Output Example

For our SISO example, we will consider 3 cycles of a sine waveform (Figure 4-8). Each cycle is split into 10 intervals, for a total of 30 intervals. Approximately 3 to 5 points (ie. a small number of points) are collected from each interval and stored in a learning set. The sampling position within each interval is random. Approximately 10 points are collected from each interval and will form the testing set. From each (x, y) point collected from a particular interval, the averages on the x and y axis, over that interval, are subtracted. The results are sections of the sine waveform having zero mean in both the x and y directions. Figure 4-8 displays the sections with the y offset removed only (if the x offset was also removed, then all sections would be plotted overlapping each other, making the figure hard to read).

The task of the method is to decide if any of the regions can be grouped (for example, the four points to the left of x=2 in Figure 4-8, region 3, could probably be grouped with the four points to the left of x=5, region 8). Once the grouping is done, a NN assigned to each group of points learns the input/output dependence, in other words learns to approximate portions of the sine wave.

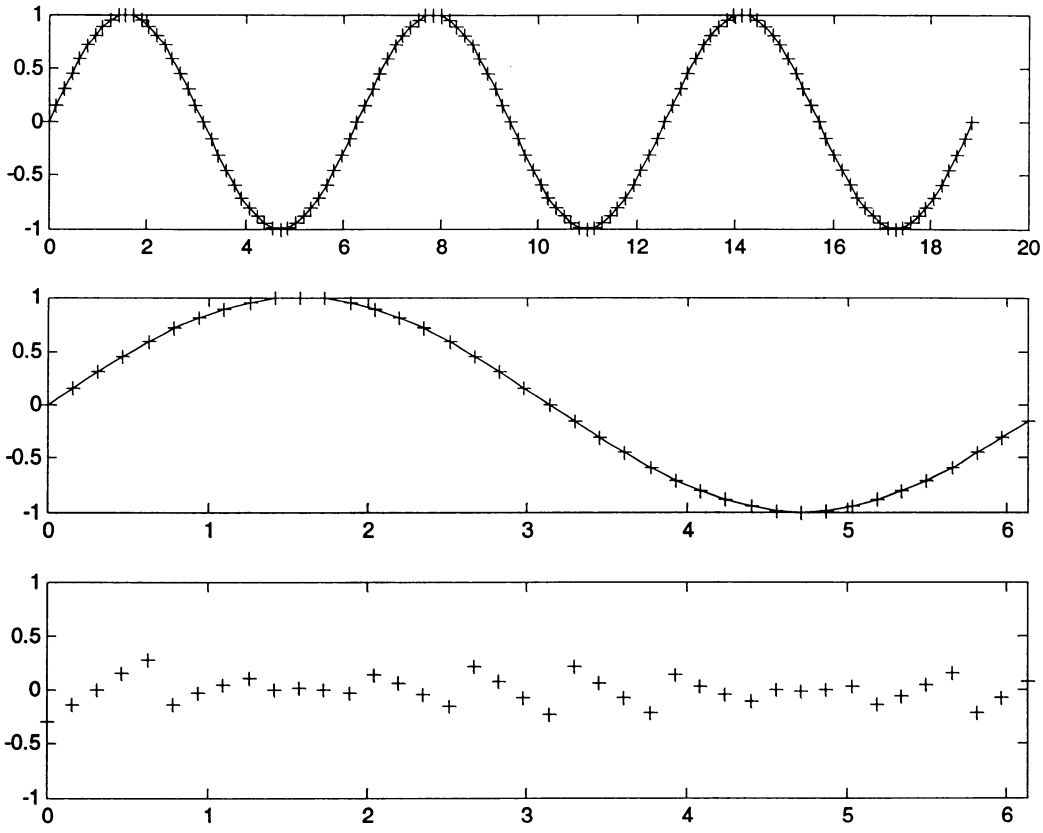


Figure 4-8 Sine waveform split into sections.

The algorithm starts by assuming that each group contains one and only one region (each region corresponds to a contingency in the Static Security Assessment problem). The region (contingency) number / group number relationship is shown in Figure 4-9. Figure 4-10 shows how well each model can predict data from all other regions. A white square indicates a good fit. A black square marks a poor fit. Figure 4-10 shows that squares on the main diagonal tend to have a better fit. This is to be expected since a model created with its own region points has a tendency to predict well points in the same region. In other words, model 1 which was created by learning points from region (contingency) 1, will hopefully be able to recall values from region 1 perhaps better than values from any other region on which it has not been trained.

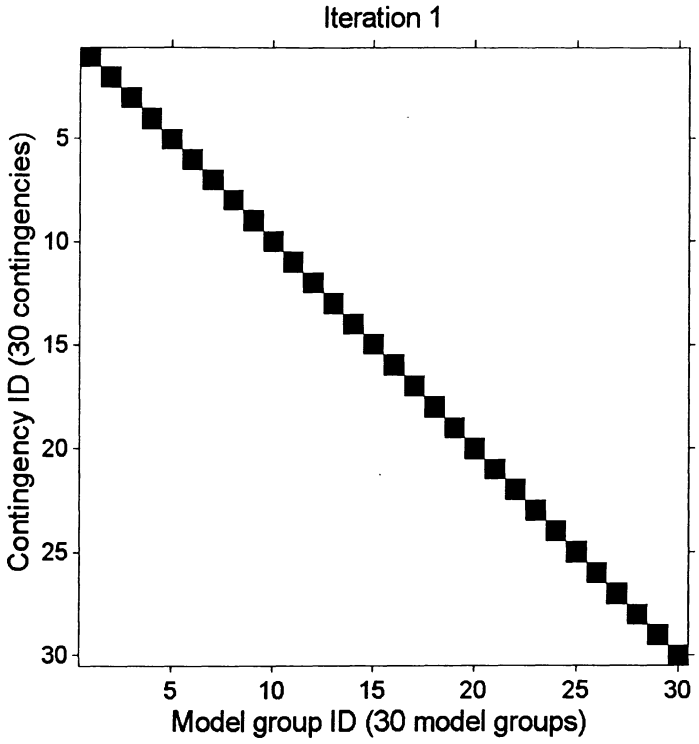


Figure 4-9 No grouping has been performed yet.

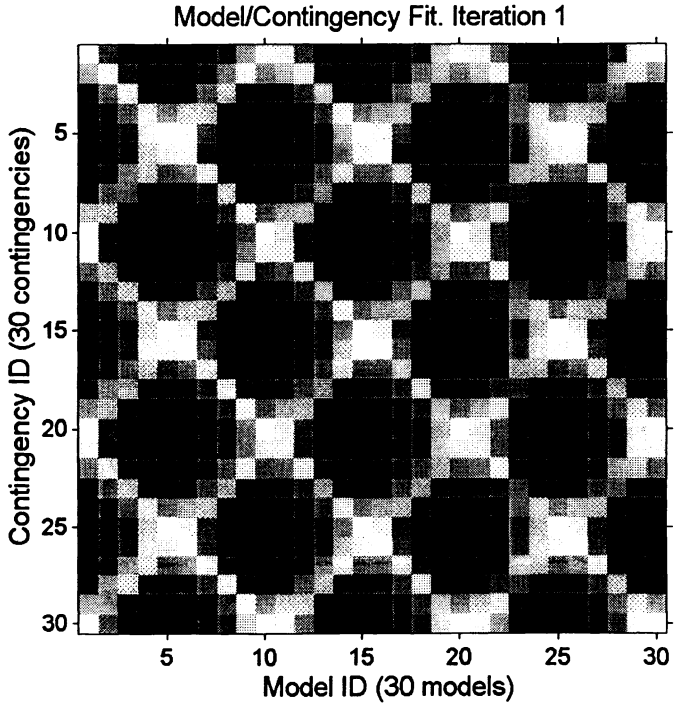


Figure 4-10 The goodness of fit for each model applied to each region.

Another point that can be made about Figure 4-10 is the symmetry exhibited around the main diagonal. This is expected since square (i, j) represents the goodness of fit of model j when it has to recall points from interval i. If the square located at (i, j) is a bright one, it indicates that model j can recall well points in interval i. Model j is a model created by learning points in interval j. If a model built by learning points in interval j can also recall well points in interval i, it is expected that the model built by learning points in interval i, will work equally well on recalling points in interval j, hence the symmetry around the main diagonal.

A third point to be made regarding Figure 4-10, refers to the patterns that are apparent. These patterns are caused by the periodicity of the sinusoidal waveform, therefore, in general, these patterns will not appear in other grouping and modeling examples.

Figure 4-11 shows the original sine waveform, the approximations computed by the 30 neural networks models, as well as a staircase indicating the NN model number that is used to approximate the original sine waveform (one step refers to a single model, step at height 1 shows model 1 application zone, step at height 2 shows model 2, etc.). As can be seen, a significant number of points are poorly approximated. This is caused by the fact that not enough points (only 3 to 4 input / output pairs) were available for the respective NNs to be able to learn reliably.

In contrast with the NN models used for the waveform modeling, all models used to perform the region grouping are Partial Least Squares models. As indicated earlier, PLS models are linear. Therefore if the region of points contains a section of the sine waveform that is curved, then PLS will have a difficult time modeling. The strength of PLS is that it can build a model with as few as 2 input/output vectors, therefore it is well suited for situations where learning must be done using few vectors.

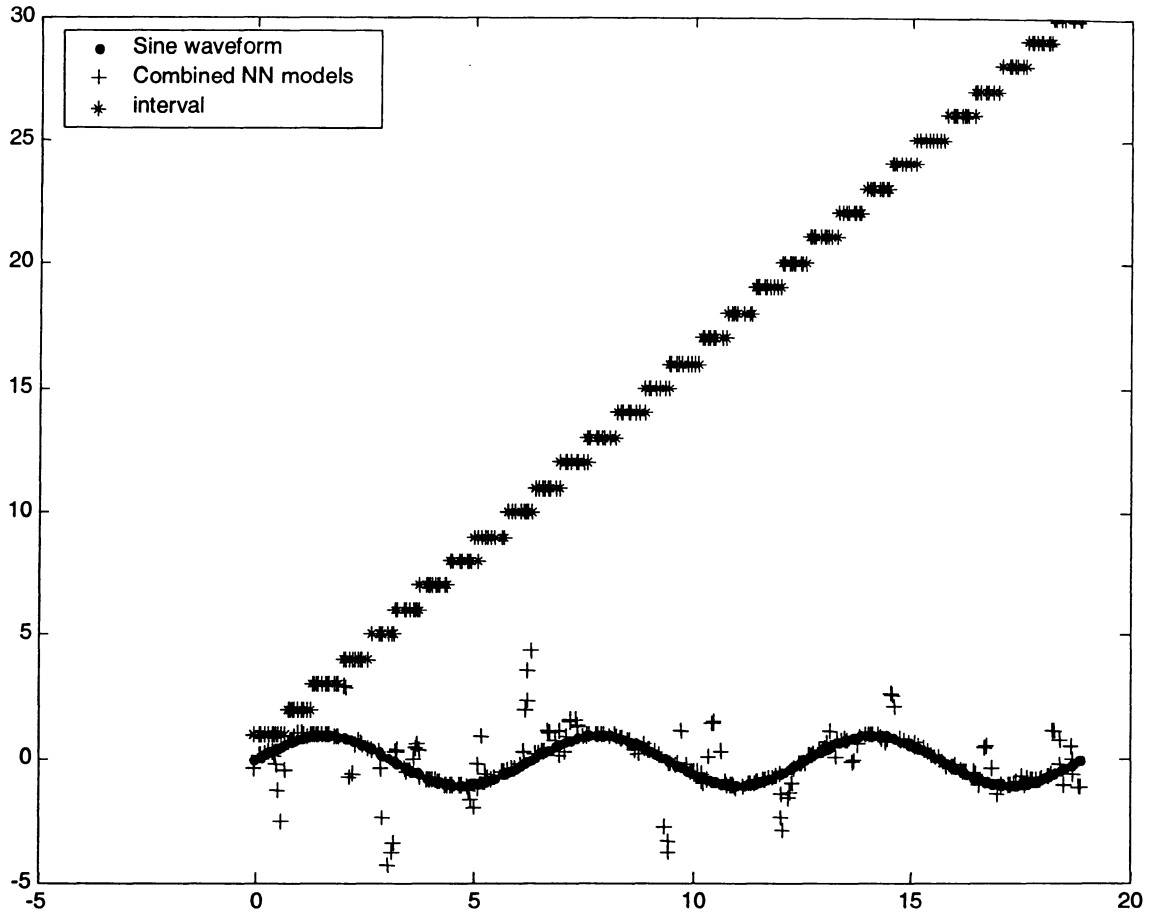


Figure 4-11 Iteration 1, Sine approximation using 30 models.

The next step of the method is to compare the behaviour of model i with that of model j. If model i, which was build on contingency i, can be replaced by model j, then model i can recall points from region j, and model j can be used to recall points from region i. Graphically, this fact will be indicated by 4 white squares in Figure 4-10 located at (i, i), (i, j), (j, i), and (j, j).

Grouping equivalent models, results in the region grouping shown in Figure 4-12. Regions 1, 10, 20, and 30 are lumped together into group 1 (parts of the waveform around the zero crossing, positive slope). This finding is encouraging since the

waveform is a 3-cycle sinusoid, each cycle containing 10 regions. In other words, the grouping mechanism has discovered some of the periodicity in our example.

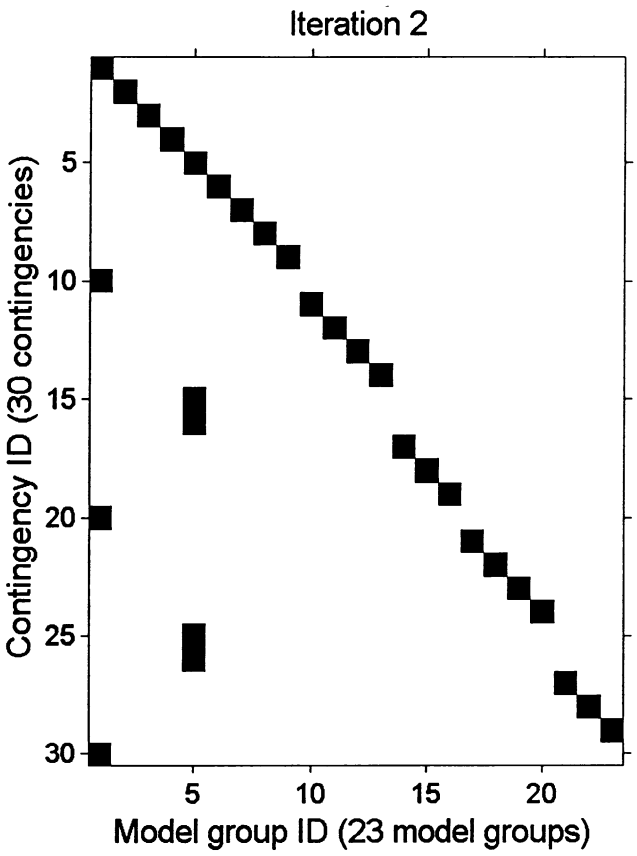


Figure 4-12 Region (contingency) grouping performed after first iteration and used in iteration 2.

Also grouping regions 5, 15, and 25 is caused by the same periodicity. These are the zero crossing, negative slope regions and are modeled by Model 5. It should be pointed out that the grouping algorithm did not group using periodicity as a grouping strategy, but did the grouping based on similar waveform sections. The periodicity grouping is only a very reasonable result of the approach.

Figure 4-12 shows that after one iteration, the number of models is now reduced from 30 to 23. This results in some models having more data available for learning, therefore potentially leading to a more robust model. Figure 4-13 shows the goodness of fit after 2 iterations. The main diagonal symmetry is still apparent.

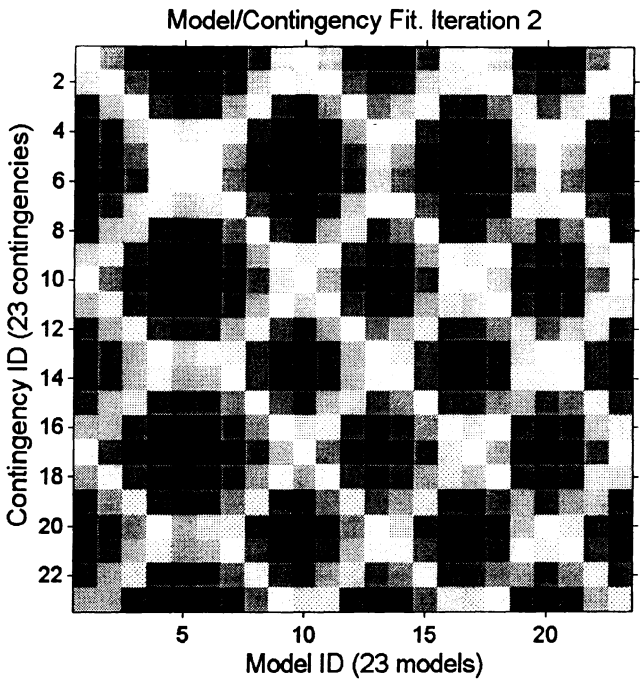


Figure 4-13 Goodness of fit after 2 iterations.

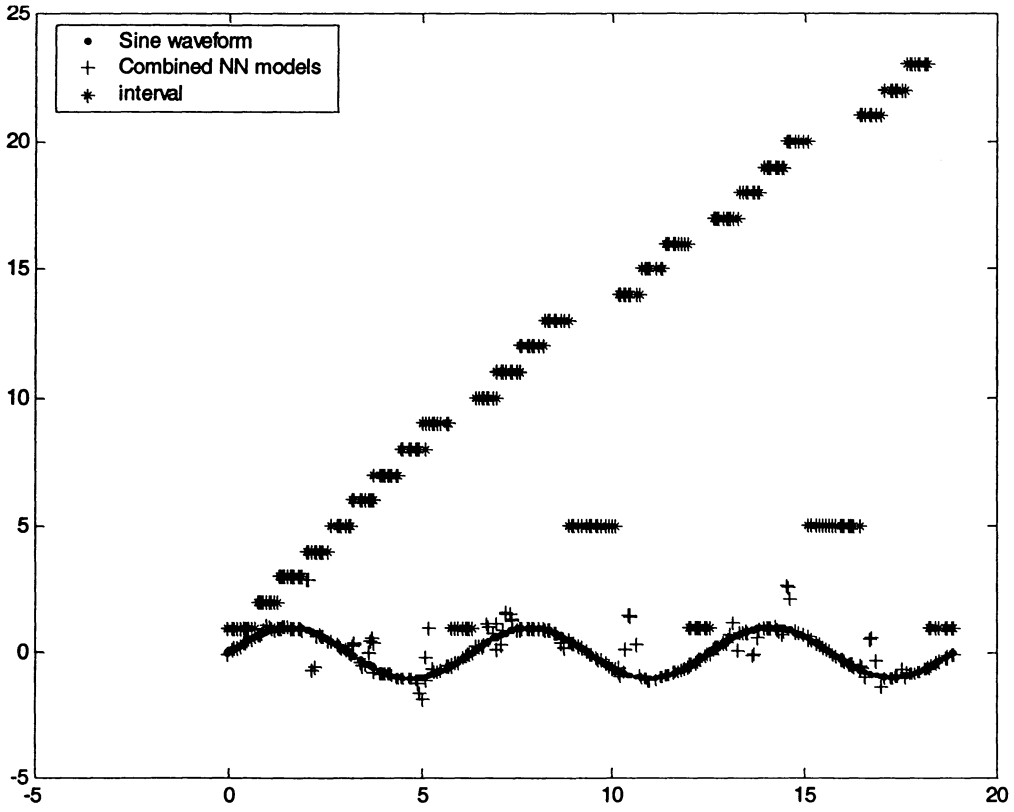


Figure 4-14 Iteration 2, Sine approximation using 23 models.

Figure 4-15 and Figure 4-18 show that after 7 and 10 iterations, the original 30 models are reduced to 12 and 7 respectively.

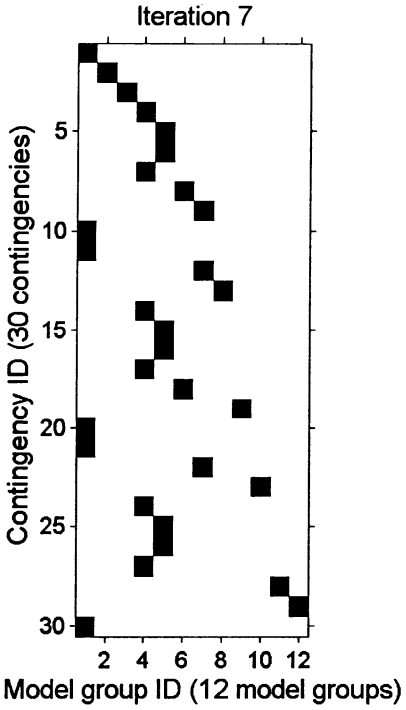


Figure 4-15 Grouping used during iteration 7.

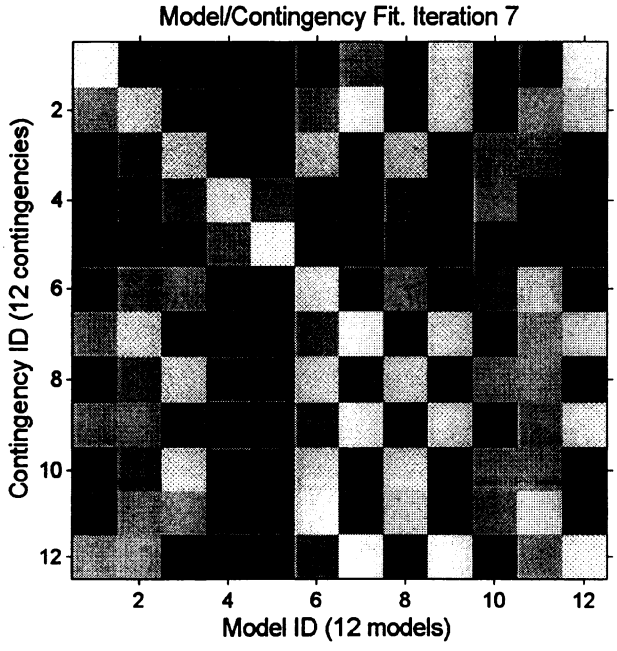


Figure 4-16 Goodness of fit after iteration 7.

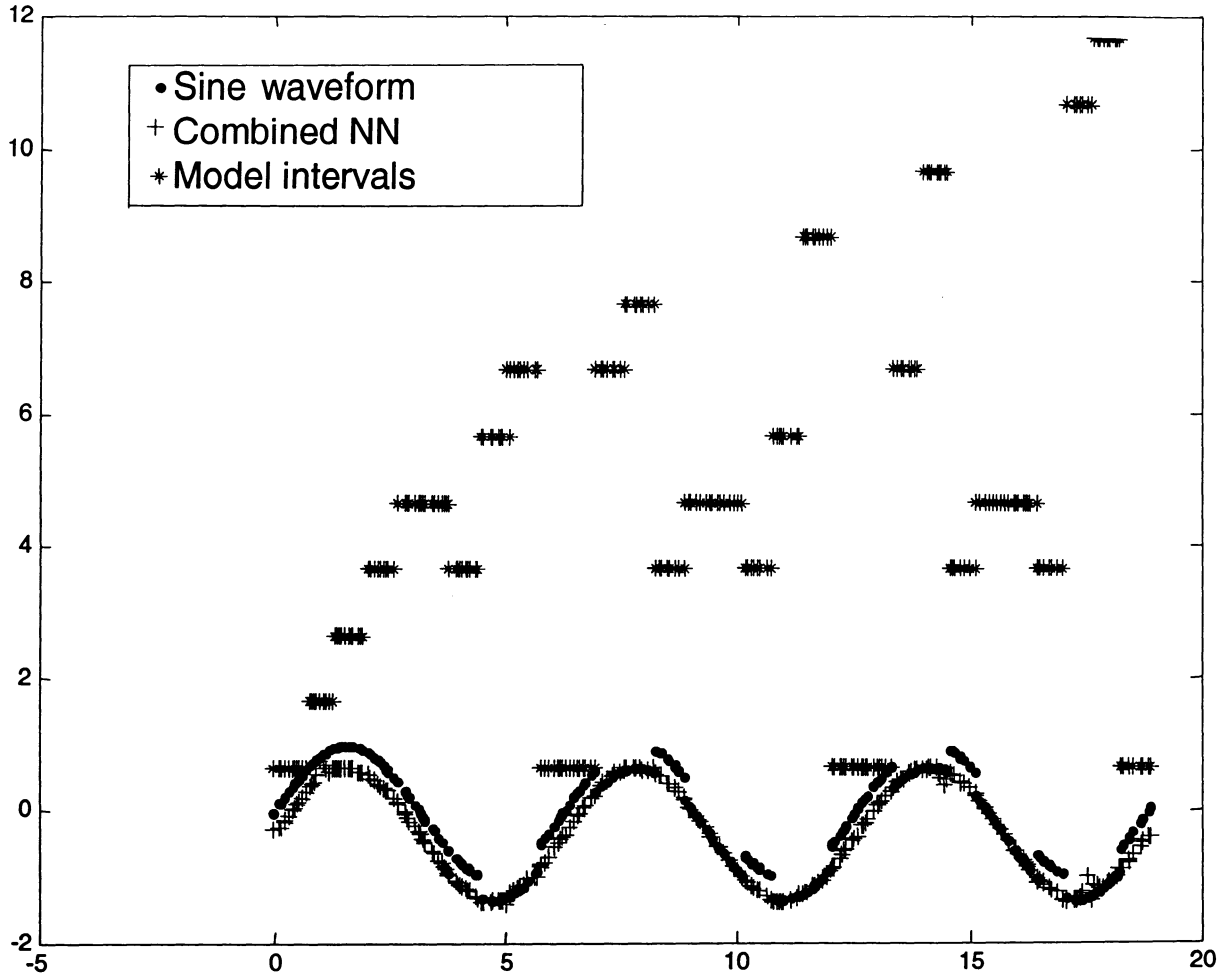


Figure 4-17 Iteration 7, Sine approximation using 12 models.

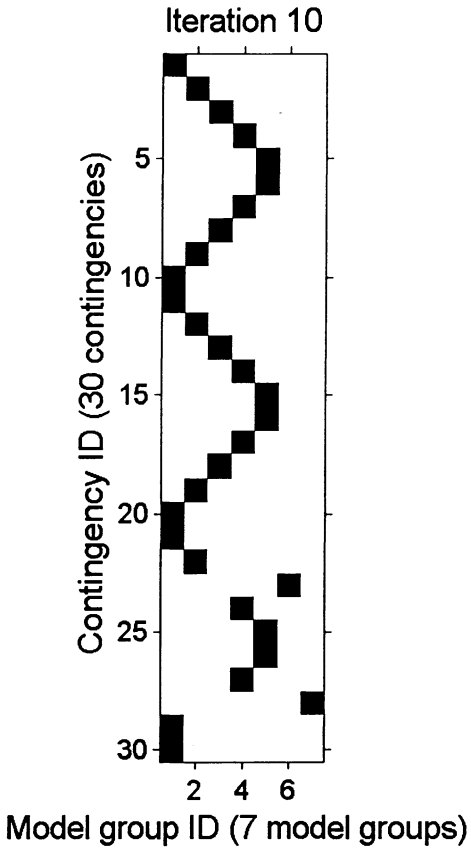


Figure 4-18 Region grouping used during iteration 10.

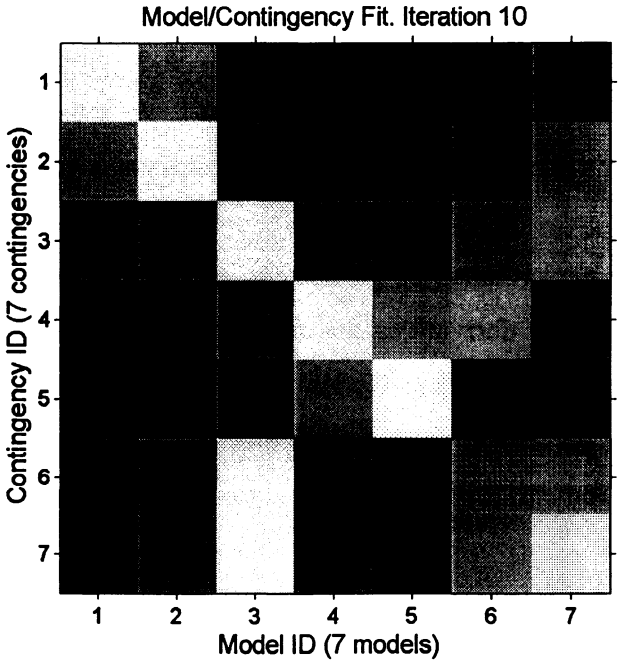


Figure 4-19 Goodness of fit after iteration 10.

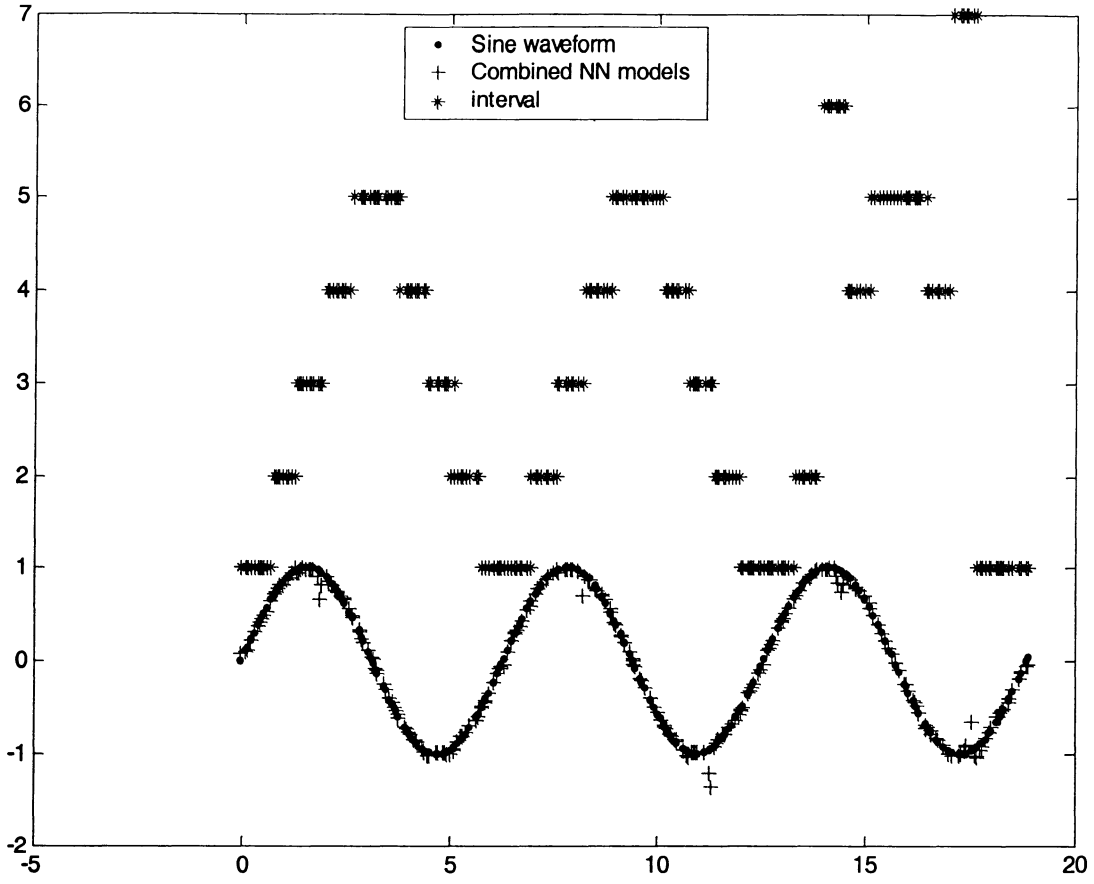


Figure 4-20 Iteration 10, Sine approximation using 7 models.

The result of an increased data set for each model, is a much better approximation performance compared to the one observed in iteration 1. A few outliers are still present in Figure 4-20, however the rest of the original sine waveform is very well modeled.

Figure 4-23 shows the performance of the system using only 2 models. It can be seen that all intervals of positive derivatives are modeled by Model 1 while the negative derivative intervals are modeled by Model 2. This clustering technique would be optimum if the original waveform were a triangular and not a sine waveform, however, given the 2 model requirement, it is the best any algorithm could do.

Finally, in Figure 4-24, by using only 1 model for the whole data set, the sine waveform is modeled by a staircase. The modeling error is large (a 1 model approach would be optimum if the waveform to be modeled were a square waveform).

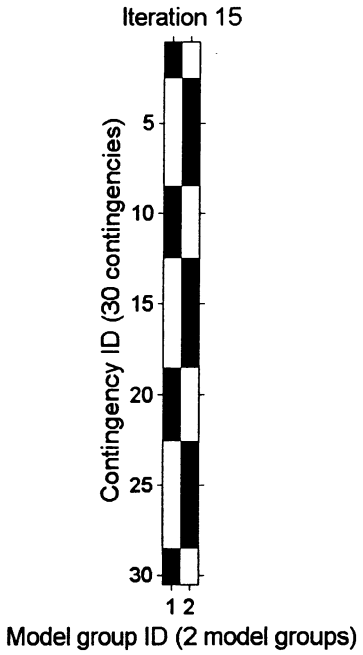


Figure 4-21 Grouping used during iteration 15. Only 2 models are used.

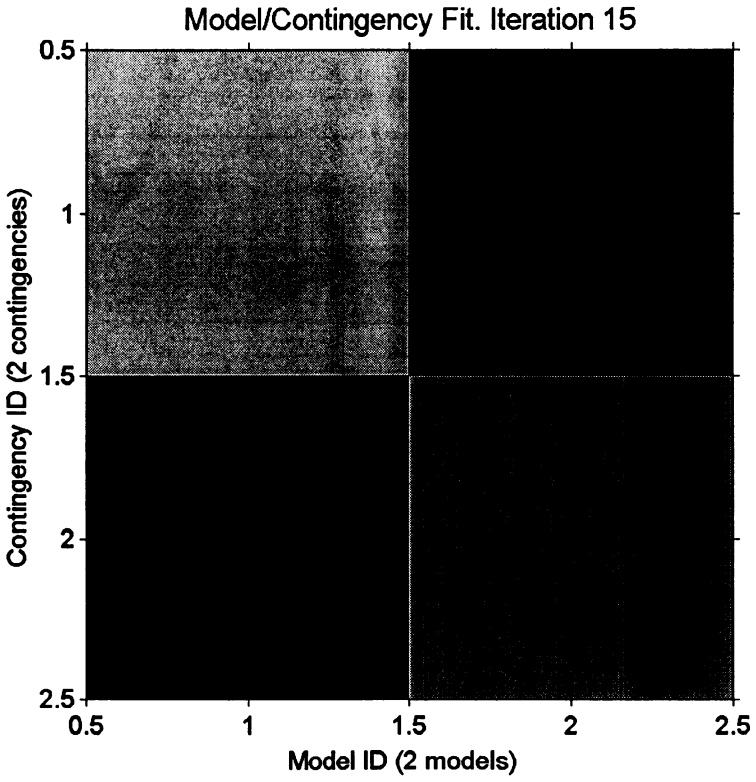


Figure 4-22 Goodness of fit using only 2 models.

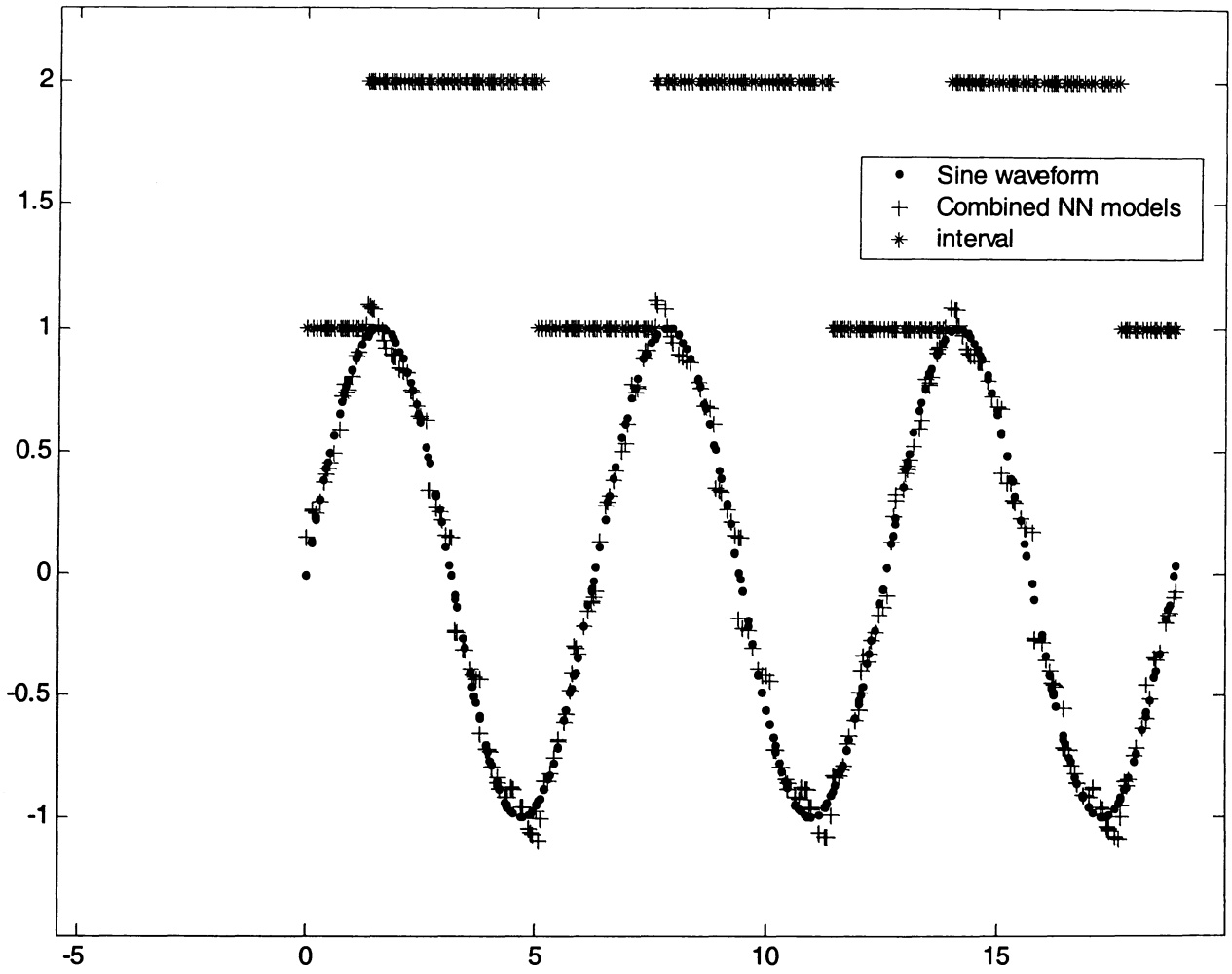


Figure 4-23 Iteration 15, Sine approximation using 2 models.

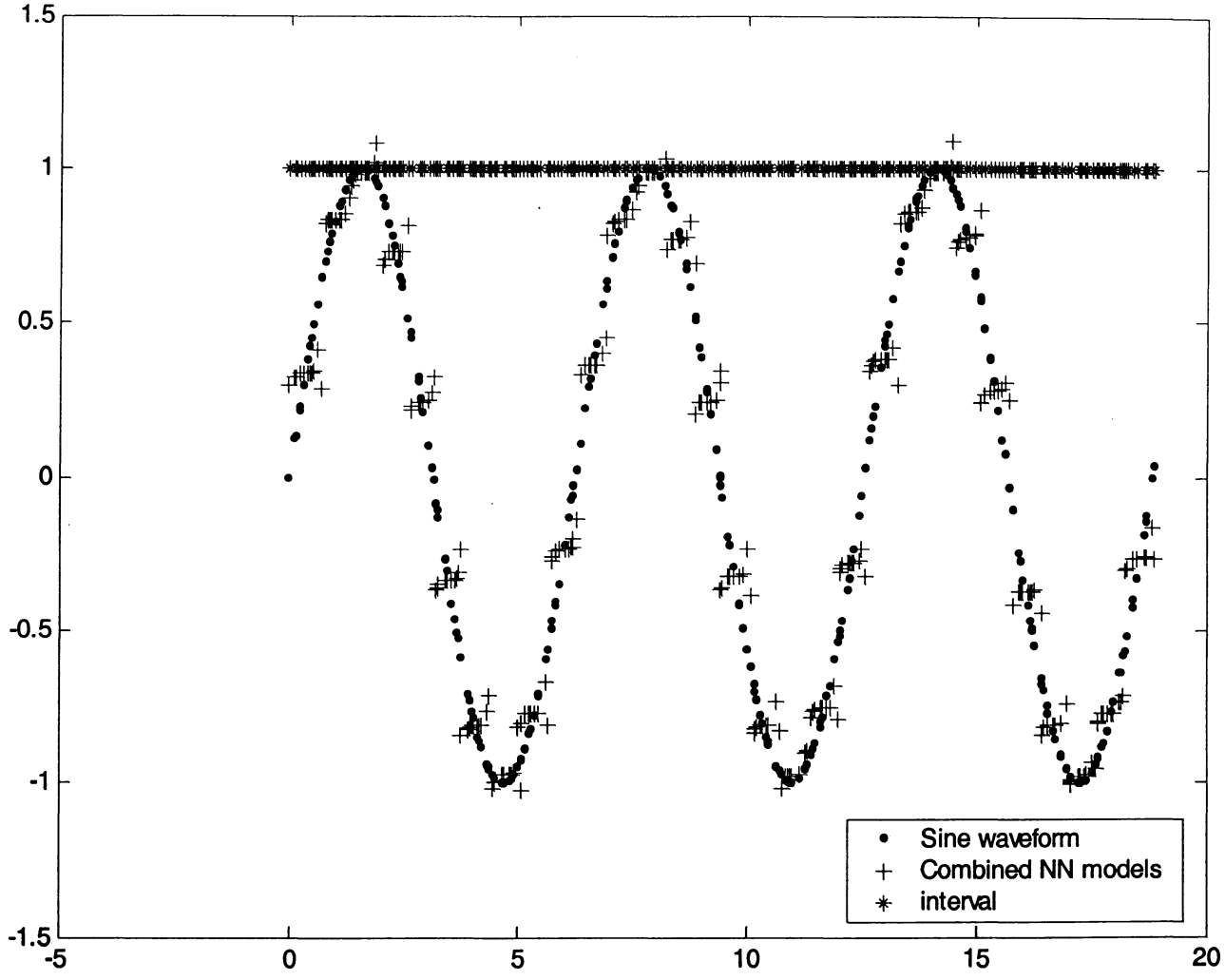


Figure 4-24 Iteration 16, Sine approximation using 1 model.

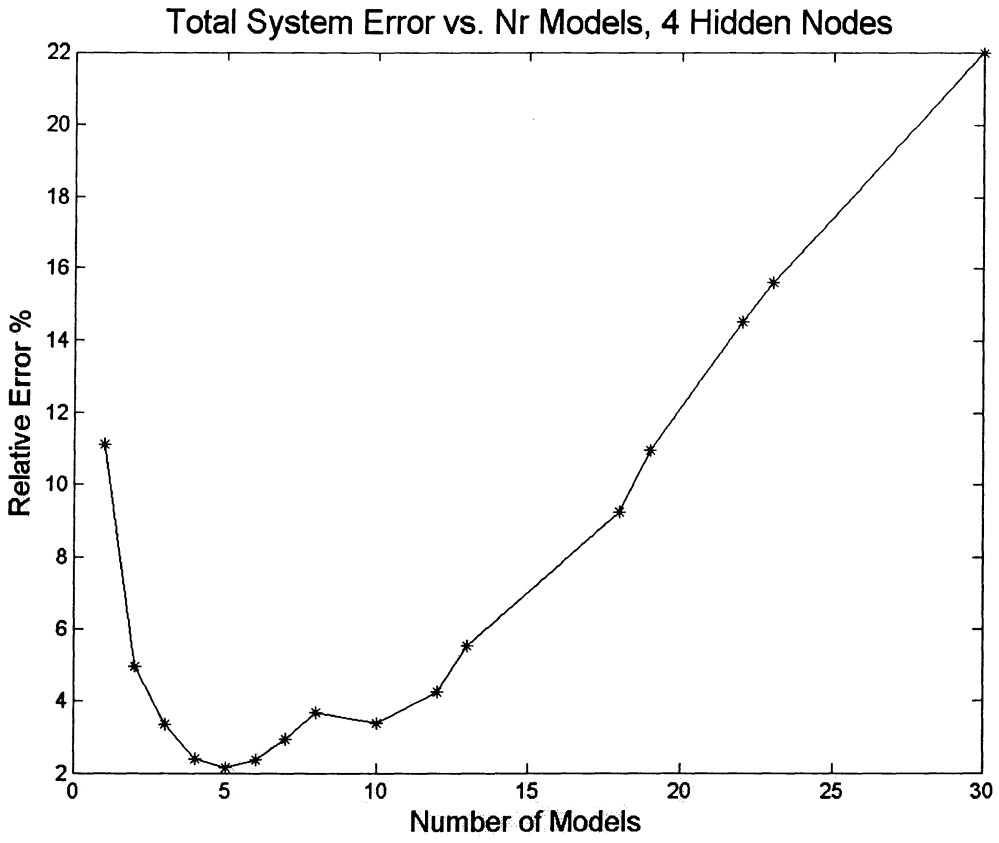


Figure 4-25 System error as a function of number of models used.

Figure 4-25 shows that an optimum is reached if about 5 NN models are used, each neural network having 4 hidden nodes. If more than 5 models are used, the scarcity of data in the training sets results in poor models that memorize and do not generalize. The recall error is large. If fewer than 5 models are used, the models are forced to compute incompatible values and the prediction error increases. For this particular problem, 5 models is the optimum solution.

Two important comments must be made at this point. The basic assumption of the data grouping method is that a decrease in the number of models used, decrease which has as a result a reduction in model flexibility (fewer models must approximate the same variation of data), is more than offset by the increase in learning data size each model will use. This is always the case. By merging two models into one, the resulting model

will learn on the larger data set made up of the original 2 data sets. However, using more data points does not always mean more information that leads to a better model.

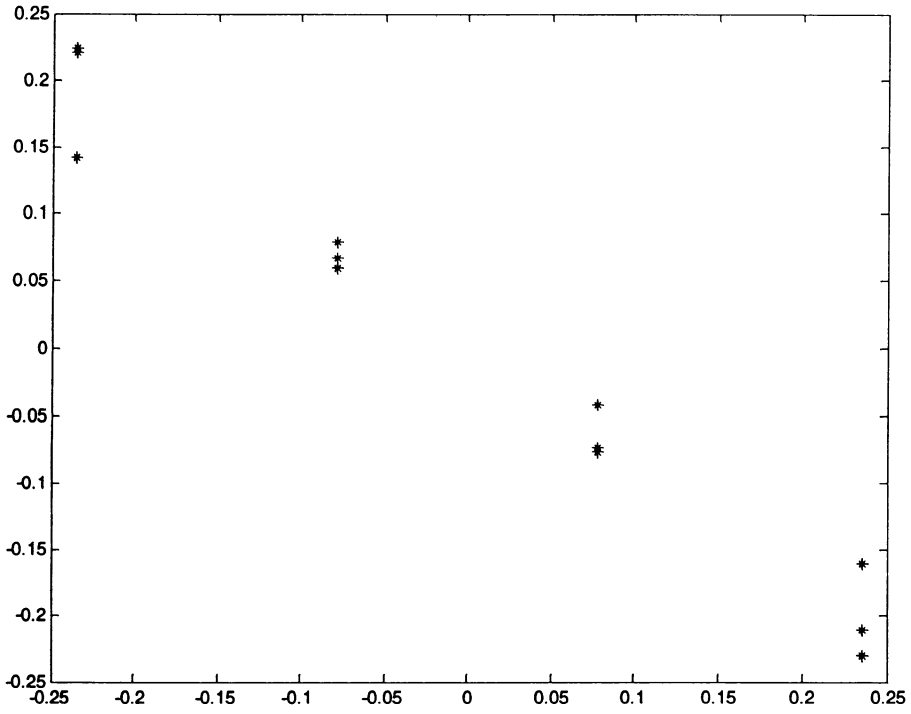


Figure 4-26 Sine values assigned to models 4,5,6,14,15,16,24,25,26 from Figure 4-11.

Figure 4-26 shows a large number of data points obtained from the negative slope portion of the sine waveform. The 32 points (most are overlapped) belong to models 4,5,6, 14,15,16, 24,25, and 26 shown in Figure 4-11. As it can be seen, the points are grouped in 4 vertical groups. A Neural Network with a large number of hidden nodes would oscillate between points regardless on whether only 4 points from one interval or all 32 points from 12 intervals are used. Bringing in more points by merging models does not improve the interpolation quality. The reason for the overlapping of points is the fact that the sine waveform was sampled using a fixed step size synchronized to the beginning of each interval.

Figure 4-27 shows data from the same intervals, however this time the data are uniformly randomly distributed in the independent variable. These data result in a robust model.

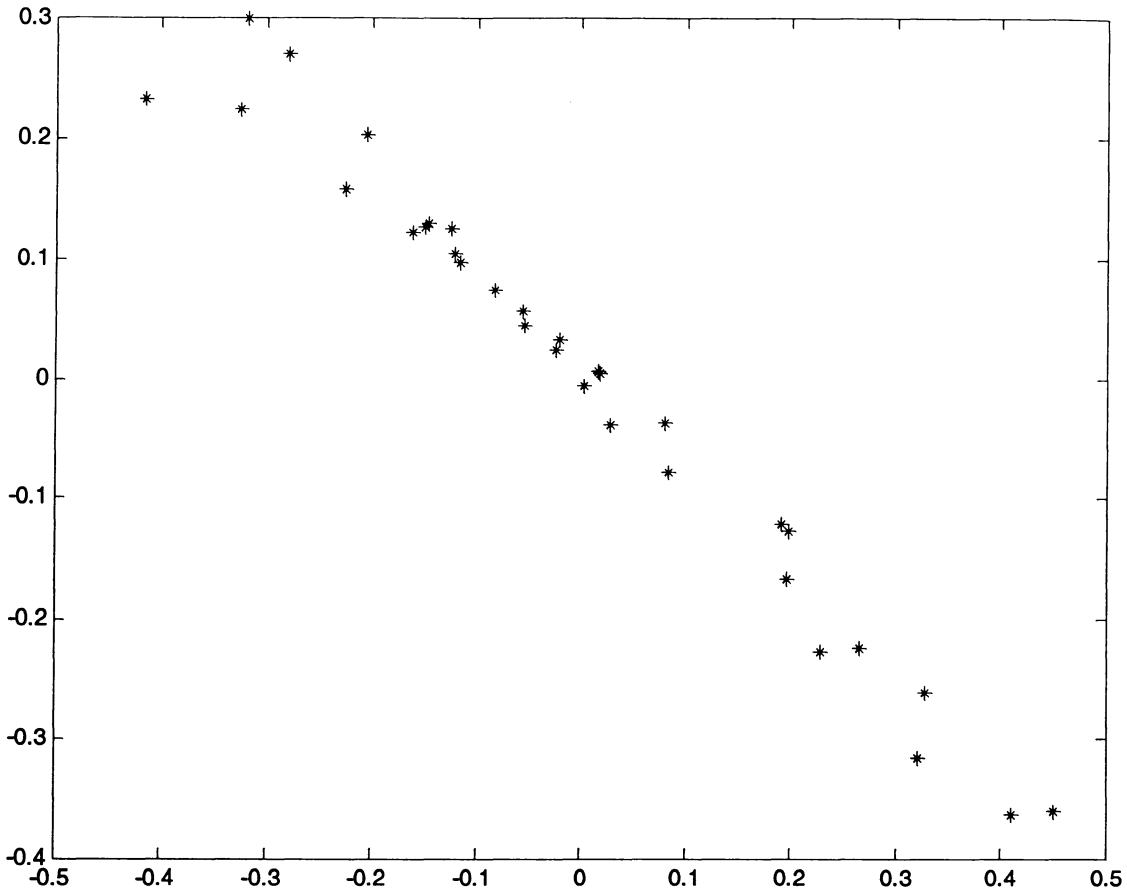


Figure 4-27 Sine values assigned to models 4,5,6,14,15,16,24,25,26 from Figure 4-11, sampling done randomly.

The second comment refers to the number of hidden nodes used by the NN models. An increase in the number of hidden nodes results in an increase in the NN flexibility.

More hidden nodes allow a NN to more easily approximate a nonlinear function. At the same time, an increase in the number of hidden nodes results in an increase in the number of parameters the NN uses. This means that a NN with a larger number of hidden nodes needs more data for learning. The converse is that, if only a limited amount of learning data is available, and if the problem is almost linear, a NN with a small number of hidden nodes will perform better than one with a larger number. One way to manage data

scarcity is to limit the number of model parameters to a minimum.

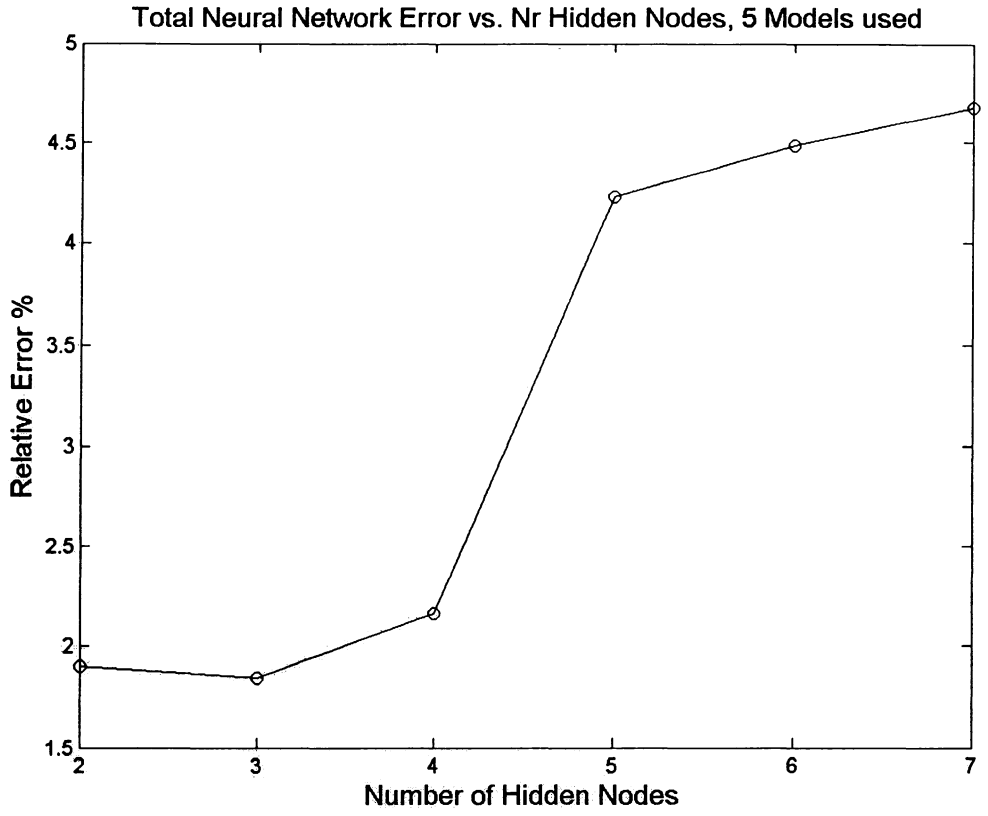


Figure 4-28 System error with respect to number of hidden nodes.

Figure 4-28 shows the system error as a function of the number of hidden nodes while keeping the number of models to 5. By using 2, 3,or 4 hidden nodes, a similar performance error is achieved, with the optimum around 3 hidden nodes. Of course, this number is application dependant. If the size of the intervals used to split the sine waveform was larger, the curvature of the waveform within an interval would be more pronounced and the number of required hidden nodes might have to be increased. Of course, more nodes result in a larger number of NN weights to be estimated, which require more data, resulting in more models needing to be merged. The presented model merging method seeks a merging optimum.

4.1.2.3 The IEEE 39-bus New England System Example

The simple SISO example showed how the presented new unsupervised data grouping algorithm achieves a larger learning set needed for robust Neural Network training. These concepts will now be applied to the IEEE 39-bus New England power system in the context of the static security assessment problem.

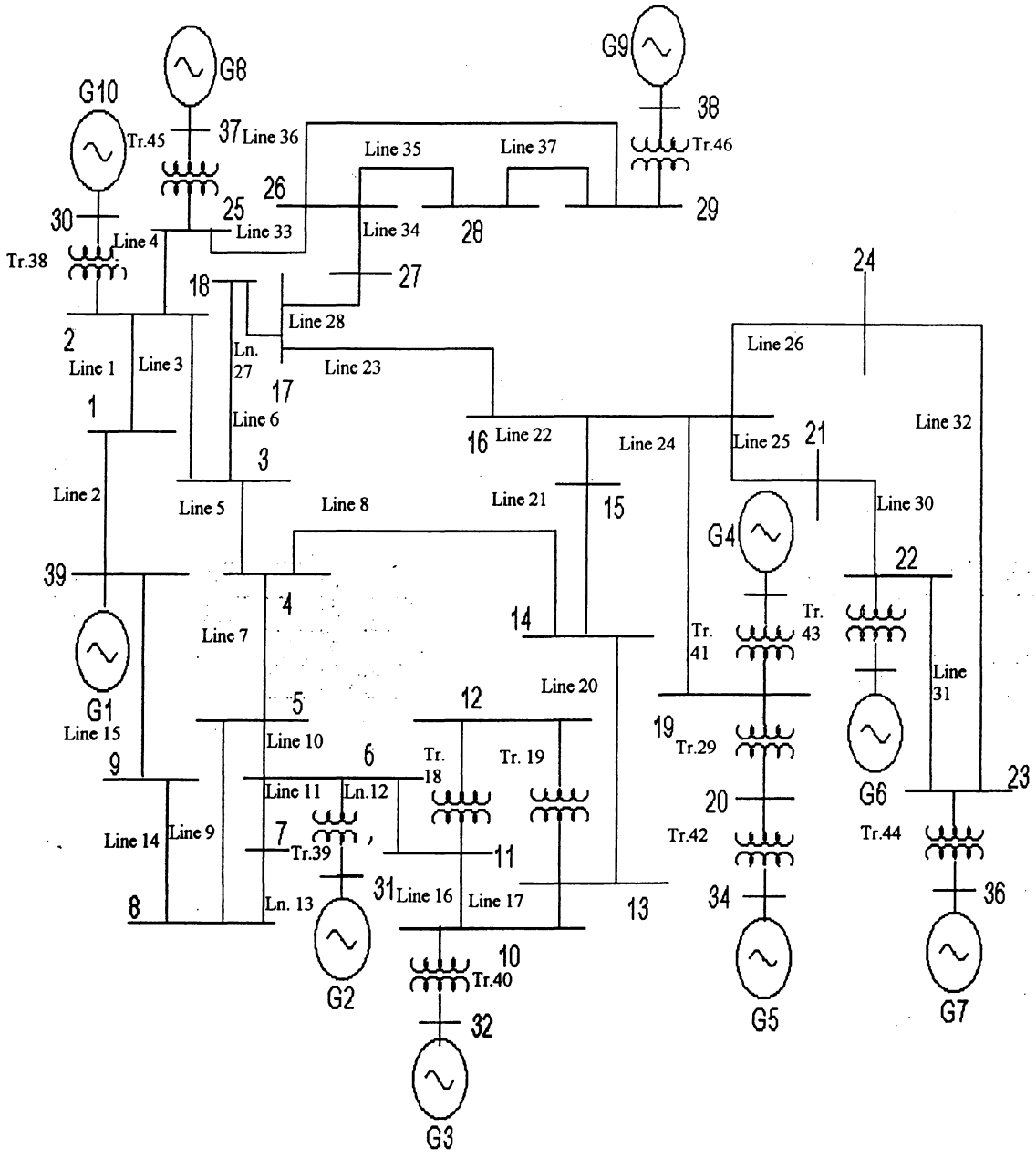
4.1.2.3.1 Data Generation

A number of load flows have been performed for each contingency in order to produce data that show the effect of the contingency on the power system. The loading is varied from 90% of base case to 110% in random increments of up to 10% for the learning set and from 85.5% of base case to 115.5% in random increments of up to 9% for the testing set. For each loading, generators outputs have been varied from 50% to 150% in up to 50% increments, one generator at a time for the learning set, and from 40% to 160% in up to 40% increments, one generator at a time for the testing set. The number of performed load flows was kept to a minimum in order to generate an extremely small learning set. Such a set has as few as 3 to 17 load flows for each contingency. Out of the total number of 46 possible contingencies, 8 contingencies have resulted in divergent load flows. Therefore, only the remaining 38 contingencies will be studied.

As can be seen, the testing set covers a larger load and generation ranges. The increment step sizes are also chosen to guarantee that no testing vector was used also for learning. Finally, the randomness in the step sizes was introduced to allow new information to be present in the additional samples and to counteract the phenomenon discussed in the previous section and shown in Figure 4-26.

The input vector presented to the static security engine contains 39 bus voltages (equal to the number of buses), 46 active powers, and 46 reactive powers (equal to the number of lines) for a total dimension of 131. The output vector contains 39 post-contingency bus voltages.

If only 1 hidden node is used in the NN responsible for modeling the effects of a particular contingency (131 weights from the input nodes to the hidden node, 39 weights from the hidden node to the output nodes, for a total of 170 NN parameters), a learning set of 3 to 17 cases is vastly undersized for robust learning, and the NN will just memorize. Once the number of hidden nodes is increased, the number of parameters to be computed increases dramatically. The small number of learning cases makes contingency grouping a desirable alternative.



10-machine, 39-Bus New England Test System

Figure 4-29 Ten machine, thirty-nine bus New England system.

4.1.2.3.2 Post-Contingency State Estimation

We can now apply the SSA contingency grouping and modeling method to the New England Power System. Figure 4-30 shows the initial contingency of the system. Each contingency has its own PLS and NN models. Figure 4-31 shows the system Squared Prediction Error (SPE). Just like in the SISO example, a white square indicates that a particular model fits well a particular contingency. In order to visually emphasize the difference in performance between models, the image in Figure 4-31 was altered using a process called histogram equalization. The process maintains the gray order between squares. A square "*i*" which was brighter than another square "*j*" but darker than a third square "*k*" will, after histogram equalization, still have an intensity level between "*j*" and "*k*". However, the absolute intensity values of "*i*", "*j*", and "*k*" will be altered in such a way as to provide optimal visual stimulus to the human eye.

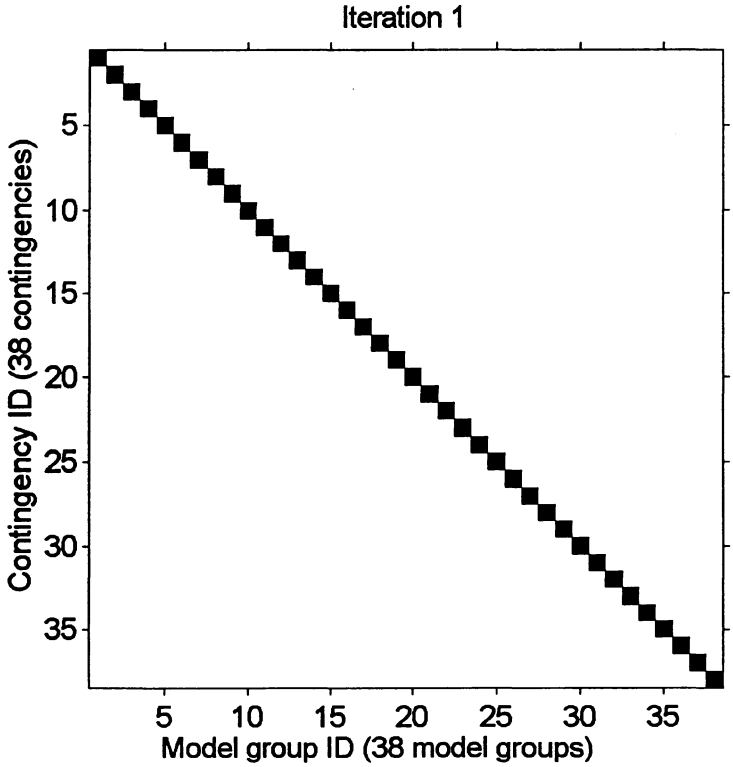


Figure 4-30 Initial contingency/model grouping for New England system.

A comment can be made about the SPE values in Figure 4-31. Just like in the SISO case, a symmetry can be observed around the main diagonal. This fact is expected since a

model "*i*" modeling well a contingency "*j*", will also model well contingency "*i*" (on which it was trained). Therefore, a model "*j*", which was trained on contingency "*j*", and models it well, is expected to also model well contingency "*i*", since contingency "*j*" is modeled well by both models "*i*" and "*j*". This fact is somewhat present in Figure 4-31, but not for all squares. This breaking in symmetry is caused by the poor recall performance of the trained Neural Networks. As mentioned before, these models are trained on insufficient data. Therefore, if contingencies "*i*" and "*j*" are equivalent, it is possible that the model trained on "*i*" can recall well values that belong to "*j*", however, the model trained on contingency "*j*" may not generalize well and be able to adequately predict bus voltages when presented with the effect of contingency "*i*".

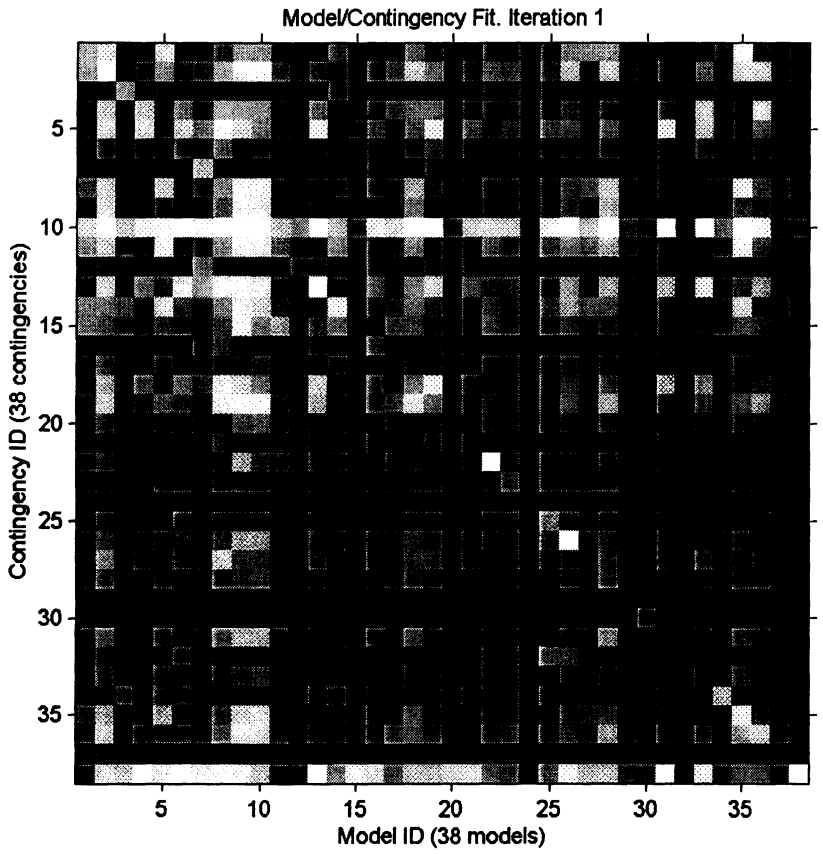


Figure 4-31 Squared Prediction Error (SPE) for New England system with no grouping.

Figure 4-32 shows the contingency grouping to be used before iteration 3. As can be seen, model 8 takes responsibility for 6 contingencies, while model 11 groups 3

contingencies. Since the grouping is not much different from the original one contingency per group, the SPE table is similar to the original one.

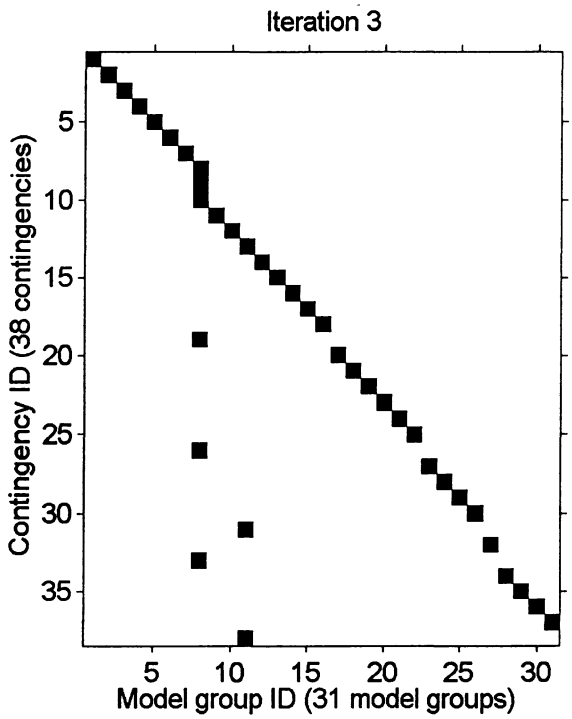


Figure 4-32 Contingency grouping for the New England system before iteration 3.

Figure 4-34 and Figure 4-35 show the performance of the system after 12 iterations. 8 models perform the modeling for the 38 contingencies. Examining Figure 4-34, one can see that models 5, 6, 7, and 8 are based on the original ungrouped contingencies 15, 20, 24, and 30. Therefore, these models had just as a poor learning set as the original ungrouped models. This fact explains why, in Figure 4-35, model 7 cannot accurately predict bus voltages after contingency group 7 occurs, in spite of the fact that model 7 was trained on contingency group 7 data. The results of iterations 14 and 16 are shown in Figure 4-36 to Figure 4-39. The number of models is further reduced to 4 and 2 respectively.

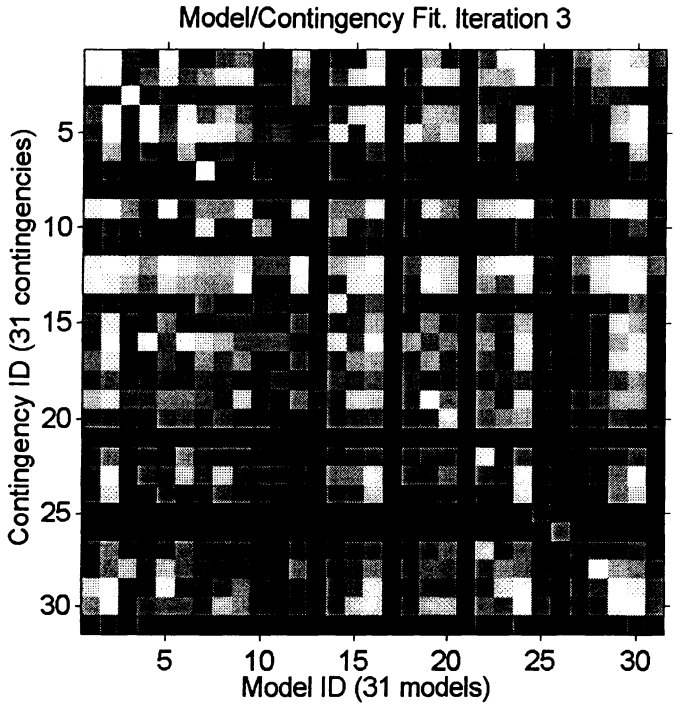


Figure 4-33 SPE for the New England system after 3 iterations.

Examining Figure 4-37, one can see that model 4 cannot forecast the data it has been trained on, contingency data group 4. From Figure 4-36, model 4 is exclusively based on contingency 24. The same situation takes place in Figure 4-39 where model 2 is based on contingency 24 (as per Figure 4-38). To see why contingency 24 cannot be grouped with any other contingency, we examine Figure 4-29, the diagram of the IEEE 39 bus New England system. Line 24 connects bus 19 to the rest of the system via bus 16. Both generators G4 and G5 are connected to bus 19 via transformers T41, T42, and T29. Therefore, if line 24 trips, the two generators become islanded. From this point of view, line 24 is a contingency like no other in the system. This explains why the contingency grouping algorithm was so reluctant to lump line 24 with any other piece of equipment in the system.

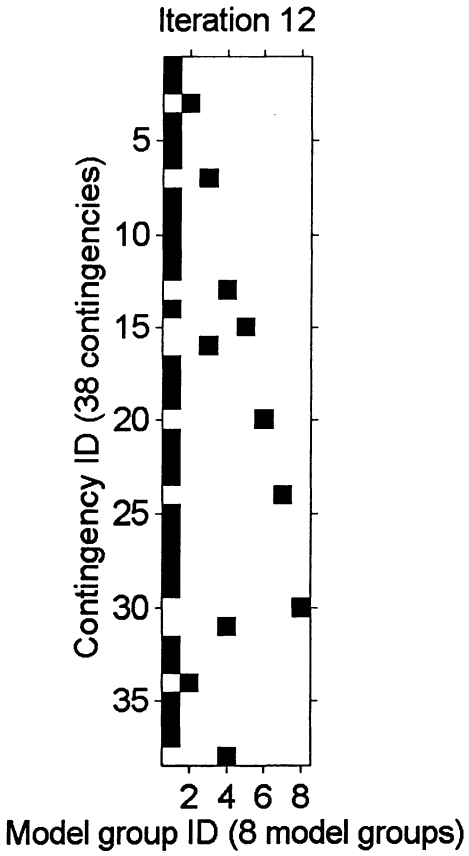


Figure 4-34 New England contingency grouping before iteration 12.

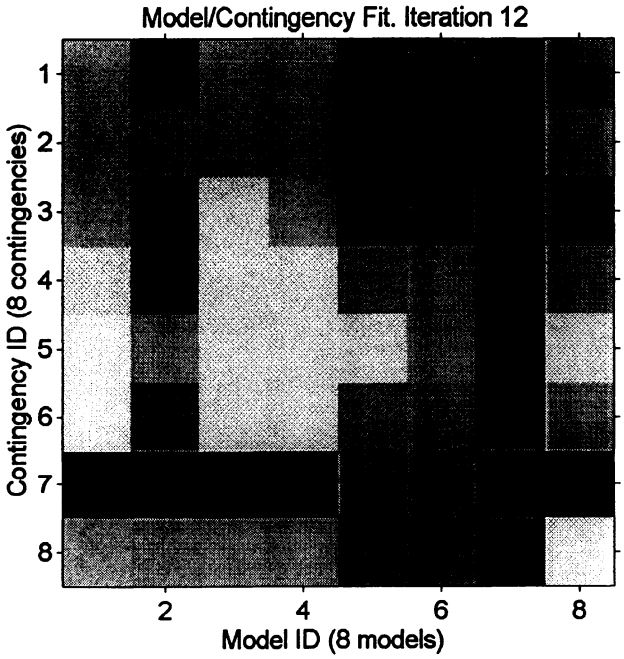


Figure 4-35 SPE for New England system after iteration 12.

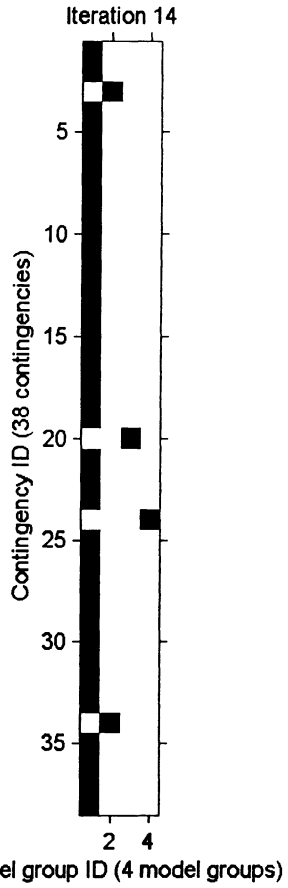


Figure 4-36 New England contingency grouping before iteration 14.

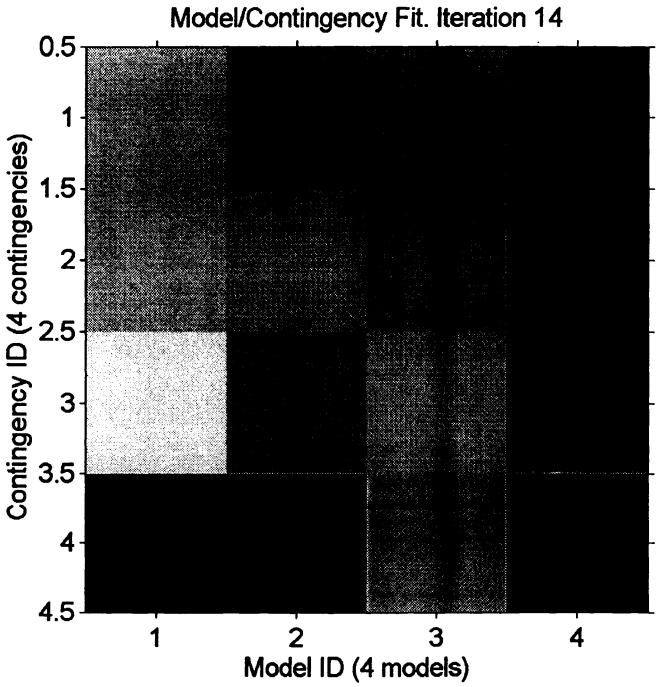


Figure 4-37 SPE for New England system after iteration 14.

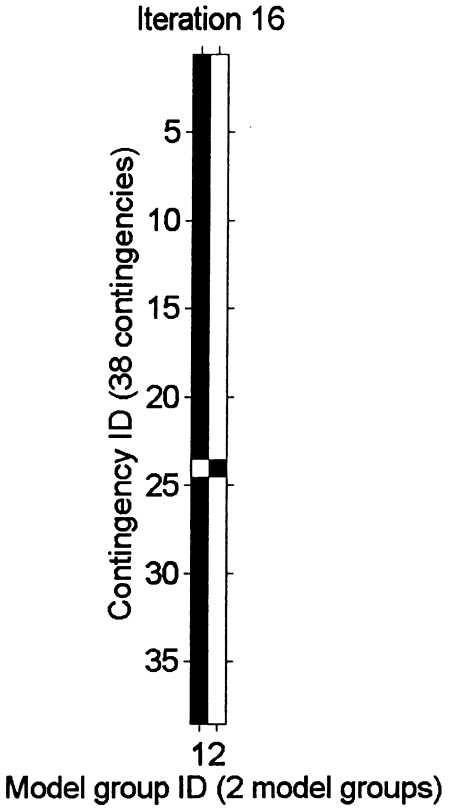


Figure 4-38 New England contingency grouping before iteration 16.

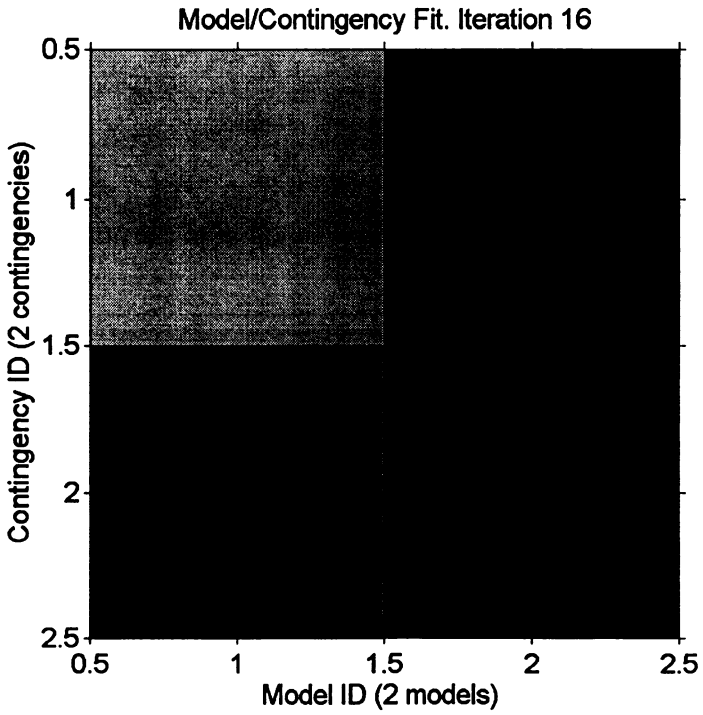


Figure 4-39 SPE for New England system after iteration 16.

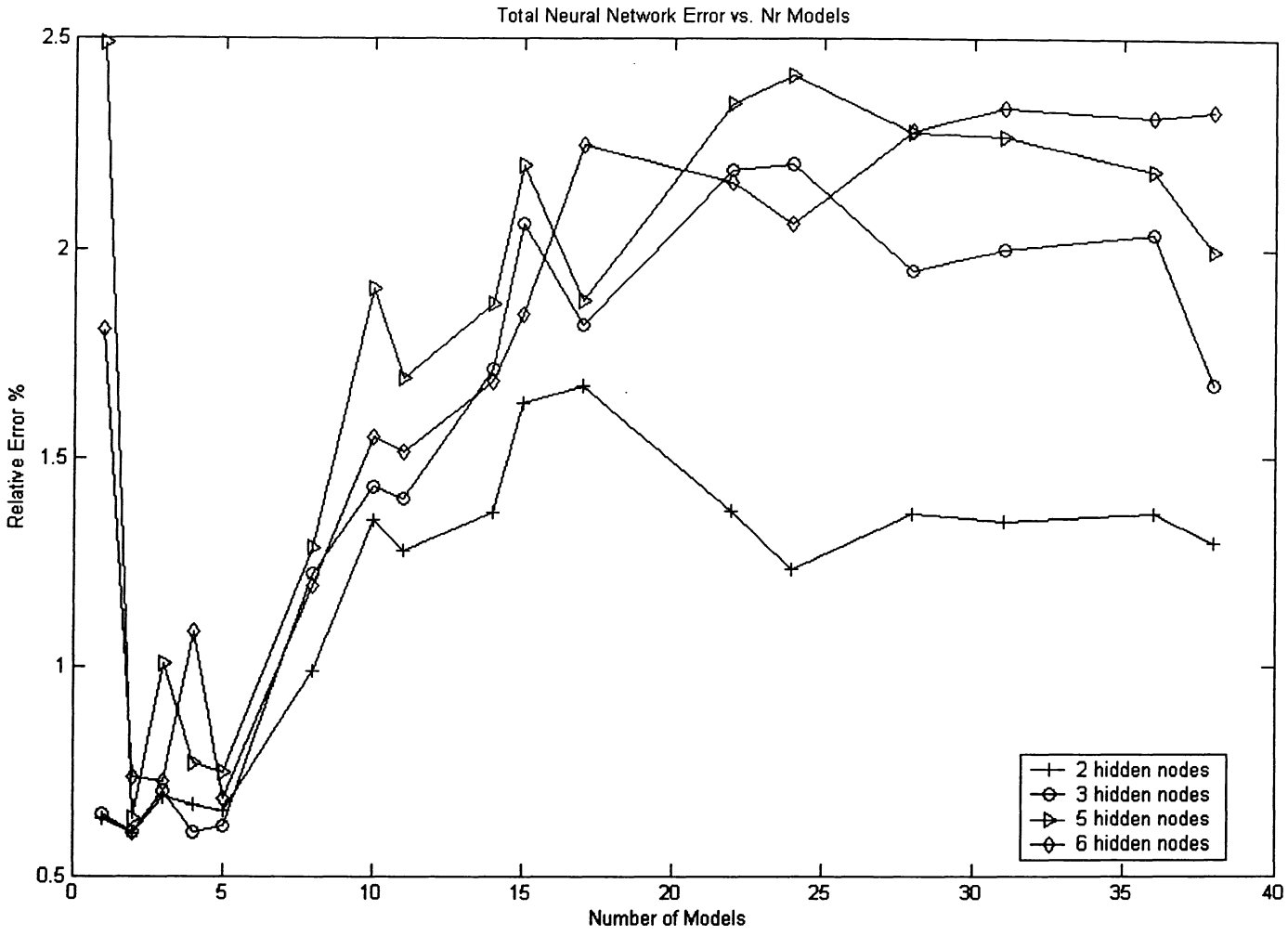


Figure 4-40 Total system relative error as a function of the number of models.

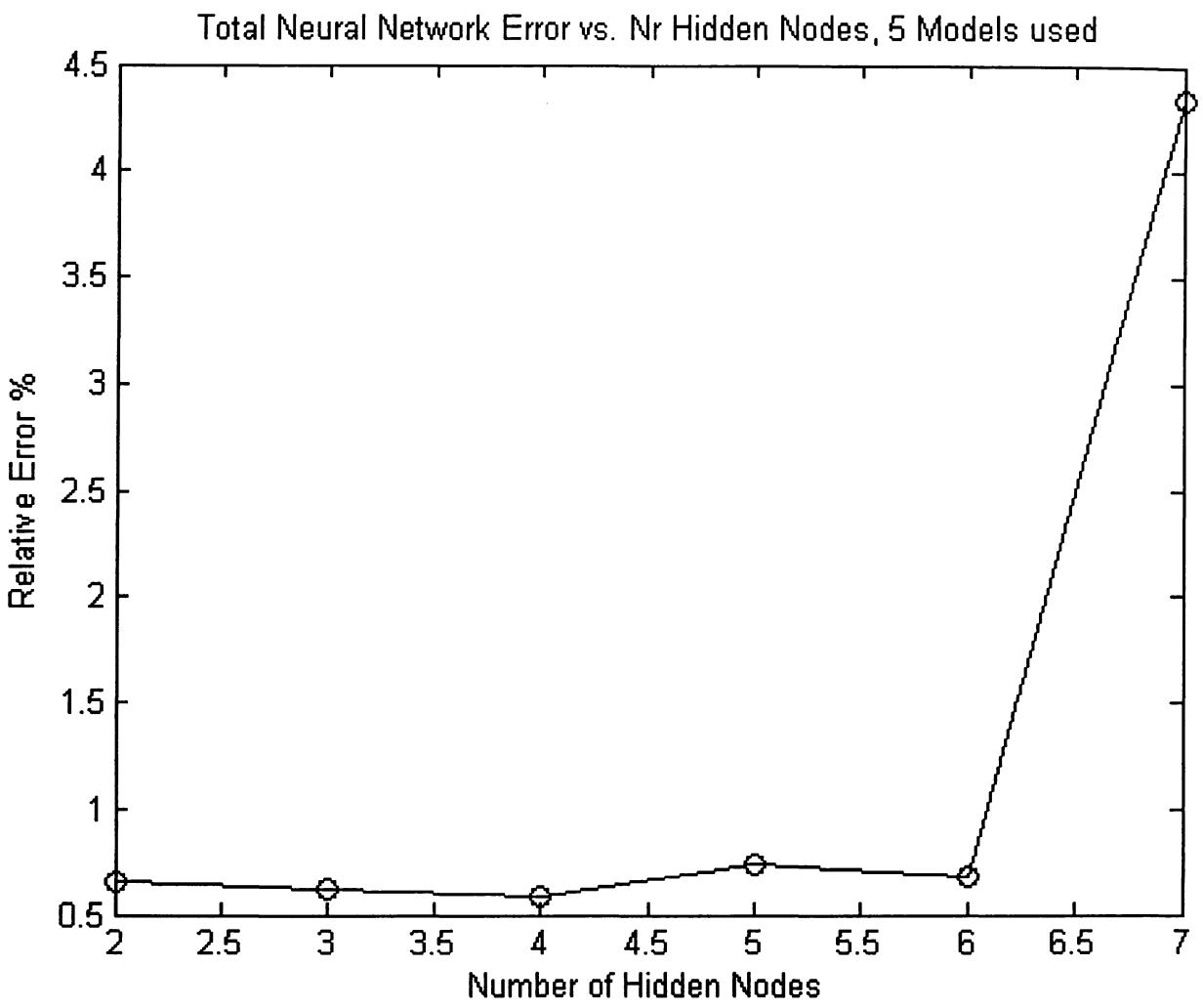


Figure 4-41 Total system relative error as a function of the number of NN hidden nodes.

Figure 4-40 shows the total system relative error as a function for the number of models used. The number of Neural Network hidden nodes is also varied as a parameter (all Neural Networks associated with a model group have the same number of hidden nodes). As expected, clustering the models from the original 38 to about 4 or 5 has resulted in a reduction in the total error.

Also, Figure 4-40 shows that increasing the number of hidden nodes results in a larger total error when 38 models (no clustering) were used. A smaller number of NN

parameters (hidden nodes and therefore weights) is an effective way to mitigate the small learning set available.

Figure 4-41 shows the dependence of the system error on the number of hidden nodes, for a fixed number of system models. It also shows that the optimum error is achieved if the used Neural Networks have 4 hidden nodes. The minimum is very shallow indicating that the added modeling flexibility of a 4 hidden nodes NN is probably not needed and that a 2 hidden node network would be preferable resulting in a sparser model with a similar generalization ability.

4.1.3 Discussion

This case study has shown how an unsupervised clustering algorithm, working in conjunction with a set of Neural Networks, results in a model builder. This approach is used in order to mitigate the effects of a very small learning set which is typical in the case of an On-Line Static Security Assessment problem.

Two examples are demonstrated. First, a Single Input Single Output function is modeled. A second example is the prediction of post-contingency bus voltages on the IEEE 39-bus New England power system. The procedure performed well in both examples. If model clustering was not be done, one would have two choices for model building:

1- build one model for each data interval/contingency
2- build a single model that is to be applied to all data intervals/contingencies

As shown, our approach is superior to both choices. One of the strengths of the method is its ability to perform the clustering, using very little data. The clustering allows the Neural Networks, which have heavier data requirements, to be estimated.

The potential weakness of the approach is rooted in the same area, the clustering stage. As discussed before, the clustering is done using Partial Least Squares, a linear parametric model. The linear aspect of PLS allows the method to be functional using little data, however, the modeling of a non-linear problem is less accurate. This may be a weakness in some applications if the inaccuracy of PLS leads to incorrect clustering. However, we expect that in many problems, beside the ones described in this case study, the achieved clustering will be accurate enough to allow the non-linear Neural Network to perform the modeling.

4.2 Oil Leak Detection In Underground Electric Power Cables

On occasion, High Pressure Fluid Filled (HPFF) underground cables develop small, hard to detect leaks. The public views any such event as a significant environmental issue and expects utilities to respond quickly to detect and fix the problem. A low cost system is needed to make monitoring feasible on all transmission cables. This system must be simple to use and reduce the need for long-term monitoring before a small leak is identified. Such a sensor-model detection system would use a minimum of operating parameters to detect leaks as small as 1 gal/hour.

A HPFF underground transmission line consists of a steel pipe which contains three high-voltage conductors. Each conductor is made of copper or aluminum, insulated with high-quality, oil-impregnated kraft paper insulation, and covered with metal shielding (usually lead), and skid wires (for protection during construction) [87]. The conductor and its wrappings are often referred to as "cables". The cables are surrounded by a dielectric fluid. The dielectric fluid is an oil at approximately 200 to 300 pounds per square inch, which saturates the kraft paper insulation. This pressurized fluid prevents electrical discharges in the conductors' insulation. The oil also moves heat away from the conductors.

The analysis of this case study will start by describing the published state of the art in underground cable leak detectors. Following this step, input and output quantities to be used for failure detection will be listed. The issue of availability of fault data will be examined and a data generator will be introduced. Finally, a number of Artificial Intelligence based implementations will be described, followed by a discussion outlining the strengths and weaknesses of the different approaches. A block diagram of a system that uses all the presented technologies will be shown.

4.2.1 State of the Art

There are a number of techniques currently used by power utilities to detect leaks in underground systems. One method uses tracers. Perfluorocarbon tracer (PFT) gases are placed inside the pipe system [88], [89]. Leak detection is achieved by detecting the PFT above-ground. The advantage of the method is that it achieves leak location without the need for digging. The disadvantage, is that it requires a network of sensors to be installed above the piping system, if it is to be used as a fault detection method, and not just for location.

Another conventional leak detection monitoring technique makes use of indicators for low oil tank level, frequent pump operations and low pressure [90]. In addition, leak detection is accomplished by comparing modeled flows to measured flows at the pumping plants. A divergence of modeled vs. measured flows indicates a fluid leak. While these indicators are useful in allowing a FDS to detect a leak, they impose a high implementation cost. Flow meters, in particular, are expensive to install, and require safety bypasses. Their accuracy, at low flow levels, may also be inadequate.

Our attempts will focus on implementing an oil leak detector, without using flow meters, or tank level indicators. The goal is to check the feasibility of a system that can explain oil volume, and hence pressure changes, solely based on temperature variations, while using a minimum number of sensors. We will use oil and soil temperature, as well as electric current measurements.

4.2.2 Input / Output Data

4.2.2.1 Data Analysis

For this case study, we have access to data from one field installation. The collected data are considered typical for this problem and is shown in Figure 4-42.

The quantities monitored at one or several locations, are:

- oil pressure
- oil pipe temperature measured by a themocouple glued on the outside of the pipe. The actual oil temperature is not available.
- soil temperature
- the RMS value of the electric current entering the cable at one end. No current phase information is available.

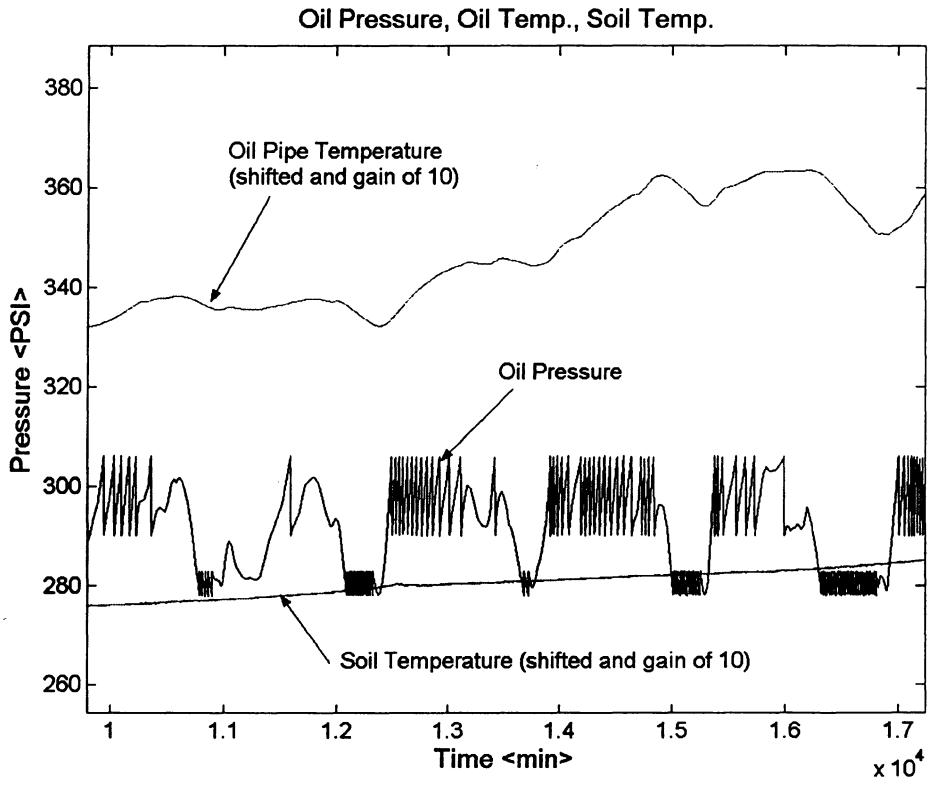


Figure 4-42 Typical oil pressure, oil pipe temperature, and soil temperature waveforms.

From Figure 4-42 it can be seen that the oil pressure is maintained between approximately 275 and 310 PSI. If the lower pressure limit is reached, an oil temperature reduction results in oil pump operations. Oil is pumped from holding tanks into the pipe

until the appropriate pressure is reached. Once the system reaches the upper pressure level, a further temperature increase results in pressure valves opening and allowing oil to spill back into the holding tanks.

As can be seen the oil temperature is the driver behind the pressure variation. Therefore, if an oil temperature / oil pressure model could be built, it would be useful in a fault detection system.

The oil temperature varies, depending on the soil temperature and electric current supplied by the cable. Therefore, another model useful to a FDS could be one that uses the electric current and soil temperature as inputs, and computes the oil pressure.

Finally, if an electric current / soil temperature to oil temperature model could be built, it also could prove to be useful for the problem at hand. The output of this model could be fed into a Kalman filter, which, together with the oil temperature measurements, would produce a better (less noise) oil temperature estimate. This estimate could be fed into the oil temperature to oil pressure model, in order to compute the expected oil pressure.

The correlation between the oil temperature on one hand, and the electric current and soil temperature on the other hand, is shown in Figure 4-43. In order to be able to plot all three quantities on the same graph having one axis, the two temperatures have been multiplied by 100 and have been shifted.

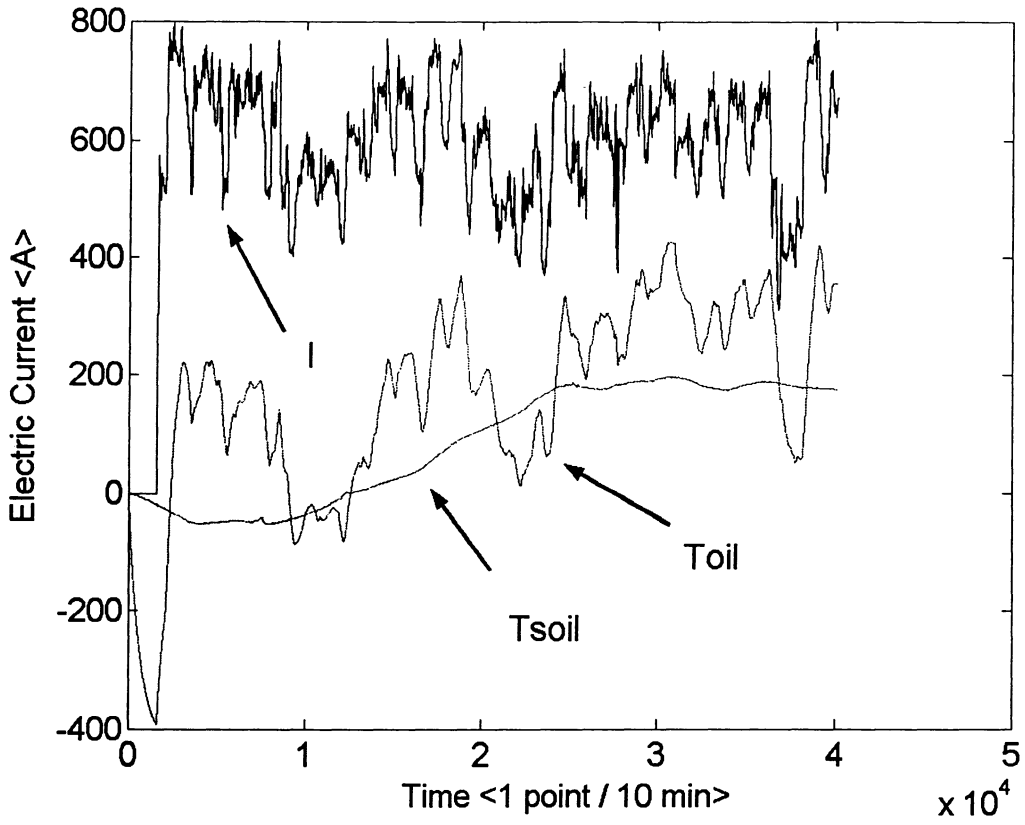


Figure 4-43 Electric current, oil, and soil temperatures. The two temperatures have been shifted and have a gain of 100 for plotting purposes.

In order to estimate the bandwidth required for non-aliasing sampling, a Fast Fourier Transform (FFT) of the electric current has been computed. We chose to use the electric current rather than the oil temperature, since we expect the oil temperature to depend on the integral of the square of the current.

Therefore the spectra of the two functions can be expressed by:

$$F\{Toil(t)\} = O\left(\frac{1}{s} F\{I^2(t)\}\right) \tag{4-32}$$

The spectrum of the oil temperature is expected to decay faster than the current spectrum because of the *1/s* term.

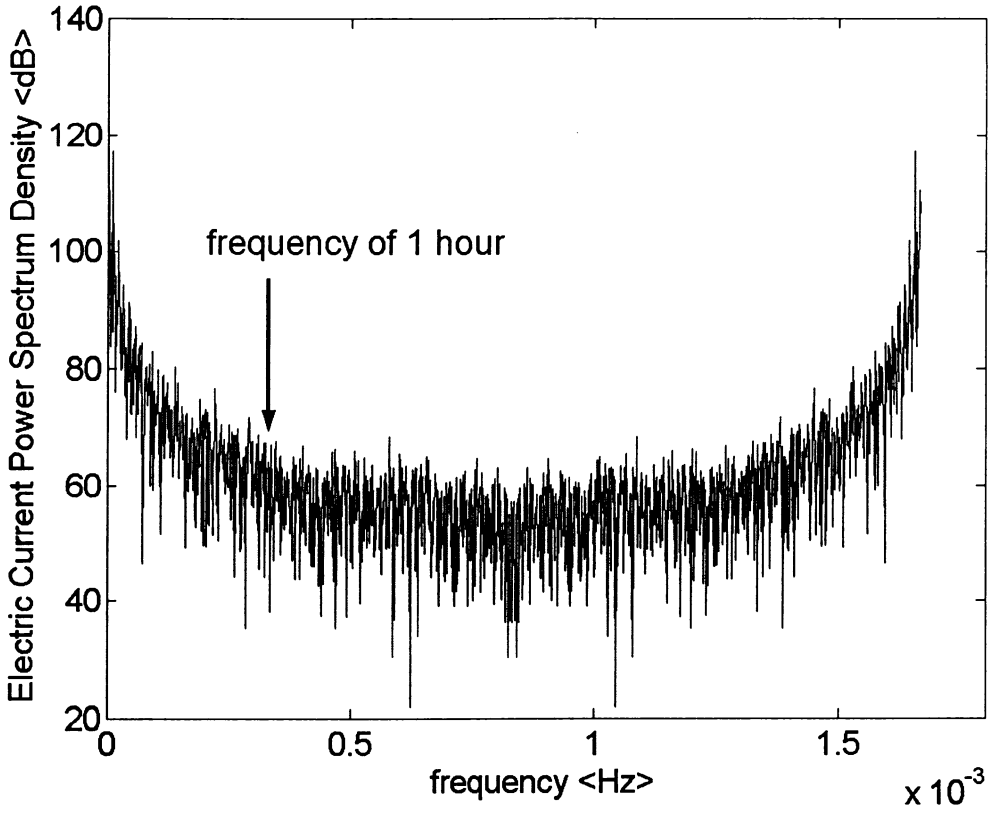


Figure 4-44 FFT of electric current.

Figure 4-44 shows that the power spectral density of the electric current falls by 60 dB within the first 0.3 mHz. This frequency corresponds to a period of about 1 hour. Therefore, a sampling period of less than 0.5 hour is expected to properly capture the thermal effects in the oil system. Since the data were acquired at 10 minute intervals, no aliasing took place.

A 10 minute sampling interval is a problem for sampling the oil pressure. Figure 4-45 shows the pressure waveform over a 24 hours period. Both pump and valve operations can be clearly seen. An expanded view is shown in Figure 4-46. Only 6 samples for each pump cycle is clearly not enough to avoid aliasing, since this rate would hardly be sufficient to capture only the first harmonic of the pressure variation during pump operations.

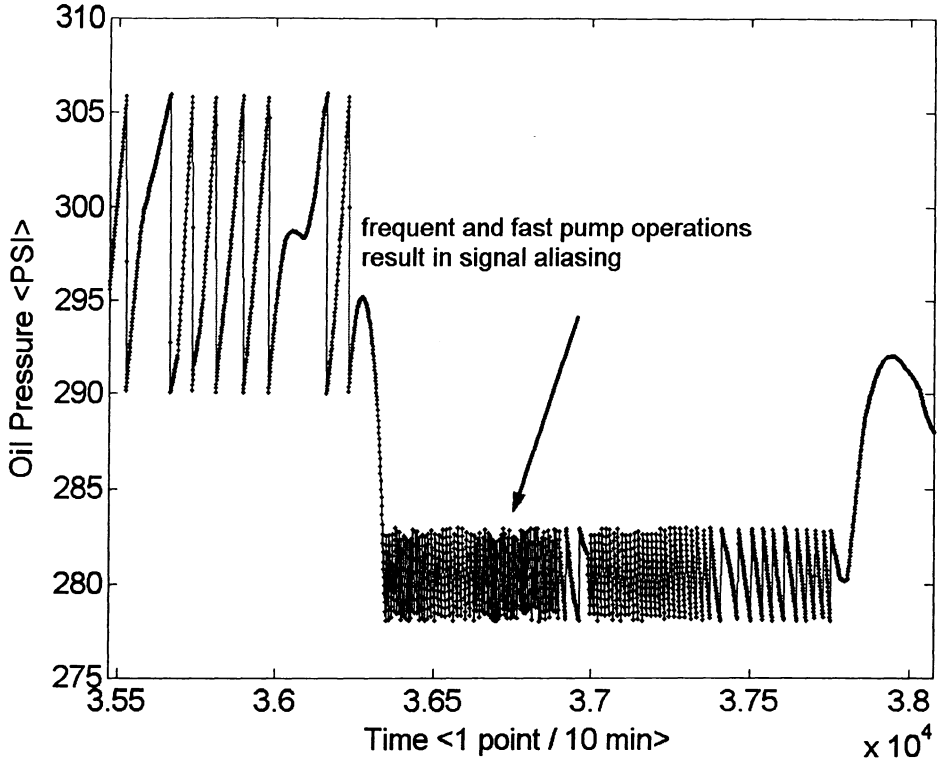


Figure 4-45 Pump operations result in aliasing.

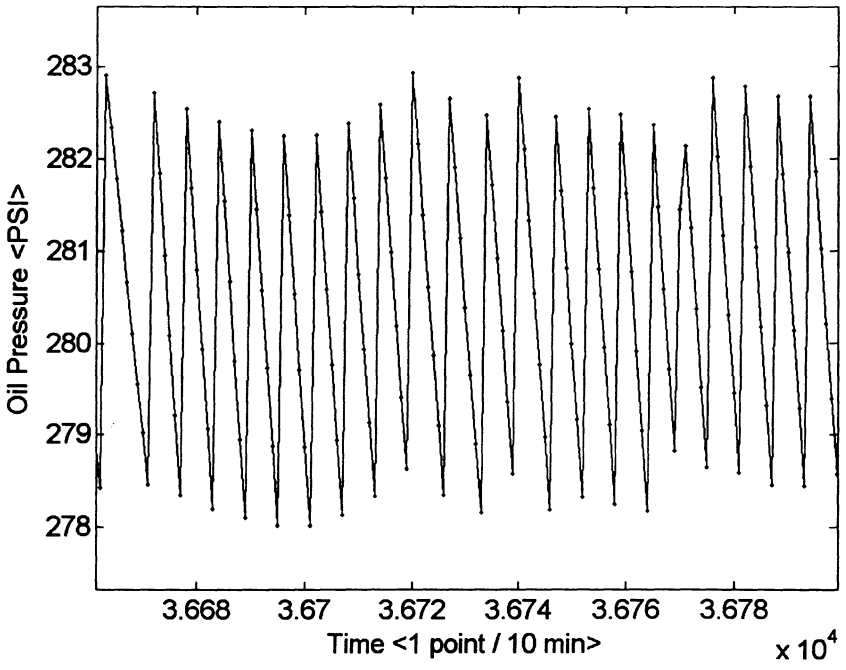


Figure 4-46 Expanded pump operations area.

This fact is one of the main reasons why the available field data are not suitable for directly testing the FDS we have designed. The second reason why the collected field data cannot be used in the complete testing of the leak detector, is the unavailability of leaks. As in the majority of cases, data are obtained from a properly functioning system.

In order to overcome these issues, we have designed a digital system which simulates the thermal processes that take place in the piping system. We have used this simulation to produce data, and have compared the synthetic data with the field measurements. Once we have satisfied ourselves of the good match, we have produced data at a higher sampling rate, under a variety of operating conditions.

4.2.2.2 Data Synthesis

There are two ways of producing synthetic data:

1. Use physical considerations that pertain to heat generation and transfer and produce an analytic model which will mimic the behaviour of the real oil system.
2. Based on the available field data, build a black-box model that empirically explains the input/output relationships, then manipulate the inputs and produce new outputs.

The advantage of using an analytic model for data generation resides in the fact that the model is based on the laws of physics that are expected to govern the modeled processes. These laws are well understood and have been tested over time. The weakness of such a model stems from (a) using laws that simplify the actual processes, and (b) using material parameters whose values may not be completely correct.

The advantage of using an black-box model for data generation results from the fact that the model is based on actual field measurement. No assumptions regarding simplified

physical models or material parameter values are made. There are several weaknesses in such an approach:

1. The empirical model is based on input/output data points and is used for values which are not in the input set. Like any other extrapolator, the predicted values (which have never been seen by the model before) may have little to do with values a real system may experience.
2. The empirical model is based on input/output data points which, probably, represent a relatively small volume from the potential data set. For instance, only Winter data (3 months) may be available for model generation. Therefore, it may be risky to use this model to generate Summer data.
3. The input data may be correlated most of the time. For example the load current and soil temperature are correlated most of the time. Both are driven by a 24 hours cycle. However, this correlation may be affected by weekends and holidays when the load current may be lower than usual. If the model computes the oil pressure and assigns the slightly incorrect coefficients to the load current and soil temperature which are used as inputs, this model may not be appropriate during weekends.

Both modeling approaches have strengths and weaknesses. We have used a physical model in our simulator.

Figure 4-47 shows the measured oil pressure as well as the result of the simulation, over a 48 hour period. The simulation matches well the observed values even exhibiting aliasing during pump operations. To perform the simulation, we have used, as inputs, the field electric current and soil temperature values. Based on this, the simulator computes the oil temperature and oil pressure. The valve and pump threshold values have been extracted from the field data.

The simulation does not perfectly match the measurements. This is due to several factors:

- We have ignored the pressure differences along the cable. The assumption is that a uniform oil temperature along the cable will result in a uniform pressure. In reality, neither the oil temperature, nor the oil pressure are constant with position.
- Our simulation is driven by the soil temperature measurement. This sensor is mounted close to the cable itself. Therefore, rather than measuring the soil temperature (the cold temperature source, sinking heat), this sensor will provide a quantity which is a mixture of soil temperature, oil temperature, and noise.
- We have used a simplified thermal model of the system.

In spite of these shortcomings, we have successfully used the simulator to produce data, which can validate the basic design of the leak detector.

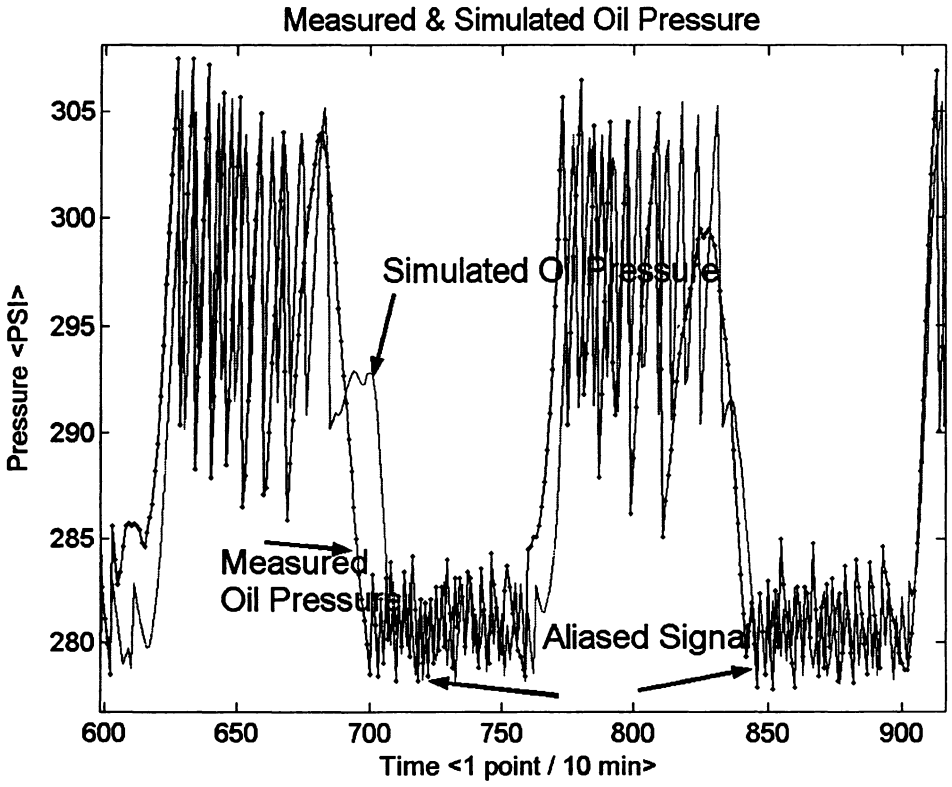


Figure 4-47 Measured and simulated oil pressure.

Using the simulator we have produced data at an equivalent sampling rate of 1 sample per minute. This rate should be able to avoid signal aliasing for all normal operating conditions, with the exception of, perhaps, pressure changes during intensive pump operations during a large leak. We will show that while this phenomenon does occur, the leak detector has a way to mitigate it.

The equivalent of one month of data is produced. The first week is used for learning, while the rest is used for testing. During testing, a total of 5 leaks of varying sizes, are introduced. Figure 4-48 shows pressures from both, the normal, no-leak, system, as well as the system with the leaks.

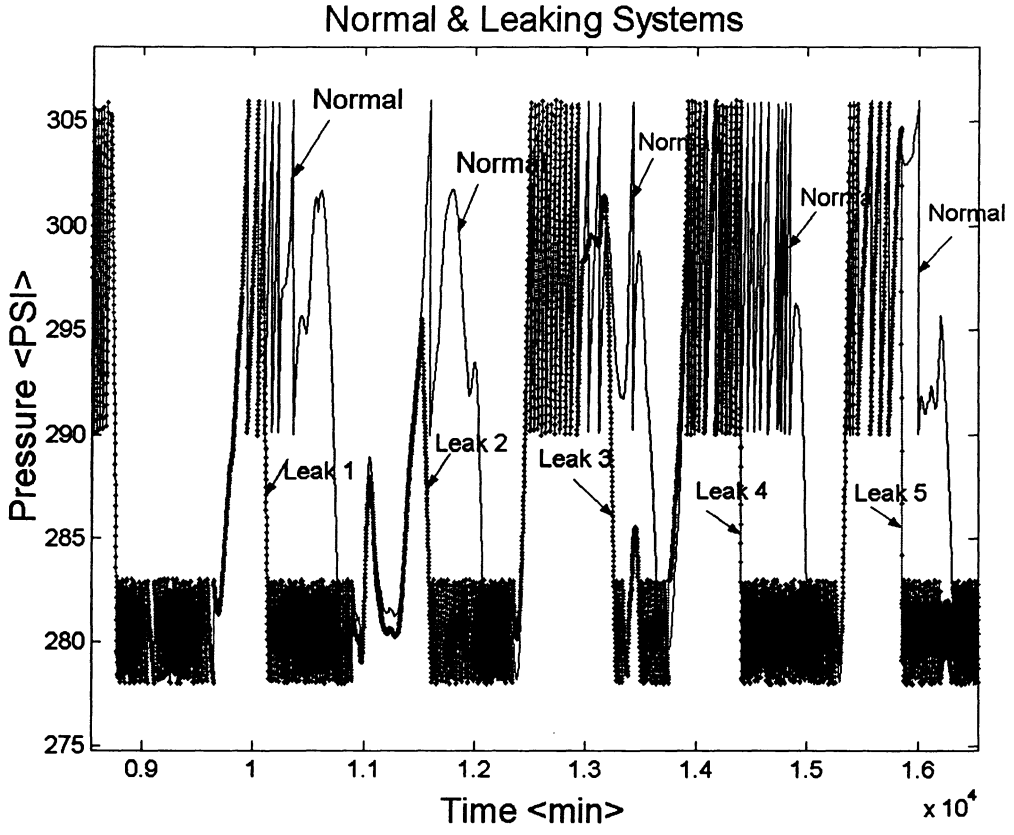


Figure 4-48 Five simulated leaks. Normal & leaking system pressure shown.

All leaks were started at midnight and were kept on for 12 hours. The first leak was programmed to result in a total pressure drop of approximately 400 PSI (or about 80 pump operations), the second leak was ½ the size of the first one, while the third one was ¼ of the size. The fourth leak was double the size of the first leak, while the last one was four times the size of the first leak.

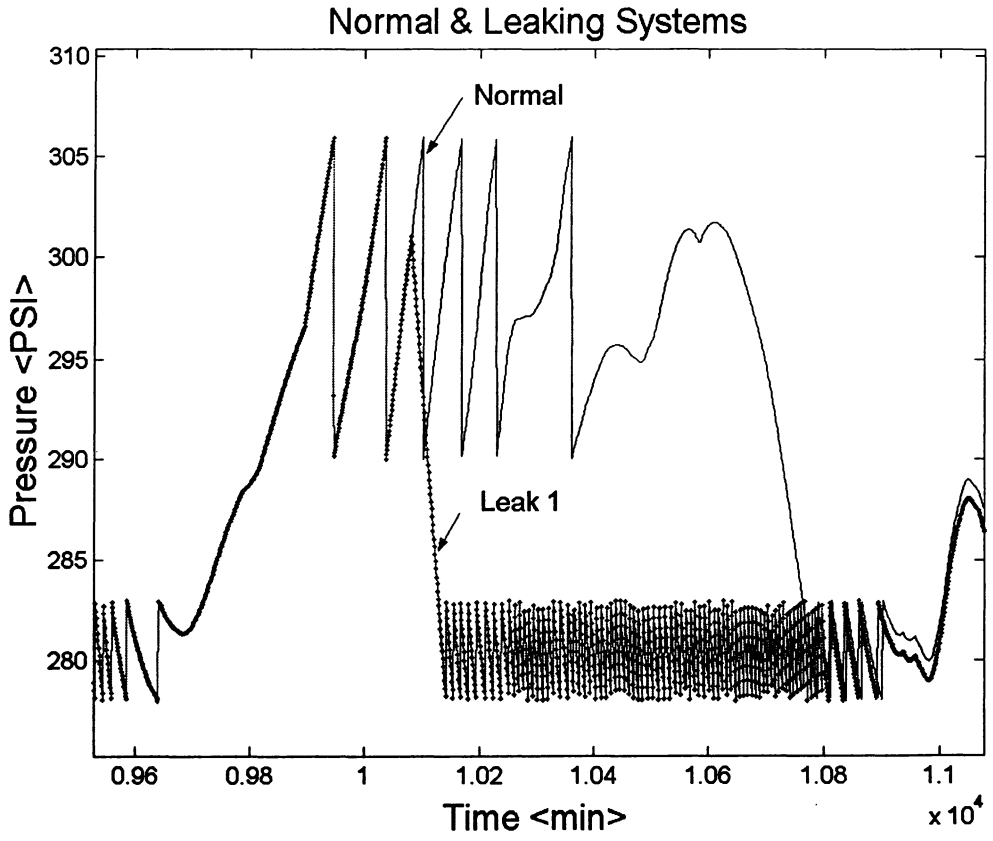


Figure 4-49 First leak waveform expanded.

Figure 4-49 and Figure 4-50 show expanded plots of the pressure during the first and last leaks. As it has been mentioned, during intensive pump operations, during the last leak, signal aliasing takes place This fact can be noticed by the variable pressure turn on and turn off points of the pump. We will discuss the effects of aliasing on the FDS in the next section.

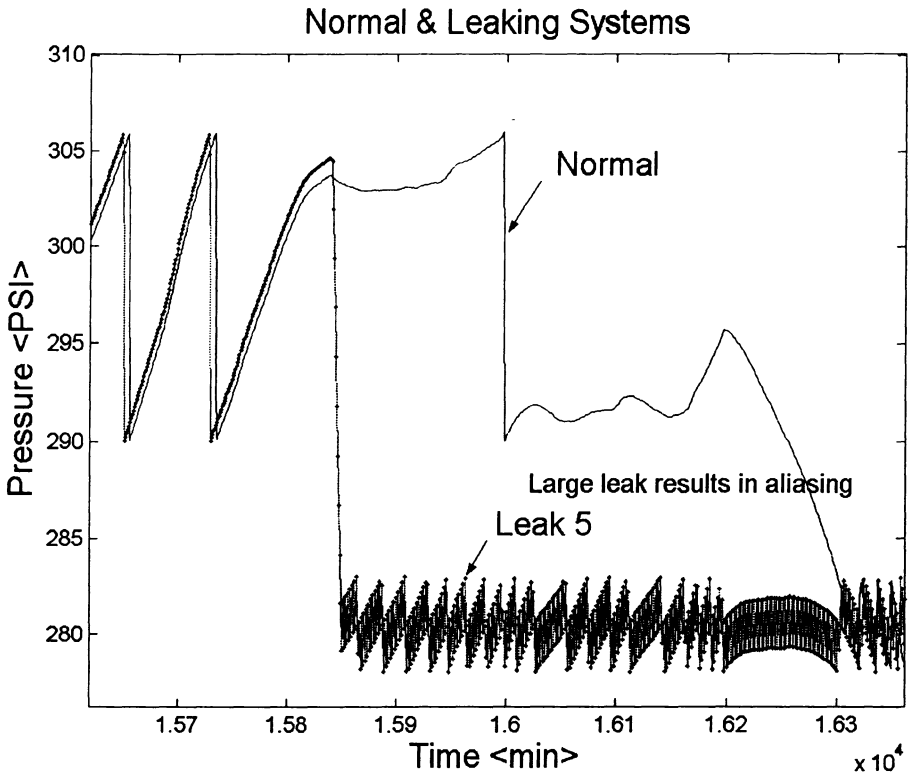


Figure 4-50 Last leak waveform expanded to show signal aliasing.

4.2.3 Artificial Intelligence Based Development

As it was mentioned in the introduction section of this case study, the quantities available to the leak detection system are: oil temperature, soil temperature, electric current carried by the cable, and the oil pressure. The goal is to construct a residual of the oil pressure, which can be used by an alarm block determining the existence of a leak. We will first discuss the details of a model that computes this residual based on the oil pipe temperature only. An alternative model based on electric current and soil temperature will then be presented. Next, we will show how an oil temperature estimate can be computed based on the electric current and soil temperature used as inputs. Such a model will be integrated in a general leak detection system which will be presented in the last section of the case study. This system will integrate all the leak detection approaches described. Finally, there will be a discussion of the strengths and weaknesses of the technology.

4.2.3.1 Oil Temperature Based Implementation

During normal system conditions, when there are no oil leaks present, the oil pressure changes are caused by three sources:

1. oil temperature changes
2. pump operations
3. valve operations

Therefore, in order to build a model capable of estimating the oil pressure, the above processes will have to be considered. Figure 4-51 shows the block diagram of the oil pressure estimator.

Four major activities take place in this model: pump modeling, valve modeling, thermal modeling, and finally, a merger of the three processes. As in any modeling endeavor, each block can be implemented using either a black box or a physical approach. For the pump and valve models, a physical implementation will be based on the operation details of the devices (model types), as well as some measure of their maintenance status (on/off threshold adjustments, physical device condition, etc). Since this information is not available many times, and when it is available, it may be unreliable, such an approach is considered unfeasible.

Both the pump and valves could be modeled using a black box approach. This method would be able to determine the average turn on and turn off points. However, important effects, such as temporary prolonged operations caused by imperfect equipment, as well as the effect of temperature variations during the valve or pump operations, would be very difficult to capture.

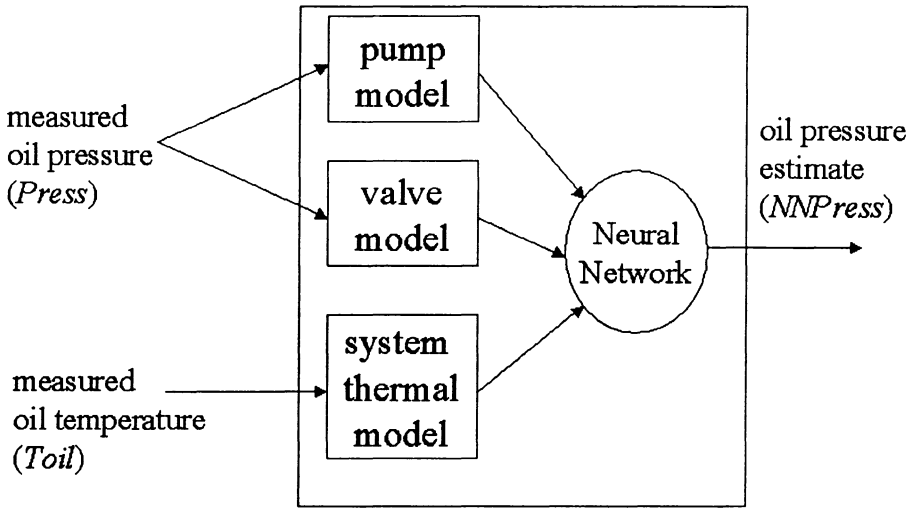


Figure 4-51 Oil pressure model using pump/valve models.

To overcome these issues, we use a different approach. Rather than trying to model the pump and valve, we are building a system which computes an oil pressure which would be experienced if the valve and pump did not exist. We call this quantity "a corrected oil pressure" and is strictly a theoretical entity.

The corrected oil pressure can be negative, if the physical pressure drops sufficiently, and can have very large positive values, values which could not be sustained by a physical pipe. The advantage of using a corrected oil pressure is that its variation depends exclusively on the thermal properties of the system.

The pump and valve operations are considered disturbances and are eliminated. Figure 4-52 and Figure 4-53 show the operation of the disturbance canceller.

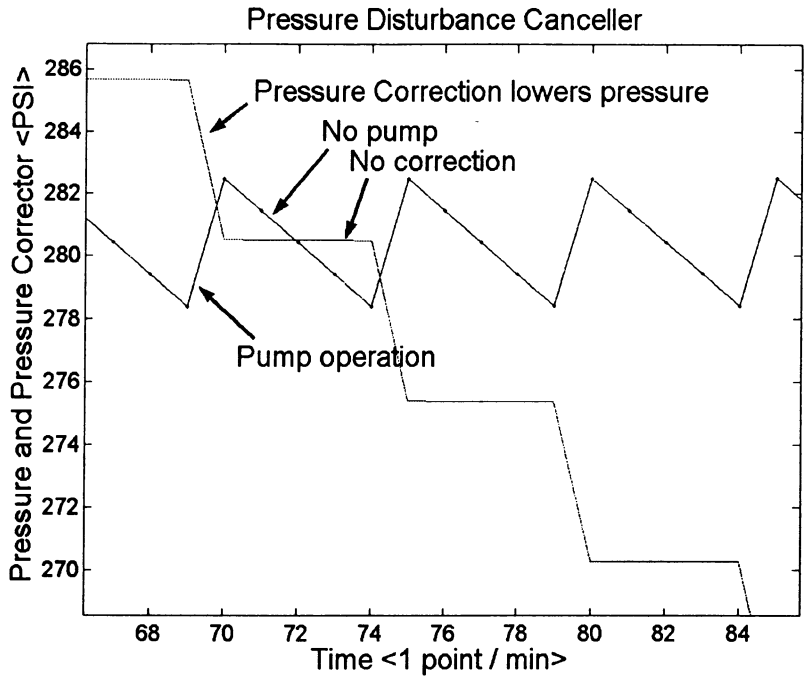


Figure 4-52 Pump pressure increase cancelled.

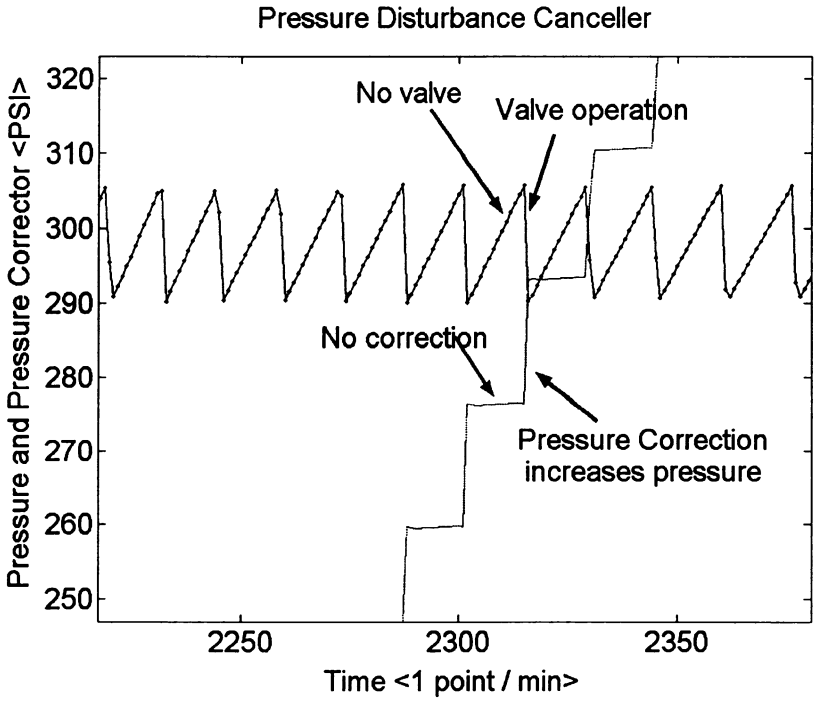


Figure 4-53 Valve pressure decrease cancelled.

Four pump operations can be seen in Figure 4-52. Their effect is to increase the oil pressure. The disturbance canceller computes the pressure correction that must be added (subtracted) to the actual measured pressure to cancel the pump effect. Since the effect of the pump on the oil pressure depends on the integral of the pump flow, the canceller will also integrate its output. The effect is a signal which becomes more and more negative with every pump operation (in Figure 4-52, the canceller waveform is shifted on the graph to overlap the pressure), and is constant during pump in-activities.

One important comment to make at this point, is that the canceller does not attempt to model the pump or valve, it just attempts to cancel the effect of each particular operation. This simplifies the leak detector's design and increases its pressure modeling accuracy. We will discuss the disadvantages of this approach at the end of the case study, after detection results are presented.

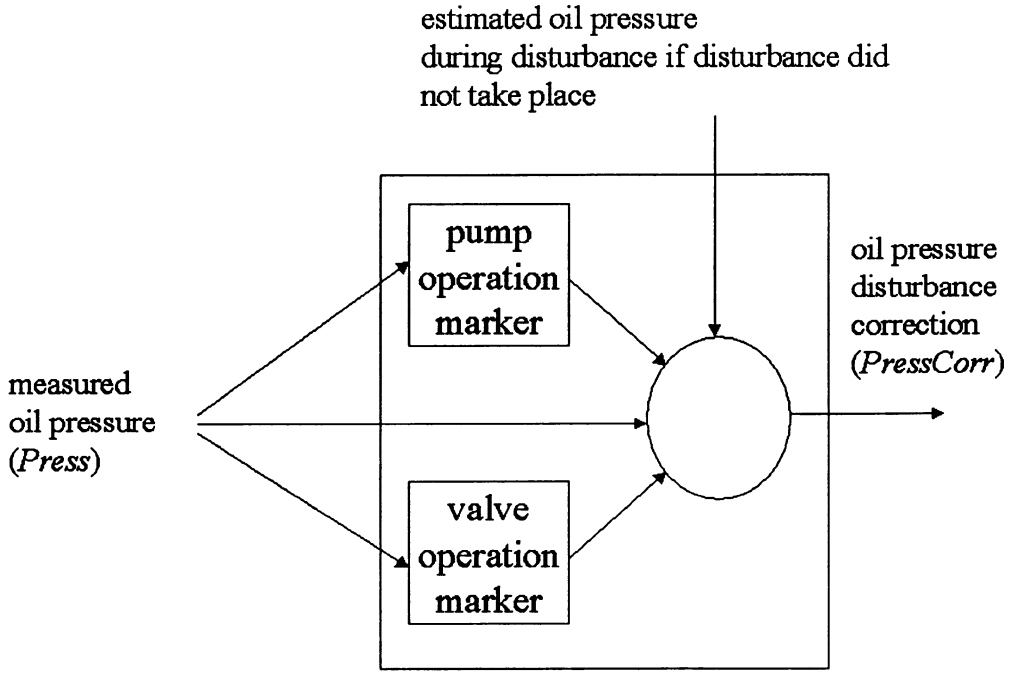


Figure 4-54 Pressure disturbance (pump/valve) canceller.

Figure 4-54 shows the inner workings of the canceller. The valve and pump operation markers indicate the location of the disturbance. Using an estimated oil pressure which captures the thermal effects during the disturbance, a pressure correction is computed. Once this correction is added to the oil pressure, the disturbance is cancelled and all pressure variations in the resulting waveform are caused by temperature effects. One important point to make about the canceller is that it uses, as an input, a quantity which cannot be obtained by measurement: the estimated oil pressure during the disturbance, if the disturbance did not take place. While this quantity cannot be measured, it can be computed by a thermal model which estimates the oil pressure in the absence of valve and pump disturbances. However, to obtain such a model, one has to have the disturbance corrected pressure in order to use it for training. An iterative approach emerges and is shown in Figure 4-55.

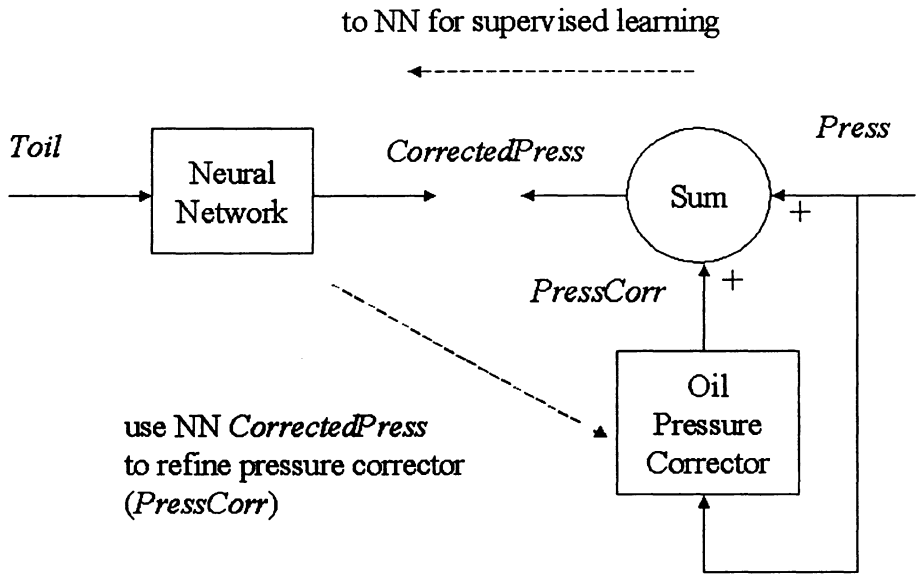


Figure 4-55 Oil pressure model based on oil temperature.

In order for the disturbance corrector to start performing its task, it needs a way to estimate *PressCorr* without the use of the corrected pressure (since this signal is not available yet). As an approximation, it assumes that the oil temperature is constant during the relatively short disturbances. In this case the oil pressure variation is caused exclusively by the disturbance and is corrected by:

$$\text{PressCorr}_i = \sum_{k=0}^{i} \left(\text{Press}_{2k} - \text{Press}_{1k} \right) \qquad (4\text{-}33)$$

where:

PressCorr$_i$ is the pressure correction after disturbance "*i*"

Press$_{2k}$ is the pressure value at the end of disturbance "*k*"

Press$_{1k}$ is the pressure value at the begining of disturbance "*k*"

Applying *PressCorr* from equation (4-33) will result in a corrected oil pressure which will be constant during disturbances (valve/pump operations). This is reasonable, since (4-33) is based on the assumption that the oil temperature is constant during disturbances, therefore the pressure should not change once the disturbance is eliminated. Having an initial estimate of the corrected pressure (*CorrectedPress* in Figure 4-55), the Back Propagation Neural Network responsible for modeling the thermal effects is trained. Once the training is completed, the Neural Network can provide an estimate of the oil temperature variation during all times, including during disturbances. The disturbance canceller uses this estimate and produces a new correction, *PressCorr*. The behaviour of the two blocks, the Neural Network and the disturbance canceller, is equivalent to a student learning the approximate solution from a teacher who does not have the exact answer. Once the student completes the learning, it can help the teacher improve its accuracy.

Figure 4-56 shows the disturbance corrections for the one week learning set. Two points can be made examining the plots:

1. the algorithm converges in about 5 iterations. The pressure correction curves do not change significantly if one increases the number of iterations.
2. the average of correction signal should be bounded. If, over a large period of time, the pressure correction keeps decreasing, the canceller indicates that a net pressure increase disturbance affects the system. This means that pump

operations increase the pressure more than valve operations. An oil leak is present. If, on the other hand, the pressure correction has a trend upward, this is an indication of data acquisition failure, or of oil spilling back into the system through the valve. The assumption this analysis is based on, is that in the long term there is no drift in the average value of the oil temperature. If, for instance, the load current supplied by the cable has a slight increase over the long term, the amount leaked by the valve will be larger than the amount pumped in, and a net downward pressure correction will be experienced.

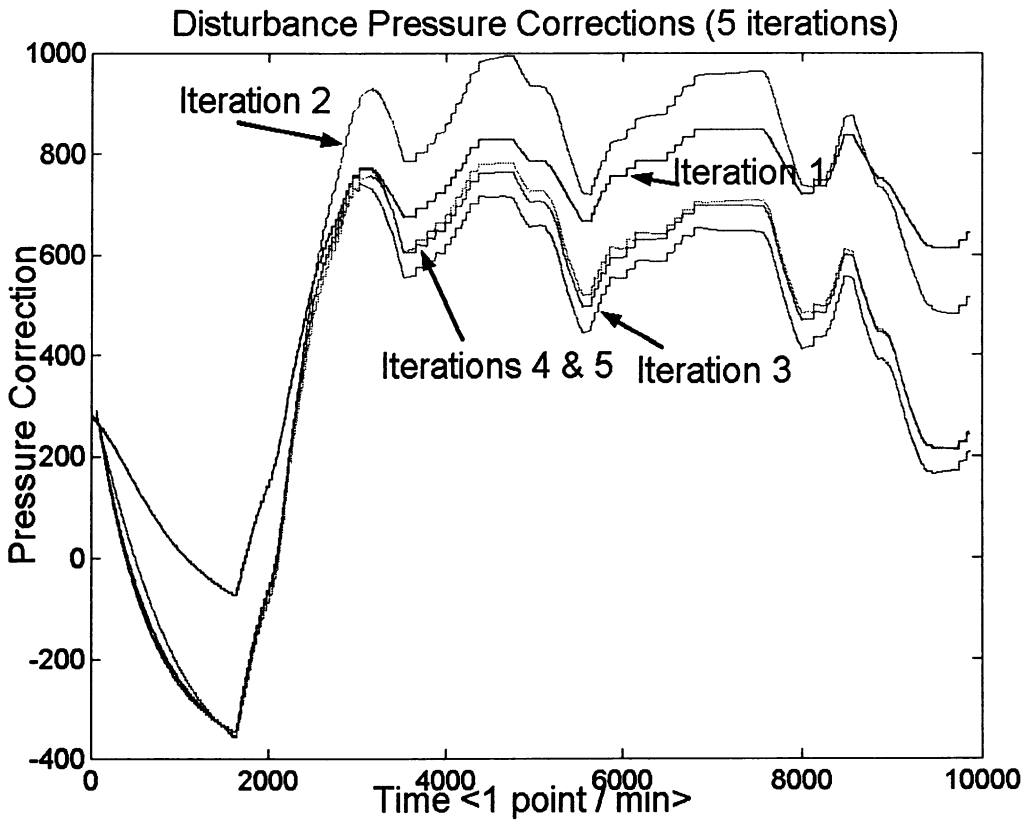


Figure 4-56 After 5 iterations, pressure corrections show convergence.

Before we give a formal proof of algorithm convergence, there is one more comment that must be made regarding the learning data presented to the Neural Network.

The network must learn the dependence between oil temperature and the corrected oil pressure. As has been mentioned, the corrected pressure is an integral quantity. This

means that errors present at the beginning of the set will affect the whole set in a proportion larger than errors present at the end of the set. The Neural Network performs during learning an error minimization and treats all errors equally. This means that the network will try to reduce the errors that occur at the beginning of the data set more than the ones at the end.

In order to mitigate the integral effect, we will present the network with the derivative of input and output sets during learning. Based on the assumption that the input / output relationship is mostly linear (which is a reasonable assumption, since the pressure variation between the pump operations and valve operations is very small), the Neural Network trained on the differential data, will recall using directly the un-differentiated data.

Now, we will provide the formal proof of algorithm convergence.

Let f be the thermal system transfer function:

$$CorrectedPress = f(Toil) \tag{4-34}$$

Expanding in a Taylor series around $Toil_0$:

$$\text{CorrectedPress} = f(Toil)\Big|_{Toil_0} + \frac{df}{dToil}\Big|_{Toil_0} (Toil - Toil_0) + \ldots \tag{4-35}$$

Taking the time derivative of (4-35):

$$\text{CorrectedPress}' = \frac{df}{dToil}\Big|_{Toil_0} Toil' + \ldots \tag{4-36}$$

Keeping only the linear term in *Toil* results in:

$$\text{CorrectedPress}' = K * Toil'$$

where:

(4-37)

$$K = \left.\frac{df}{dToil}\right|_{Toil_0}$$

The Neural Network will determine K such as the total error term is minimized:

$$\varepsilon = \sum_{\text{i=all samples}} \left(\text{CorrectedPress}'_i - K * Toil'_i\right)^2 \tag{4-38}$$

By setting the derivative of ε with respect to K to zero, results in:

$$\sum_{\text{i=all samples}} \left[\left(\text{CorrectedPress}'_{R,i} - K_R * Toil'_i\right) Toil'_i\right] = 0 \tag{4-39}$$

The subscript "R" attached to the *CorrectedPress* and K terms denotes the fact that these quantities are computed after "R" iterations. From (4-39), the term K_R becomes:

$$K_R = \frac{\sum_{\text{i=all samples}} \left(\text{CorrectedPress}'_{R,i} Toil'_i\right)}{\sum_{\text{i=all samples}} Toil'^2_i} \tag{4-40}$$

The numerator of (4-40) can be split between the samples that are not affected by disturbances and the ones that are:

$$K_R = \frac{\sum_{\text{i=distur.}} \left(\text{CorrectedPress}'_{R,i} Toil'_i\right)}{\sum_{\text{i=all samples}} Toil'^2_i} + \frac{\sum_{\text{i≠distur.}} \left(\text{CorrectedPress}'_{R,i} Toil'_i\right)}{\sum_{\text{i=all samples}} Toil'^2_i} \tag{4-41}$$

For the samples not affected by disturbances, the pressure correction is constant, and its derivative is zero. Therefore for these samples:

$$\text{CorrectedPress}'_{R,i} = \text{Press}'_i \qquad \text{for all i} \neq \text{disturbance} \tag{4-42}$$

Combining (4-37) and (4-42) and considering the fact that a new corrected pressure is computed based on the Neural Network gain computed at the previous iteration, results in:

$$\begin{aligned} \text{CorrectedPress}'_R &= K_{R-1} * Toil' \\ \text{Press}' &= K * Toil' \end{aligned} \tag{4-43}$$

Substituting (4-43) into (4-41):

$$K_R = K_{R-1} \frac{\sum_{\text{i=distur.}} Toil'^2_i}{\sum_{\text{i=all samples}} Toil'^2_i} + K \frac{\sum_{\text{i}\neq\text{distur.}} Toil'^2_i}{\sum_{\text{i=all samples}} Toil'^2_i} \tag{4-44}$$

Defining:

$$\lambda = \frac{\sum_{\text{i=distur.}} Toil'^2_i}{\sum_{\text{i=all samples}} Toil'^2_i} \tag{4-45}$$

and considering the fact:

$$\sum_{\text{i=all samples}} Toil'^2_i = \sum_{\text{i=distur.}} Toil'^2_i + \sum_{\text{i}\neq\text{distur.}} Toil'^2_i \tag{4-46}$$

Equation (4-44) becomes:

$$K_R = \lambda K_{R-1} + (1-\lambda) K \tag{4-47}$$

Since before iteration 1, the corrected pressure during the disturbance was constant, since the temperature was considered constant, as a first approximation, from (4-43) it follows that K_0 is zero. Therefore, from (4-47) we can develop the following sequence:

$K_1 = K\,(1-\lambda)$

$K_2 = K_1\,\lambda + K\,(1-\lambda) = (\lambda+1)\,K\,(1-\lambda)$

$K_3 = (\lambda^2+\lambda+1)\,K\,(1-\lambda)$

.

.

.

$$K_{R+1} = (\lambda^R+\lambda^{R-1}+\lambda^{R-2}..+\lambda+1)\,K\,(1-\lambda) \tag{4-48}$$

Equation (4-48) contains a geometric series whose value is readily available. Therefore (4-48) becomes:

$$K_{R+1} = \frac{(1-\lambda^{R+1})}{(1-\lambda)} K (1-\lambda) \tag{4-49}$$

and :

$$K_{R+1} = K(1-\lambda^{R+1}) \tag{4-50}$$

From equation (4-45):

$$\lambda = \frac{\sum_{\text{i=distur.}} Toil'^{\,2}_i}{\sum_{\text{i=all samples}} Toil'^{\,2}_i} \leq 1 \tag{4-51}$$

From inequality (4-51) and equation (4-50) we can draw the following conclusions:

1. Increasing the number of iterations, the value of K_{R+1} will always converge to the theoretical value. Therefore, the method is convergent.
2. A smaller λ results in faster convergence. In other words, if the percentage of data affected by disturbances is lower, the convergence of the algorithm increases. Slow pumps and valves will result in a longer disturbance and, therefore, a slower convergence.

Figure 4-57 shows the corrected Neural Network pressure overlapping the measured corrected oil pressure. As can be seen, after approximately 10,000 minutes, the two curves diverge.

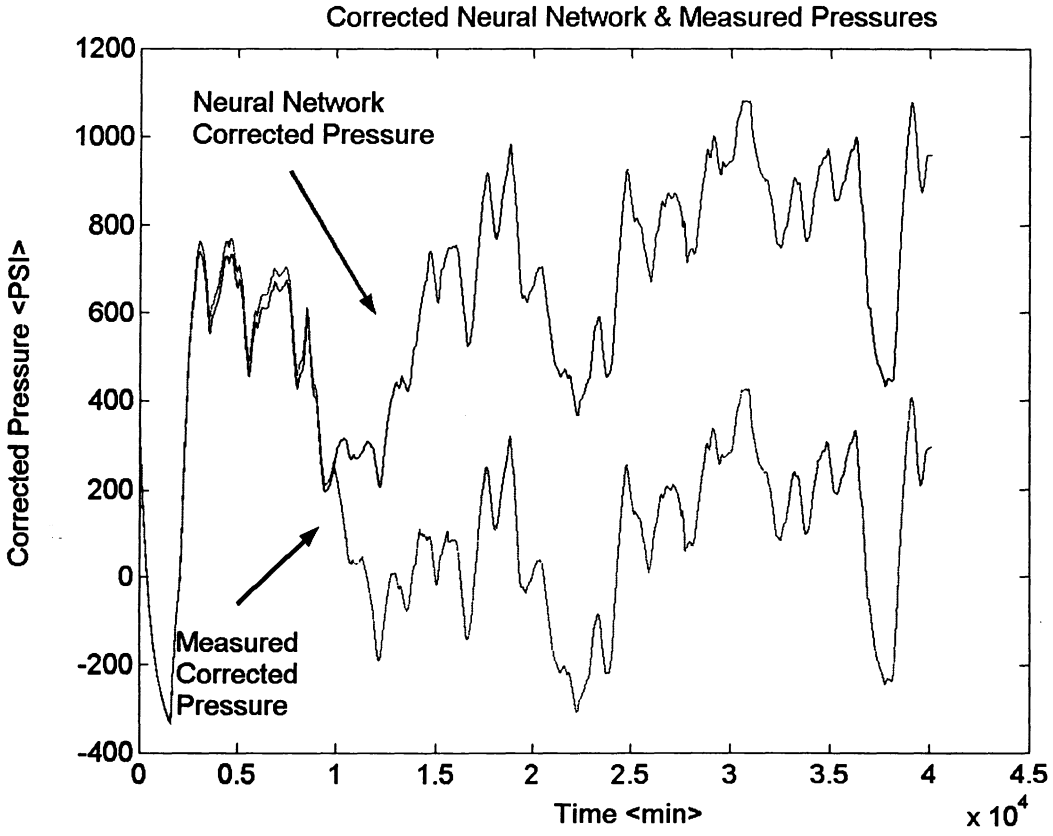


Figure 4-57 Corrected Neural Network and measured oil pressures.

Figure 4-58 shows the residual value calculated as the difference between the Neural Network corrected pressure (modeled value), and the measured oil corrected pressure.

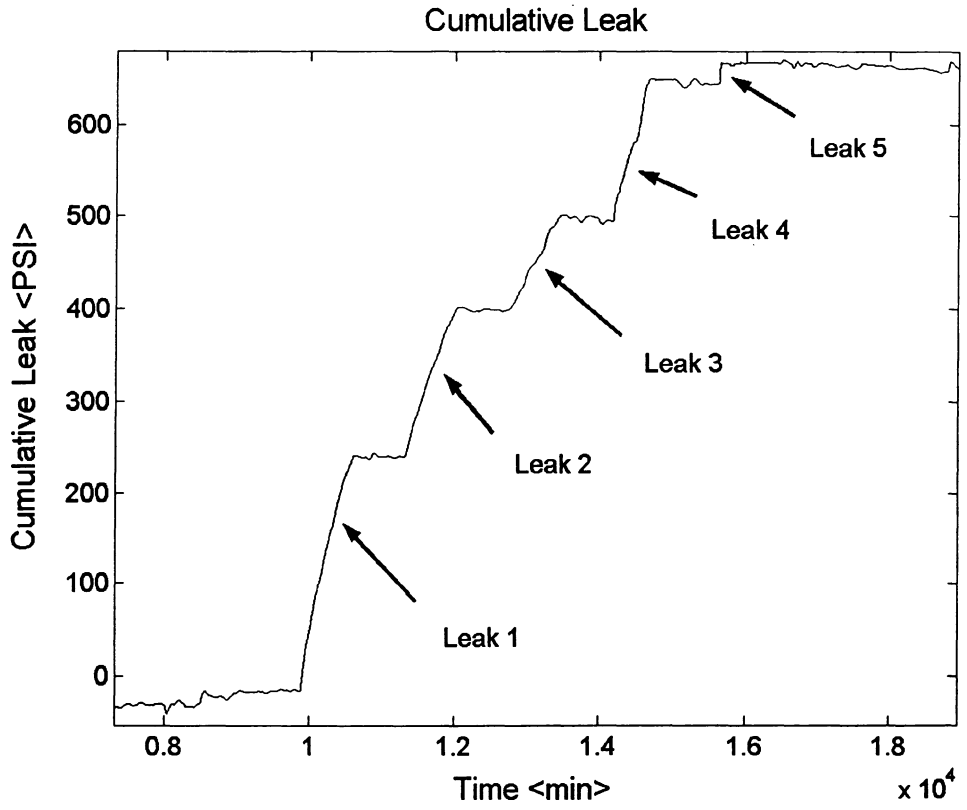


Figure 4-58 Leak to-date clearly shows 5 leaks. Last 2 leaks aliased.

The increases in the residual values, make the five leaks clearly visible. As was mentioned in the Data Synthesis section, the 5 leaks are in the proportion: 1, ½, ¼, 2, and 4. Each leak had a duration of 12 hours, or 720 minutes. These leak sizes and durations are in fairly good agreement with the residual increases in Figure 4-58.

Problems can be noticed with the last two leaks. Both leaks have residuals which are lower than the residual of Leak 1, in spite of the fact that they are 2 and 4 times larger than Leak 1. Surprisingly, Leak 5, the largest one, has the smallest residual. This problem is intrinsic to the leak detection method we have used. During a disturbance, the algorithm corrects the pressure in order to eliminate the disturbance, and no attempt to account for it is made. Therefore, the leaked oil during pump operations is not accounted for. Of course, extremely large leaks will result in many disturbances which mask the

leak itself. In the limit, if the leak caused the pump to operate continuously, this system would not detect it. On the other hand, if a leak is small, the system will reliably detect it, in spite of the fact that very few extra pump operations were needed to replenish the oil. This is the downside of the methodology mentioned at the beginning of this section. If the method modeled the pump and valve, it could flag a leak even during the disturbance. Of course, this assumes that the disturbance model is accurate enough (ie. it is possible, and practical, to accurately model the pump and valve).

To be able to handle very large leaks, another approach was used. A second leak detector works in parallel with the *Toil/Press* detector. This new detector learns a relationship that ties oil temperature variations to the number of pump and valve operations. Such a system is not very accurate if small leaks are experienced, since only a very small number of extra operations will take place. However, if a large leak takes place this approach will detect it. We will not describe this FDS in this work, since it would not bring any new elements to the discussion.

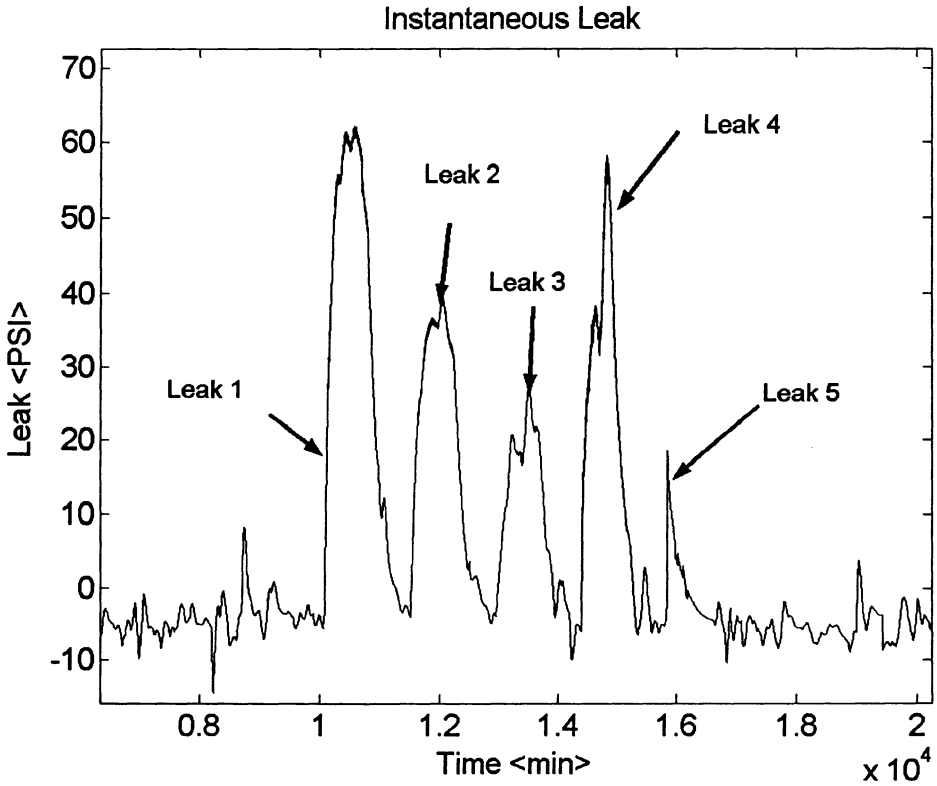


Figure 4-59 Instantaneous leak derived by *Toil* to *Press* model.

Figure 4-59 shows the differential of the residual signal. This quantity simulates the actual instantaneous leak value. The differential was computed over a 4 hour interval. If the real differential were to be displayed at the sampling rate of 1 minute, the signal to noise level would be very low. This is to be expected, since the amount of oil leaked in 1 minute is much smaller compared to the expected modeling and measurement noise levels.

4.2.3.2 Electric Current / Soil Temperature Based Implementation

The previous section has introduced the oil temperature based leak detector. Building on this experience, we now develop a detector based on the electric current and soil temperature.

While the previous detector has good performance, it suffers of one weakness: it depends on an oil temperature measurement. As it was mentioned in the previous chapters, the oil temperature is not available. The sensor responsible for the oil temperature measurement is glued on the outside of the pipe. While its measurement is correlated to the desired quantity, its output is determined, to a large extend, by the soil temperature. This is the reason why an electric current based approach is desirable.

The oil temperature is determined by two factors: the electric current and the soil temperature. The electric current is responsible for the high frequency temperature changes, while the soil temperature determines the slow frequency behaviour. The soil temperature / oil pressure relationship is expected to be quasi linear. However, the electric current to oil pressure dependence is much more complicated.

We have seen in the previous section that the oil temperature / oil pressure relation is almost linear. The oil temperature variation depends on the heat energy transferred to or from the cable. The electric current, due to the cable ohmic resistance, produces heat which is transferred to the oil. At the pipe interface heat is lost into the soil, since the soil

temperature is always lower than the pipe temperature. The produced heat depends on the integral of the square of the electric current. Of course, the actual dependences are more complicated than these simplified relationships. It is useful to start with a simple expected functional relationship in order to provide the parametric model enough input variables for proper estimation.

In chapter 2 we have introduced the ARMA model.

$$y_k = \sum_{i=0}^{N} a_i x_{k-i} + \sum_{i=1}^{M} b_i y_{k-i} \qquad (4\text{-}52)$$

The first sum of (4-52) represents the Moving Average component, is composed of a weighted average of input values. The second sum is a weighted average of past model outputs and is called the Autoregressive part.

In our application, (4-52) becomes:

$$Corrected\,\mathrm{Pr}ess_k = \sum_{i=0}^{N} a_i I^2{}_{k-i} + \sum_{i=0}^{P} c_i Tsoil_{k-i} + \sum_{i=1}^{M} b_i Corrected\,\mathrm{Pr}ess_{k-i} \qquad (4\text{-}53)$$

For the leak detector case, the second and third sums in (4-53) contain only one term. Adding more terms would imply that the corrected oil pressure depends not only on the soil temperature, but also on its the rate of change. The soil temperature changes very slowly, therefore its derivative is essentially zero.

Figure 4-60 shows the leak detector based on equation (4-53). The pressure disturbance corrector is identical to the one described in the previous section and its description will not be repeated. The Neural Network uses an autoregressive input delayed by a delay L.

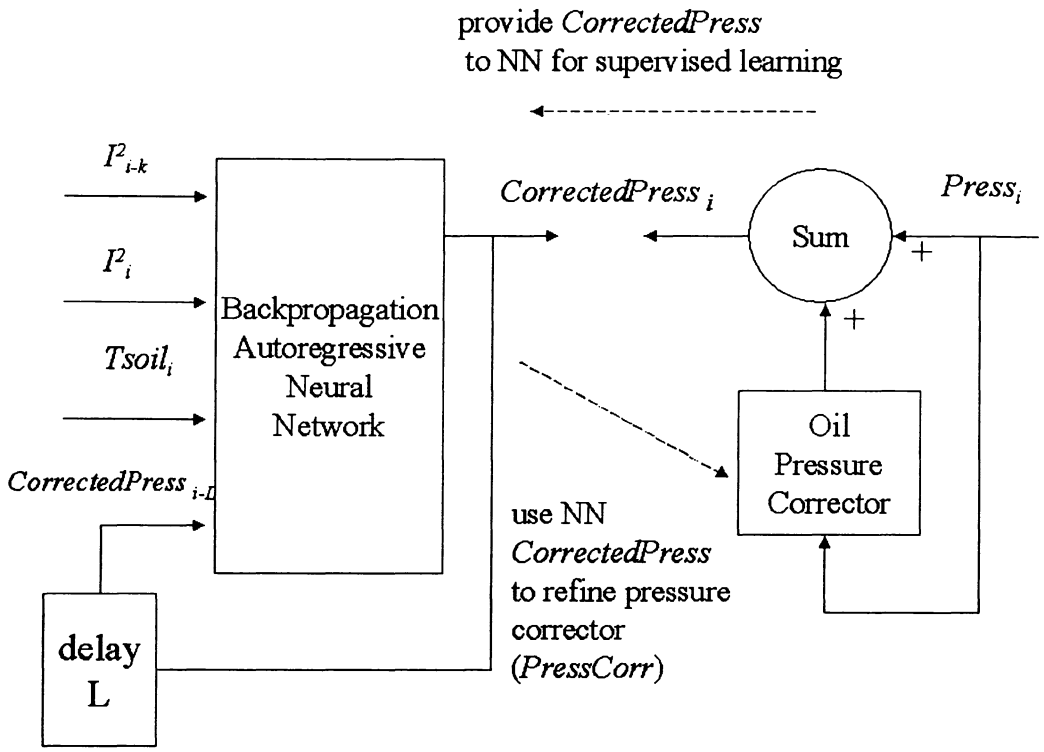


Figure 4-60 Oil pressure model based on autoregressive oil pressure, electric current, and soil temperature.

As discussed by Ljung in [25], the variance of an estimated parameter based on a given number of data, depends on the average information per sample. There is a trade-off between the noise reduction that slow sampling may offer, and the poor information about the dynamics that slowly sampled data contain.

One of the parameters to be estimated by the Neural Network is the weight of the *CorrectedPress* regressive input. The value of this weight corresponds to the b_1 coefficient in equation (4-53), where L is equal to 1. Physically, one expects the values of the $CorrectedPress_i$ and $CorrectedPress_{i-1}$ to be very similar if the sampling rate is high. After all, if the sampling interval is small, the oil pressure does not have time to change too much. Therefore, one expects b_1 to have a theoretical value close to 1. The network, seeing the high correlation between $CorrectedPress_i$ and $CorrectedPress_{i-1}$, will decide to compute the new pressure value almost exclusively based on the previous value. This will have two effects:

1. The network will largely ignore the current inputs basing its forecast on the autoregressive input
2. The autoregressive component being close to 1, will result in an unstable system. When the component is estimated, an estimation error is expected. If the theoretical value is close to 1, the estimation error may result in a component whose value is larger than 1.

To solve this problem, Ljung [25] suggests choosing the sampling interval of the same order as the physical system time constant. In our case, this is not possible since we must sample much faster to prevent the pump operations aliasing discussed in the previous chapter. A possible solution would be to use a high sampling rate when performing disturbance cancellation, but a low rate when estimating the Neural Network autoregressive model.

We use a different approach. We use the 1 sample per minute needed by our application, however, the autoregressive element is delayed by the amount L, where L is much larger than 1. In other words, we do use an autoregressive input, but choose this input in such a way as to minimize the correlation between its value and the output. Minimizing this correlation, forces the Neural Network to obtain the information it needs form the other inputs, the electric current history. This is a positive development, since the electric current is the driver behind the pressure variations, while the autoregressive pressure input only acts as the integration component of the model.

To minimize the correlation between the output and the autoregressive input, it was empirically determined that L should be equal to 6 hours. This is rather intuitive since we expect a correlation close to 1 for L close to 24 hours (since the current consumption has a periodicity of 24 hours), and expect a correlation of close to –1 for L close to 12 hours(because of the day / night consumption pattern).

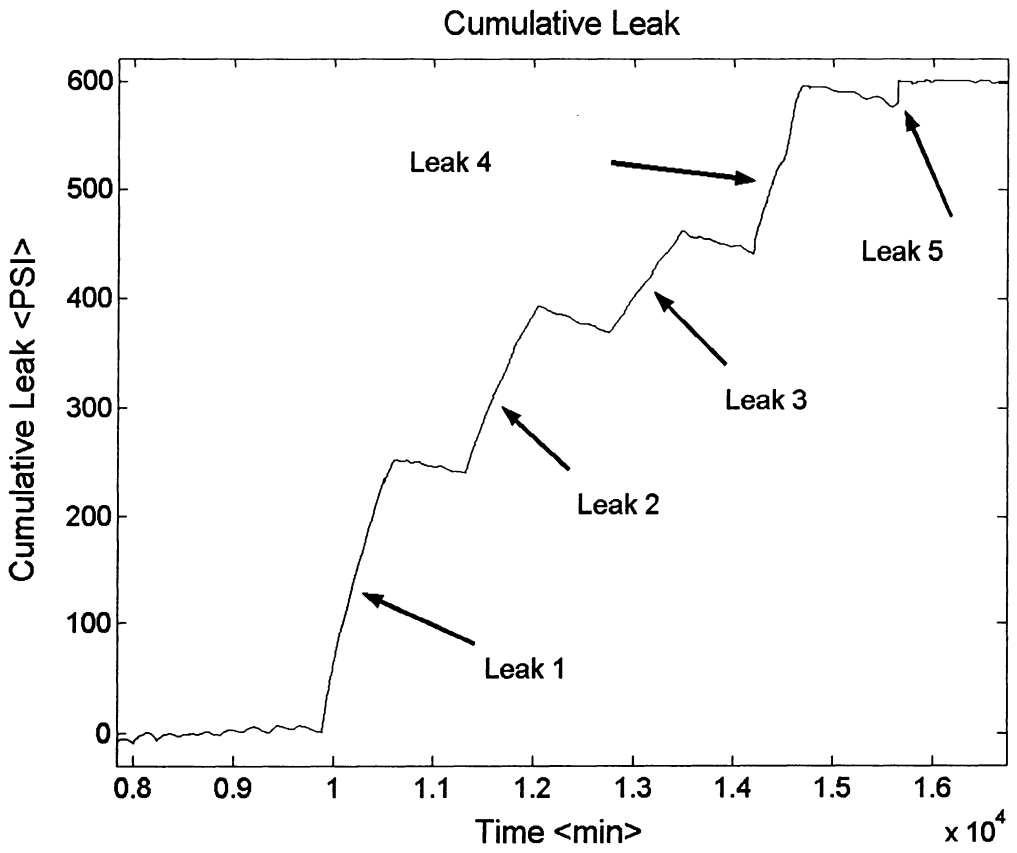


Figure 4-61 Cumulative leak computed with the *I2* to *Press* model.

Figure 4-61 shows the residual produced by the current to pressure leak detector. As in the previous case, the 5 leaks can be clearly seen. The quality of the residual seems to be not as good as the previous model. The residual values after each leak should have a constant value. A "negative" leak is experienced after the first four leaks. There are several possible explanations for this performance.

Just like the previous model, one week of data was used for training. However, the current to pressure detector has many more parameters to determine compared to the oil temperature to pressure model. This is because of the large number of current inputs used. It is possible that a larger learning data set would improve the Neural Network performance. It is also possible that the use of Principal Component Analysis, or even better, Partial Least Squares, would compress and de-correlate the Neural Network input. These aspects will be investigated in further research. Finally, the autoregressive

implementation is more sensitive to the iterative aspect of our disturbance elimination methodology.

In the previous model, the Neural Network had to learn an essentially moving target. In the current autoregressive model, the Neural Network has to learn the same moving target, while one of its inputs, the autoregressive one, also changes between iterations. We expect the estimated pressure to converge more slowly. Therefore, perhaps increasing the maximum iteration limit could further improve the Neural Network performance.

Finally, the residual differential is shown in Figure 4-62. Like in the previous model, the differential is computed over a 4 hour interval. The "instantaneous" leak value plot is consistent to the one in Figure 4-59 for the previous model.

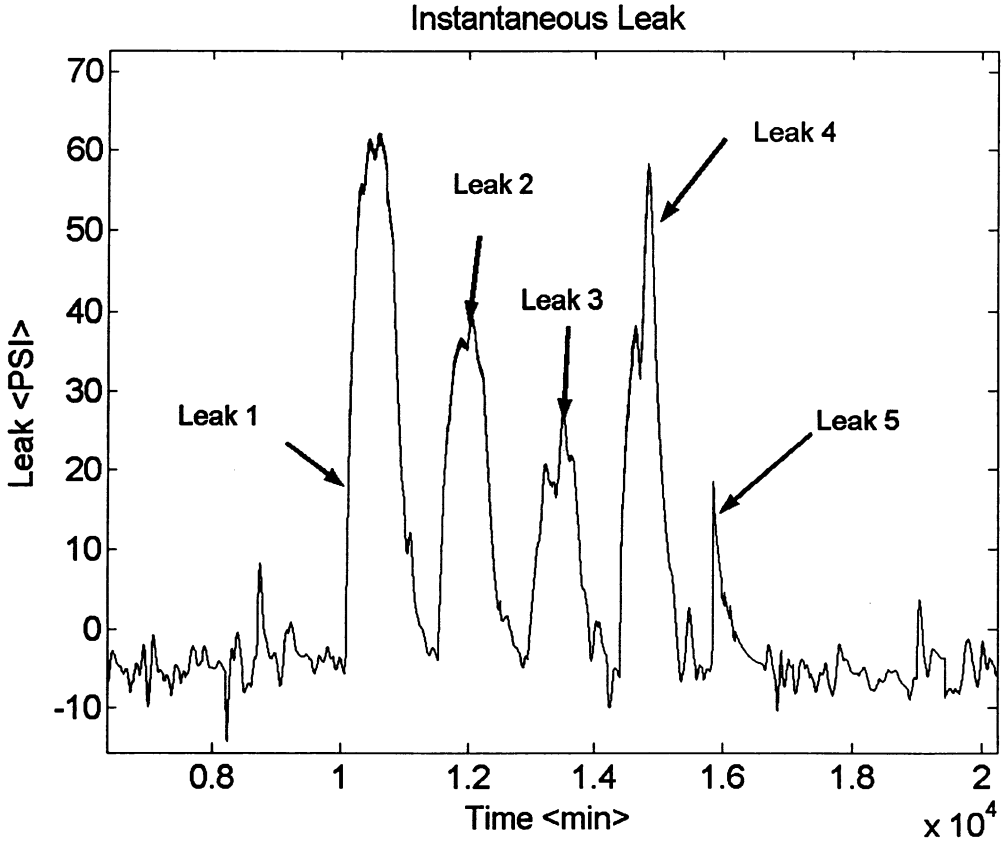


Figure 4-62 Instantaneous leak detected using the *Toil* to *Press* model.

4.2.3.3 Electric Current / Soil Temperature Based Estimator of Oil Temperature

So far, we have seen two approaches for implementing an oil leak detector. This section will briefly introduce an autoregressive model capable of forecasting the oil temperature strictly based on electric current.

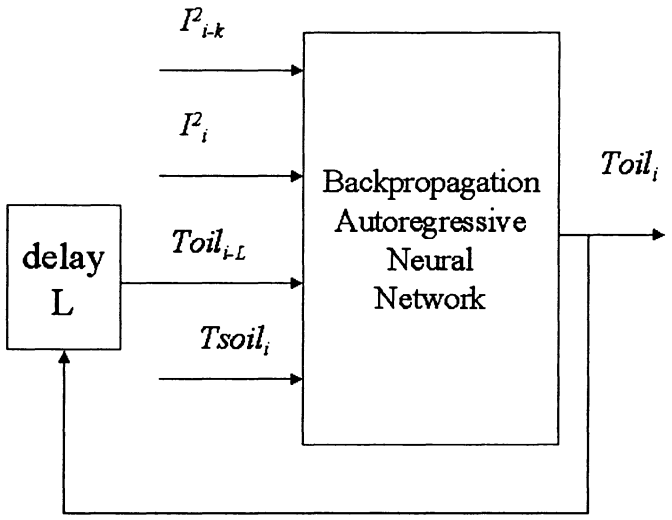


Figure 4-63 Oil temperature model based on autoregressive oil temperature, electric current, and soil temperature.

Figure 4-63 shows an arrangement identical to the one used in the previous section. The only difference is the quantity to estimate: the oil temperature replaces the corrected pressure. Figure 4-64 shows the estimation ability. It should be pointed out that, during recall, the network feeds back its own temperature estimate, and not a measured value. The plot in Figure 4-64 shows that after 4 weeks of predicting the oil temperature, one prediction per minute, the Neural Network ended up with a final prediction very close to the measured value. What is more important, the daily temperature variations are very close between the predicted and the measured ones.

An interesting point to make is about the learning process. During learning, the Neural Network does need the measured oil temperature, unlike the current based leak detector which can dispense with the oil temperature sensor completely. This means that this model cannot eliminate the oil temperature sensor, but can augment it in a Kalman filter. We will see how in the next section.

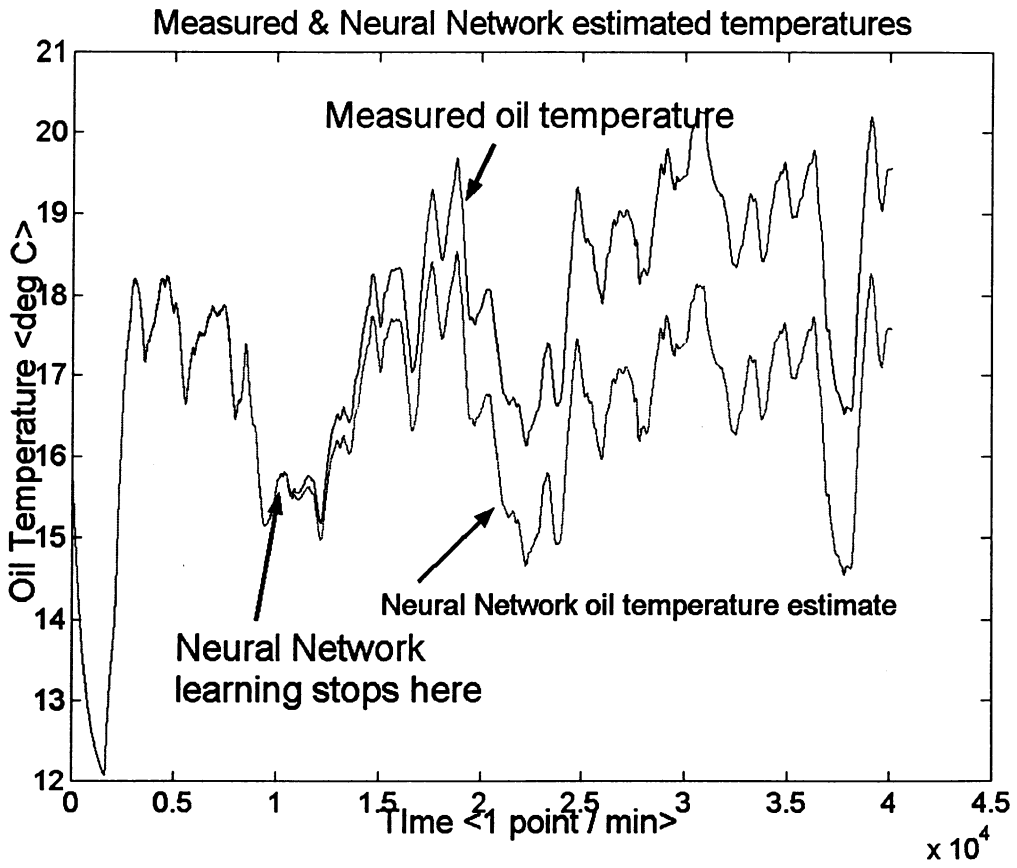


Figure 4-64 An Autoregressive Neural Network has excellent forecasting abilities.

4.2.3.4 All Inclusive Oil Leak Detector System

While the leak detectors presented so far function on their own, this section shows a block diagram of a system that includes all the techniques described. This system has several advantages:

- Redundant signals allow the detection of faults in the data acquisition and sensors, as well as leak detection
- There is a rather flexible method of balancing the false alarm / fail to alarm requirements

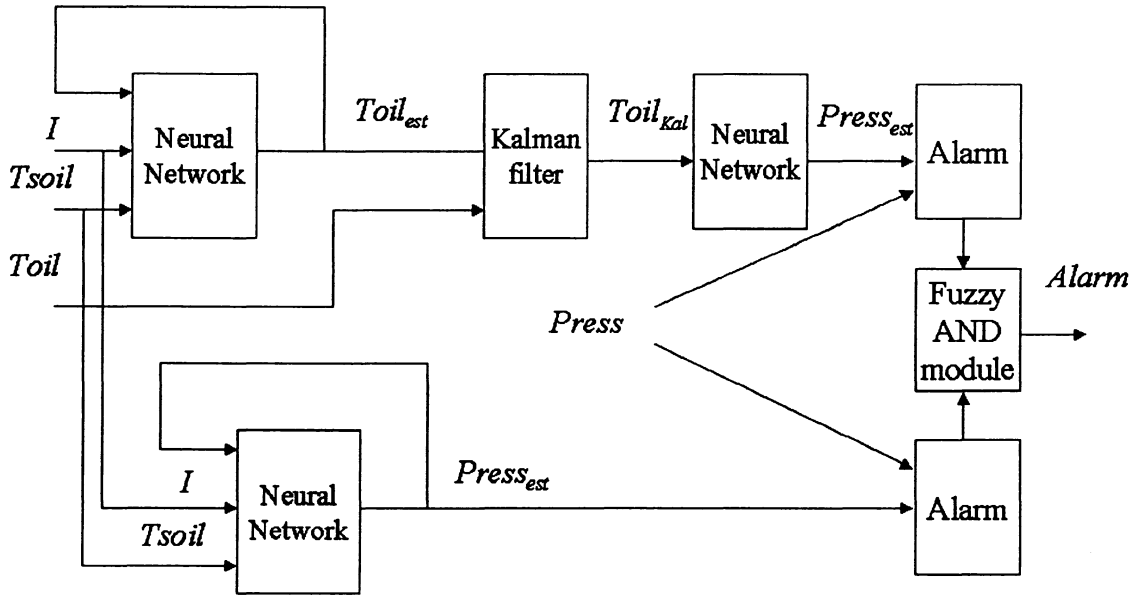


Figure 4-65 Comprehensive leak detection system.

Figure 4-65 shows the detection system integrating a number of Artificial Intelligence techniques. The bottom Neural Network implements the electric current to oil pressure detector. The top left Neural Network estimates the oil temperature based on electrical current and soil temperature. This estimate is mixed by a Kalman filter with a physical oil temperature measurement. The result is a Kalman estimate of the temperature. This

signal represents the input in the top right Neural Network which performs a oil temperature to pressure model. Alarms from the two detectors are fed into a Fuzzy AND module. The purpose of the module is to reduce the frequency of false alarms, at the cost of some delay in the alarm speed. This feature is desirable to a user, since false alarms are expensive and contribute to the user loosing confidence in the system.

4.2.4 Discussion

This case study showed a number of Artificial Intelligence based implementations to a leak detection problem. It made a number of contributions:

1. A methodology was designed to handle situations where data is distorted by elements outside the ones used for modeling. In our case the disturbing factors are pump and valve operations. However, it should be pointed out that this approach is quite general and is applicable to a host of problems. Data loss is one such application. A data gap can be simply handled by marking the gap as a disturbance. The model will learn on the rest of the data and "fill in the blanks". Of course, if a failure takes place where data is disturbed or missing, the failure will not be detected.
2. A new refinement to parameter estimation is presented for the case of autoregressive models. It should be mentioned that our approach is equivalent mathematically to tailoring the sampling rates to the modeling task at hand, which is the method suggested in the literature. The advantage of our method is its elegance. Rather than using several sampling rates within a system, the highest rate is used, and the autoregressive inputs are chosen taking input / output correlation requirements into account.
3. We have shown how a mixture of Neural Networks, Fuzzy Logic, and Kalman blocks result in a superior Fault Detection System.

4.3 Detection of Flow Restrictions in Water-Cooled Generator Windings

The last example of a Failure Detection System considers the cooling system of an electric power generator, specifically the detection of flow restrictions in liquid cooled stator windings.

The electric current produced by a power generator passes through the generator's stator windings resulting in heat buildup. To prevent overheating, pure water flows through hollow strands along the stator winding providing forced cooling of the stator bars [22]. In a generator stator for a nuclear plant, water pressurization is provided by air, nitrogen or hydrogen [23]. Figure 4-66 shows a section of a water-cooled stator bar.

Figure 4-66 Section of water cooled stator bar.

With time, there is a tendency for some of these passages to block, resulting in a local temperature rise. If such a temperature increase can be detected, the unit can be taken off line and the bars flushed therefore removing the obstruction. Failing to detect overheating may result in an accelerated aging of the machine, or a catastrophic

breakdown, a very expensive proposition. General knowledge of copper behaviour in pure water is very limited and does not provide an understanding of the mechanisms involved in the partial obstruction of hollow conductors [23]. It is imperative to monitor the temperature of the generator stator bars and to alert the operator of any abnormal higher values present for an unreasonably long period of time.

To address this problem, larger generators are instrumented with a number of temperature sensors judicially located. Since false alarms will cause the machine to be disconnected from the network resulting in considerable production loss, it is desirable to come up with a data processing algorithm which minimizes the risk of false alarm, but also maintains the high reliability of fault detection, since failing to detect is much more costly than a false detection. Building an accurate model that meets these requirements may prove to be challenging.

This chapter first introduces the published developments in the stator blockage detection field. A data analysis section will point out the characteristics of the quantities available to the failure detection system. The section on the Artificial Intelligence based system we propose, has two parts. The first one will introduce the off-line laboratory system implementation. Following this, a real-time blockage detector will be described in detail. A discussion section will conclude this case study.

4.3.1 State of the Art

A number of stator bar water temperature rise detection systems have been described in the published literature. Emshoff [22], Rioual [24], [91], and Poljakov [92] have developed models that compute the bar temperatures as a function of stator electric current, water and gas (coolants) cold source temperatures, and physical machine thermal parameters. The temperature at bar "i" is computed as:

$$T_i = a_i I^2 + b_i I + c_i T_{cold\_water} + d_i T_{cold\_gas} \qquad (4\text{-}54)$$

where I is the stator electric current and T_{cold_water} and T_{cold_gas} are the temperatures of the cold sources (cold gas temperature and cold water temperature).

The coefficients a_i, b_i, c_i, and d_i, define the fingerprints of the machine and are determined using identification techniques on data from a clean machine. The fingerprint coefficients must be computed at several machine operating modes: 95-100%, 70-80%, and 50-65% of rated load [92]. The measurements of electrical parameters must be done using measuring devices with accuracy no worse than 1.5% [92].

Using a different approach, Junquing [93] solves the machine's thermal differential equations based on boundary temperatures, coolant temperature, and machine characteristics. He uses Fourier's law of heat transfer:

$$\begin{aligned} &\frac{\partial}{\partial x}\left(\lambda_x \frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda_y \frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda_z \frac{\partial T}{\partial z}\right)+p=0 \\ &T\,|_{s1}=T_0 \\ &\lambda_n \frac{\partial T}{\partial n}\,|_{s2}=-\alpha_w(T-T_w) \\ &\lambda_n \frac{\partial T}{\partial n}\,|_{s3}=0 \end{aligned} \qquad (4\text{-}55)$$

where:

λ_x is the thermal conductive coefficient in x
λ_y is the thermal conductive coefficient in y
λ_z is the thermal conductive coefficient in z
T is the temperature of the copper strand
p is the loss per volume
T_0 is the known temperature of boundary S_1
α_w is the water convection coefficient of convection dispersion at boundary S_2
T_w is the water temperature around boundary S_2

To solve the differential equations, an approach similar to a finite element method is used. The author assumes knowledge of the boundary temperatures as well as the flow value of the coolant.

Both approaches rely on determining the stator bar temperatures based on thermal models. While such physical models can perform well, if they are sufficiently detailed, they have several downsides:

- The model will be only as good as the designer's knowledge about the physical phenomena that take place in the machine.
- The model will be highly customized to a particular machine. A system for a different generator will have to be hand crafted again.
- The model must deal effectively with the time dynamics of the system. All published models [22], [24], [91], [92], [93] assume steady state behaviour and have no time dependency included. They are vulnerable to large, quickly changing operating conditions.
- The model must deal effectively with the nonlinear relation between the electric current and the stator temperature.
- It is acknowledged that a faulty coolant temperature sensor will result in the breakdown of the whole Fault Detection System.
- It is acknowledged that electric current data loss will result in the breakdown of the whole FDS.

We propose a different approach that will overcome the above limitations by exploiting the high correlation between the temperature values amongst the different stator bars. We will prove that this method has several advantages:

- It is resilient to sensor breakdowns.
- A strongly nonlinear model is replaced by a quasi linear one, decreasing learning time.
- This approach is self-adapting being applicable to other generator installations.

4.3.2 Artificial Intelligence Based Development

Figure 4-67 shows field data collected over 10 days from a large plant generator. The number of temperature sensors monitored was 240. The plot clearly shows groupings between channels. Figure 4-68 shows data obtained from the first 10 sensors. The high correlation between sensor outputs is apparent.

Rather than attempting to predict the temperature at one physical point based on the root cause (electric current and coolant temperature), one could predict this temperature based on the temperature at all other points, thus exploiting the high redundancy between temperature readings. If a blockage starts to form, the temperature of the bar starts to increase. The correlation between this temperature and the temperatures of the rest of the sensors is obviously affected. The only time the above assumption would fail, and the blockage at one point would remain undetected, is for every bar to become obstructed simultaneously in the same proportion, a highly unlikely scenario.

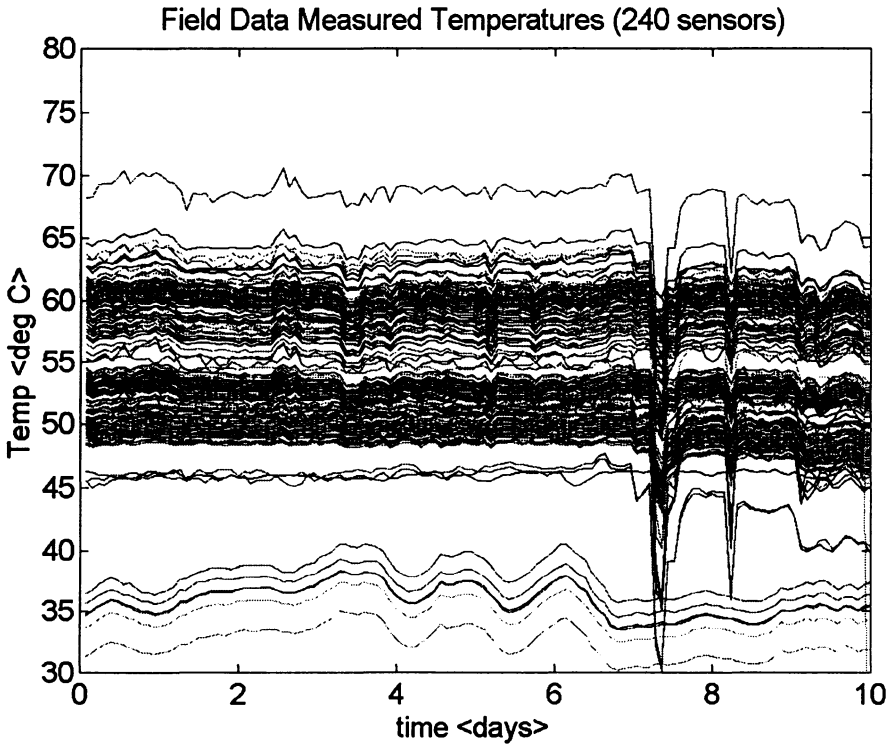


Figure 4-67 All 240 temperature sensors.

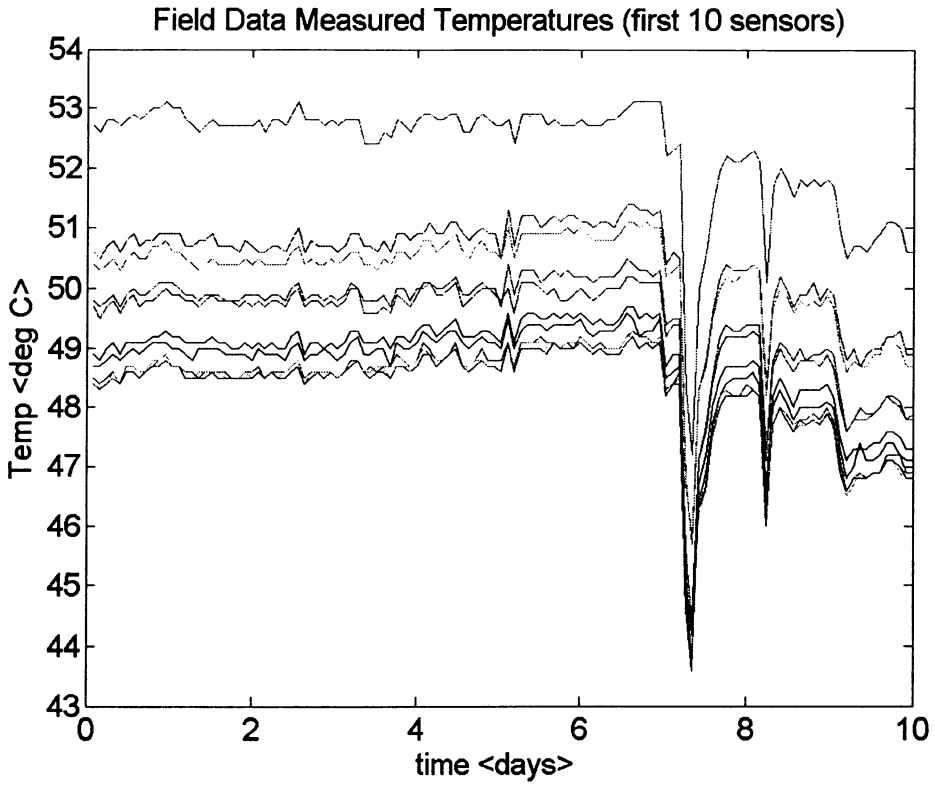


Figure 4-68 First 10 temperature sensors.

A FDS that monitors one sensor could use the outputs from the other 239 sensors as well as 2 coolant temperature readings. A 241-input to 1-output relation is established. There are 240 such analytical monitors, one for each temperature sensor.

There are several advantages to this approach, compared to systems based on electric current and coolant temperatures:

- During training, only the correlation coefficients between the different temperature points must be determined. There is no need to model the physics of thermal conduction.
- The same FDS structure applies to all machines. The correlation values will be different for different installations, and they are determined during the FDS learning stage.

- While there are time delays between a stator current change and a bar temperature change, the delays between highly correlated bar temperatures are close to zero. Therefore, the correlation-based FDS does not have to deal with these delays.
- While the bar temperature at a point varies roughly with the square of the stator electric current [25], the relationship between highly correlated temperature points is almost linear. The correlation-based FDS has a fast learning time (solution has a close form, no need for an iterative algorithm).
- A 241 to 1 model has a very low dependence on a specific input. If one of the 241 input sensors breaks down, its value is automatically replaced by its estimated value. Therefore detectors for the rest of the sensors are not affected. The strength of the approach is provided by the high input dimensionality.

The correlation-based FDS has two challenges it must meet:

1. Highly correlated input data result in a numerically ill-conditioned problem. Indeed, if the output of one of the sensors is a linear transformation of the output of another one, the determinant of a linear input/output model is zero. Only noise in the two sensors would break the perfect correlation, however the predictive abilities of the system would be compromised.

2. A high number of inputs results in a system with a large number of parameters that must be determined. This, in turn, raises the need for a large number of samples that must be made available for learning. In our case we have only 125 field samples for both learning and testing. If we use 10% of the samples for learning (12 input/output samples) and 90% for testing, the large number of parameters (240 for each sensor model) that must be computed results in an undetermined system. Moreover, even if one desired more samples to be available for learning, the low rate of sampling would put

a stringent restriction on the data acquisition stage. In our case, with a sampling period of about 2 hours, it would take several months for enough data to be available for learning.

4.3.2.1 Laboratory Off-Line Implementation

Figure 4-69 shows the detailed block diagram of a FDS responsible for monitoring one temperature point. We are going to use as many such FDSs as there are stator bar temperature sensors.

As mentioned before, all the temperature sensor values, except for the one to be monitored, are fed into the FDS. The Pre-Processing, Feature Extraction, and Modeling Blocks are included in what used to be called, the Nominal Model Block in Figure 1-1, Chapter 1. A residual value is calculated and applied to an optimum Bayes classifier for Alarm generation.

The need for the Pre-Processing and Feature Extraction blocks will become apparent shortly. A Principal Component Analysis (PCA) will be used to perform the feature extraction. The Modeling Block will be implemented using Multiple Linear Regression (MLR) or a Neural Network. The effect of PCA on a MLR block and then on a Back Propagation Neural Network (NN) will be studied. A Partial Least Squares (PLS) block will also be introduced as an alternative to the PCA / MLR and PCA / NN pairs. Finally, PLS will be combined with NN to provide non-linear modeling capability. For all the block combinations, the following questions will be studied [94]:

1. is PCA the best feature extractor that also reduces/eliminates the effects of input variable correlations while maintaining the advantages of redundancy?
2. is PLS a better alternative to PCA?
3. is the linearity of PLS a limiting factor?

4. is a nonlinear NN a better alternative to MLR?

5. is a PLS/NN combination a better performer?

Obviously, the answers to the above questions are application specific.

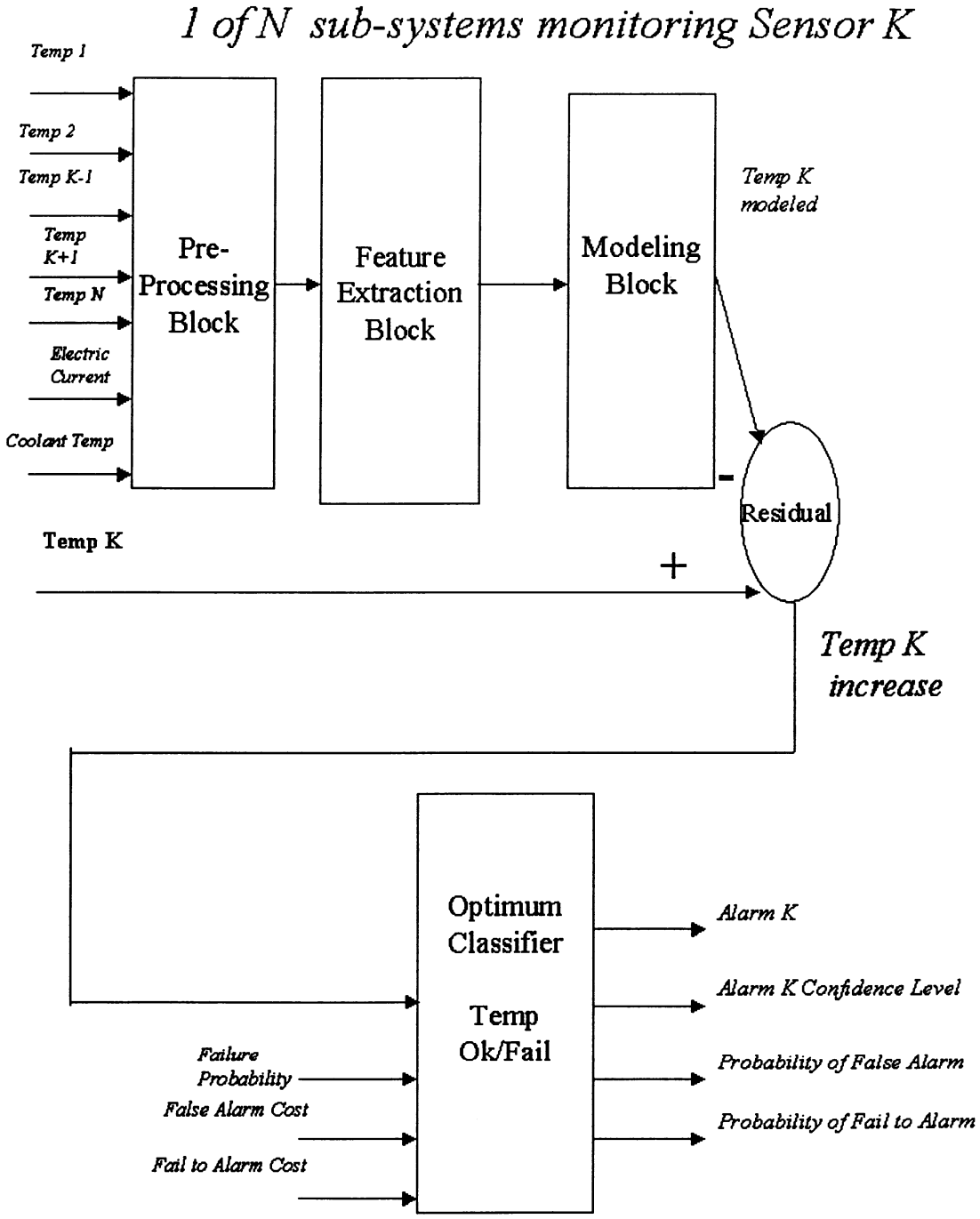


Figure 4-69 Temperature FDS block diagram.

4.3.2.1.1 The Effect of No Pre-Processing

If no pre-processing and feature extractions are performed, and the parameters of a Multiple Linear Regression (MLR) block (Modeling Block in Figure 4-69) are computed based on the first 10% of the data, the resulting block has the predictive ability shown in Figure 4-70 and Figure 4-71. The plots show the residual of the temperature measured by sensor 1 and the estimate of the MLR block. As expected, the MLR block memorized the values it was trained on (the first 12 samples) and was not able to generalize when presented with the rest of the data. The residual is zero for the first 12 samples and then it increases to 10^{14}°C for the remaining data set. Obviously, the output of the MLR block is pure noise.

One possible solution is to use only a few of the signal inputs, reducing the number of parameters to be estimated. But discarding correlated inputs would also drastically reduce the reliability of the system. An engineering approach commonly used in industry is to average all or groups of sensors, and use the resulting average or averages as inputs to the MLR block. The disadvantage of this approach is the equal weight the average puts on each sensor, regardless of the fact that some sensors may be better suited than others in predicting a certain physical effect (also the noise level may vary between sensors).

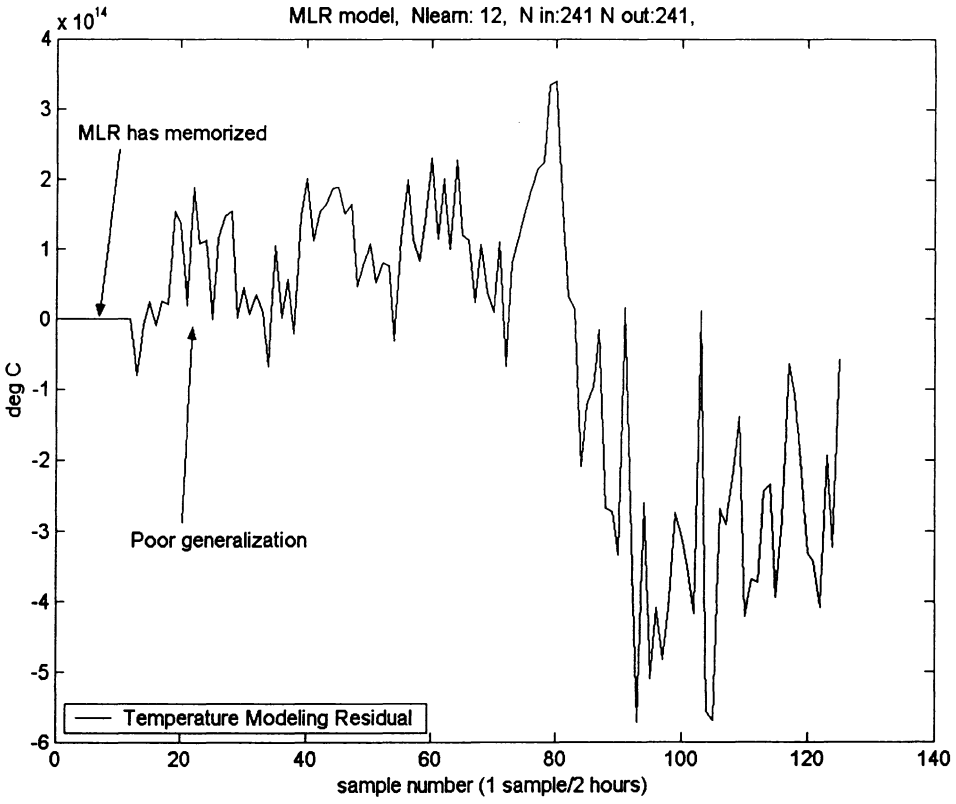


Figure 4-70 MLR residual.

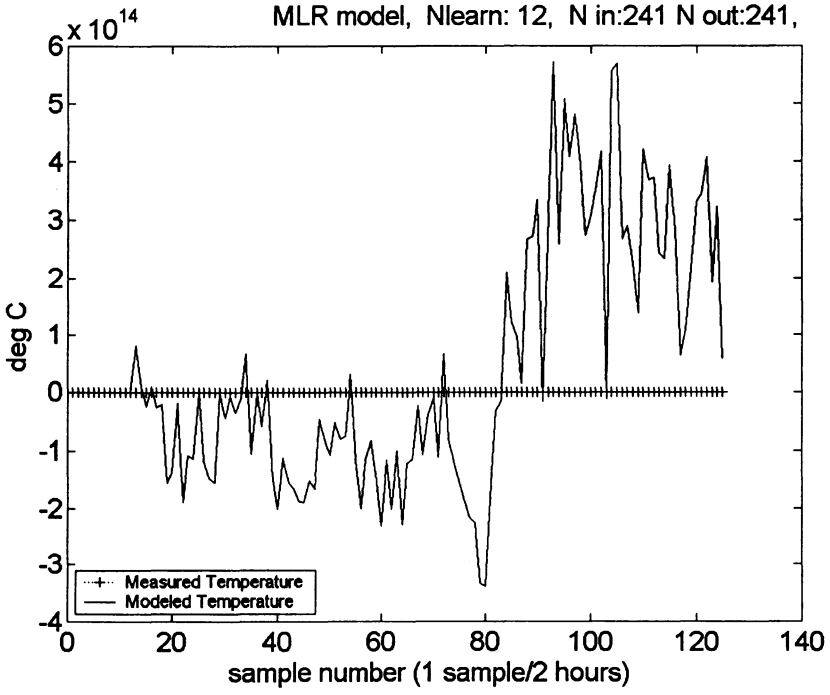


Figure 4-71 MLR output for sample 1 is 10^{14}.

4.3.2.1.2 Principal Component Analysis Preprocessing

A consistent way to arrive at a weighted average of the signal inputs is by using Principal Component Analysis (PCA) [95], [96], [97]. The pre-processing block will subtract the mean from each temperature input and rescale the input to have a standard variance. The feature extraction block will be performed by PCA. We will connect the output of the PCA block to the input of an MLR block responsible for the temperature modeling. The PCA block replaces its N inputs (241 in this case) with M outputs (3 in our case). Each component of the M outputs is a linear combination of the N inputs. PCA chooses the weights in such a way as to minimize the information loss introduced by the compression. To see the effect of PCA, Figure 4-72 plots the first 3 dimensions of the input data set while Figure 4-73 shows the 3-dimenssional output of PCA (in this case we did a compression 241:3 in order to be able to plot the output; the actual compression ratio is problem dependent)

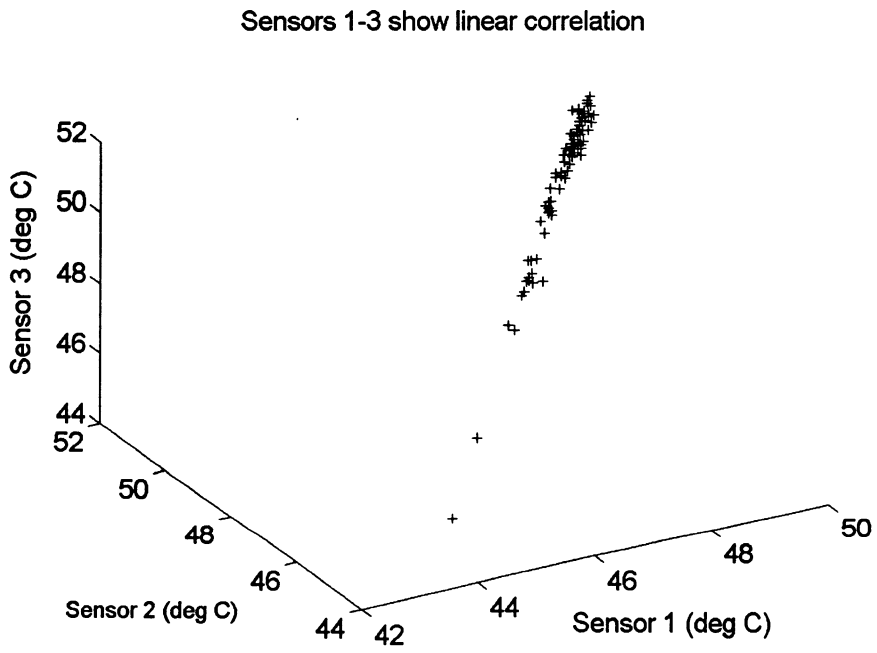


Figure 4-72 The first 3 input dimension are shown

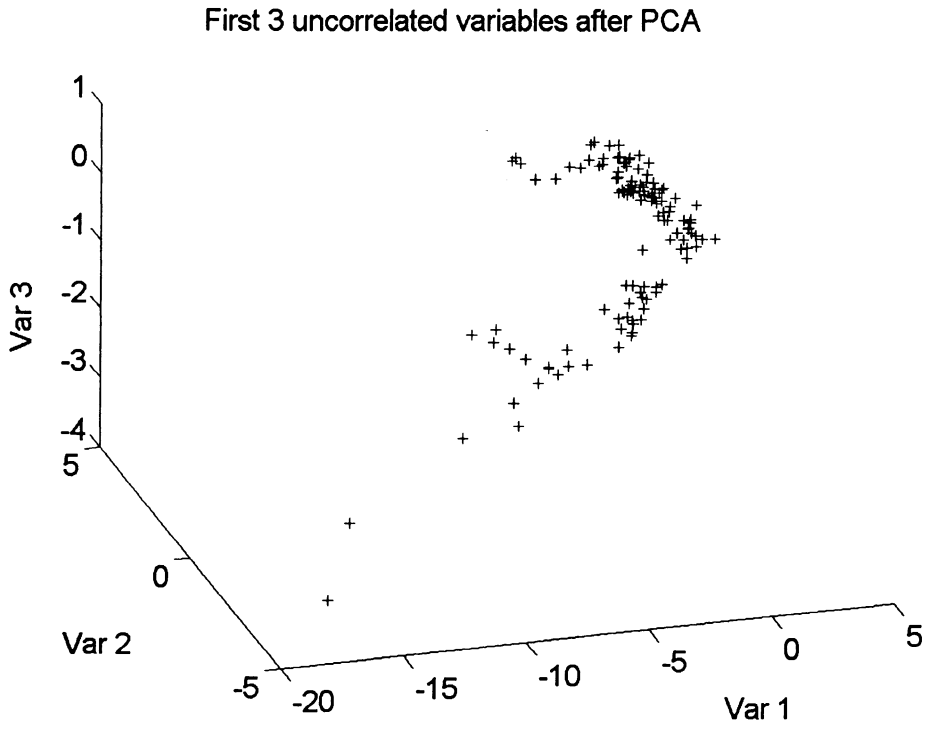


Figure 4-73 Data set after PCA.

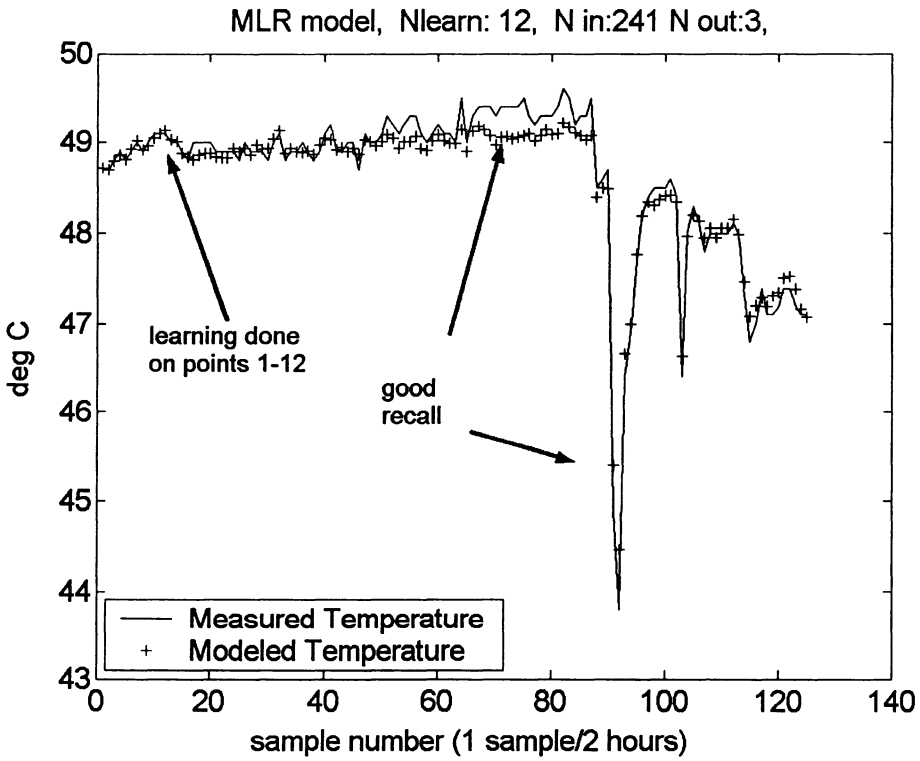


Figure 4-74 PCA/MLR output.

As can be seen, PCA breaks the strong correlation between input quantities. It should be pointed out that PCA has no way of knowing what the decomposition will be used for (it does not know that its output will be fed into another block which will model a temperature). Therefore the applied transformation may not be optimum from a modeling point of view. Another point to make is that PCA does a linear transformation. It is possible that a higher compression ratio could be achieved if a nonlinear transformation is used. Figure 4-74 shows the output of the PCA/MLR algorithm while Figure 4-75 shows the disagreement between the measured quantity and its estimate.

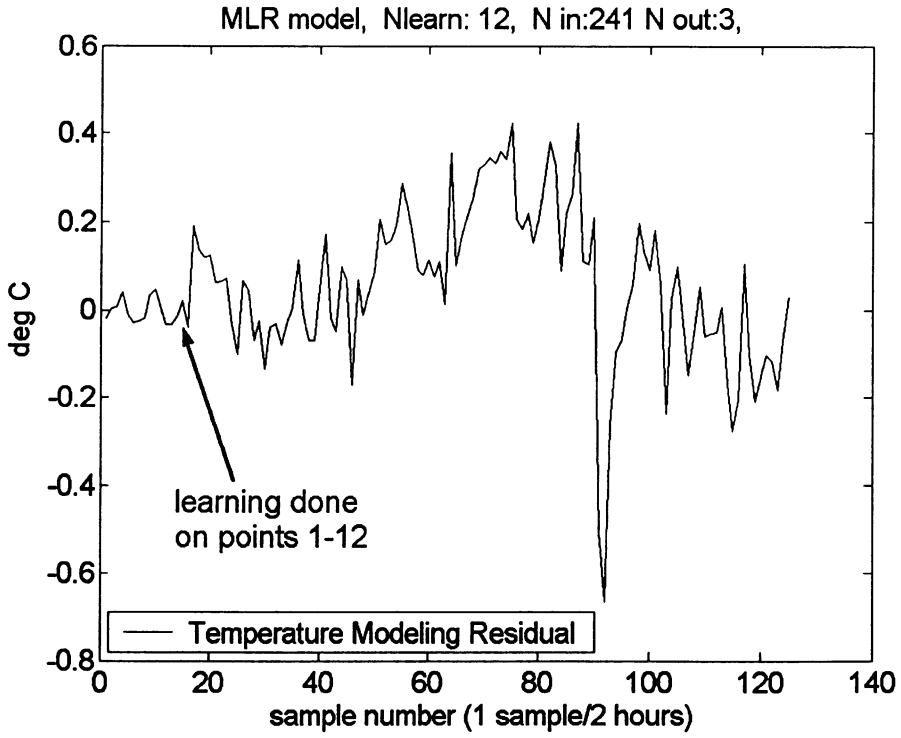


Figure 4-75 PCA/MLR output disagreement.

It is interesting to note that training MLR on the first 12 samples has resulted in the residual for sensor 1 being less than 0.05°C for the first 12 points. From then on the disagreement increased to +/- 0.3°C with a maximum of 0.7°C experienced during the large temperature variation at sample 92. It is expected that an increase in the training data set size would have reduced this error even more.

The next step is to replace the MLR block with a Backpropagation Neural Network (NN) and assess the benefits of non-linear modeling. Figure 4-76 shows the output of the

PCA/NN algorithm while Figure 4-77 shows the disagreement between the measured quantity and its estimate.

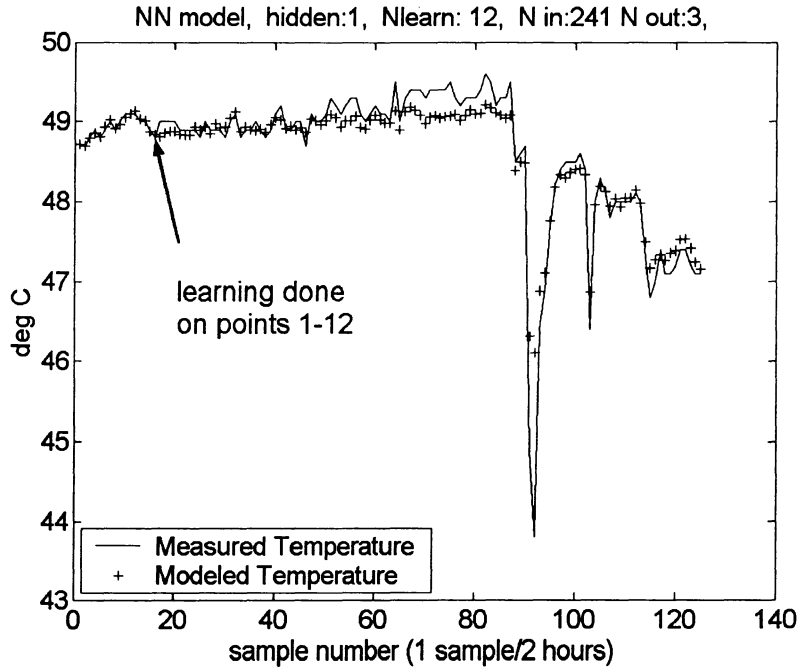


Figure 4-76 PCA/NN output, 1 hidden node.

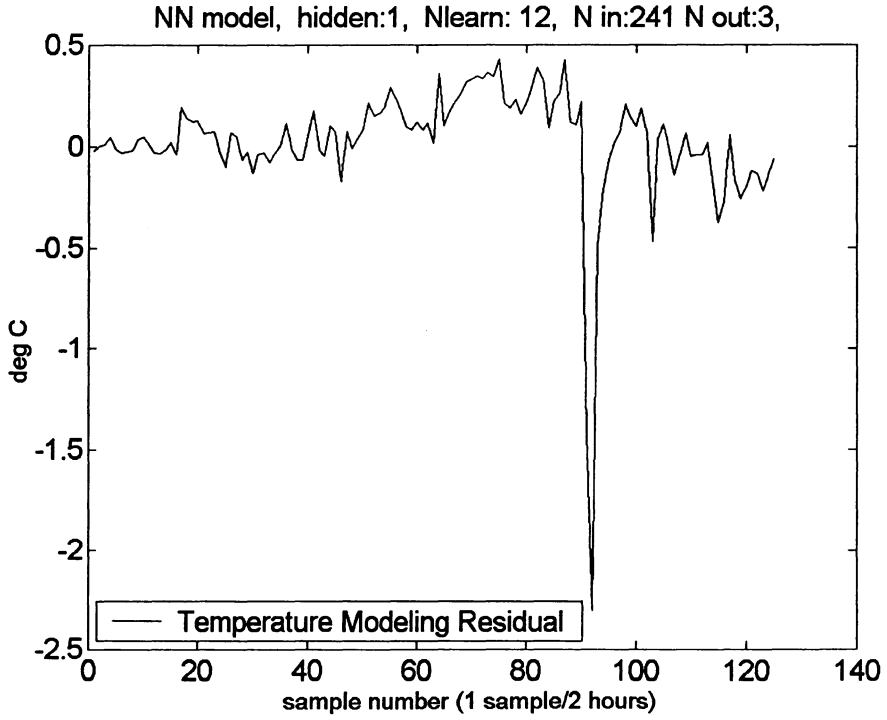


Figure 4-77 PCA/NN output residual.

While the initial performance of the NN was slightly better than the MLR block, the maximum residual obtained is over 2°C compared with 0.7°C produced by MLR. The nonlinear modeling ability of the NN allowed it to memorize better than MLR over the 12 training samples rather than generalize. To verify this fact, the training set size was increased to 91 samples (in order to include part of the large temperature swing which increased the information content). Figure 4-78 and Figure 4-79 show that NN can accurately estimate the temperature reading.

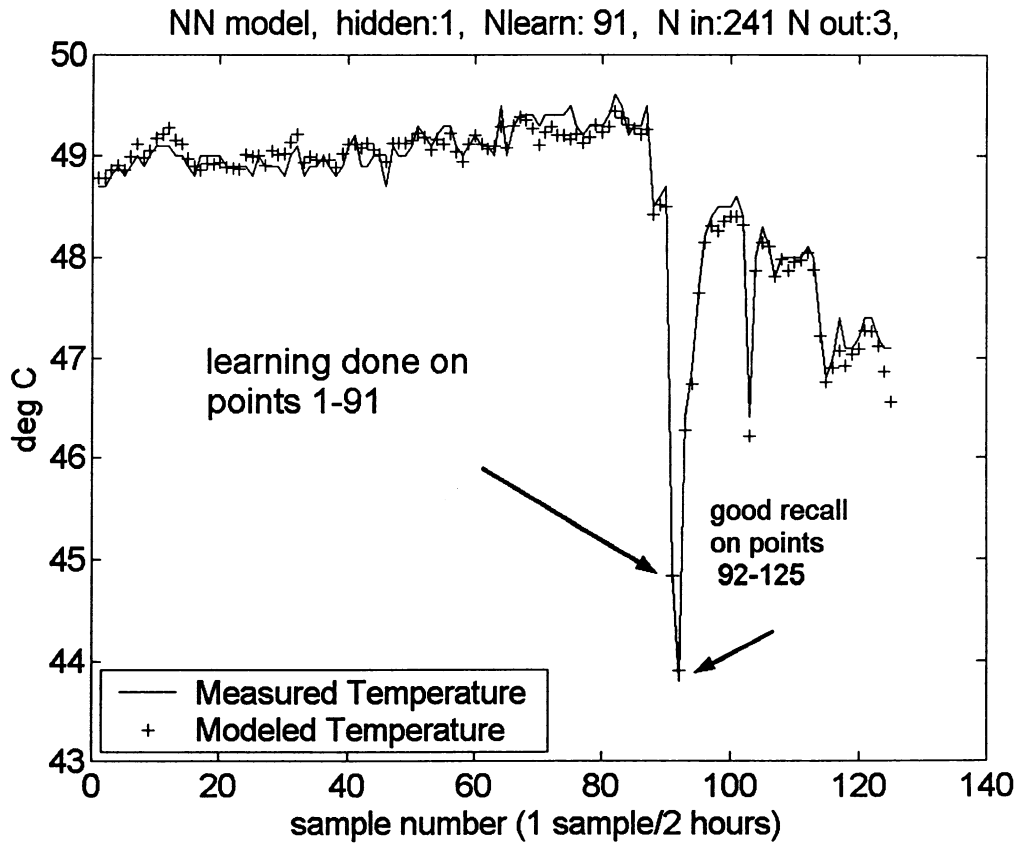


Figure 4-78 PCA/NN, 1 hidden node, 91 training samples.

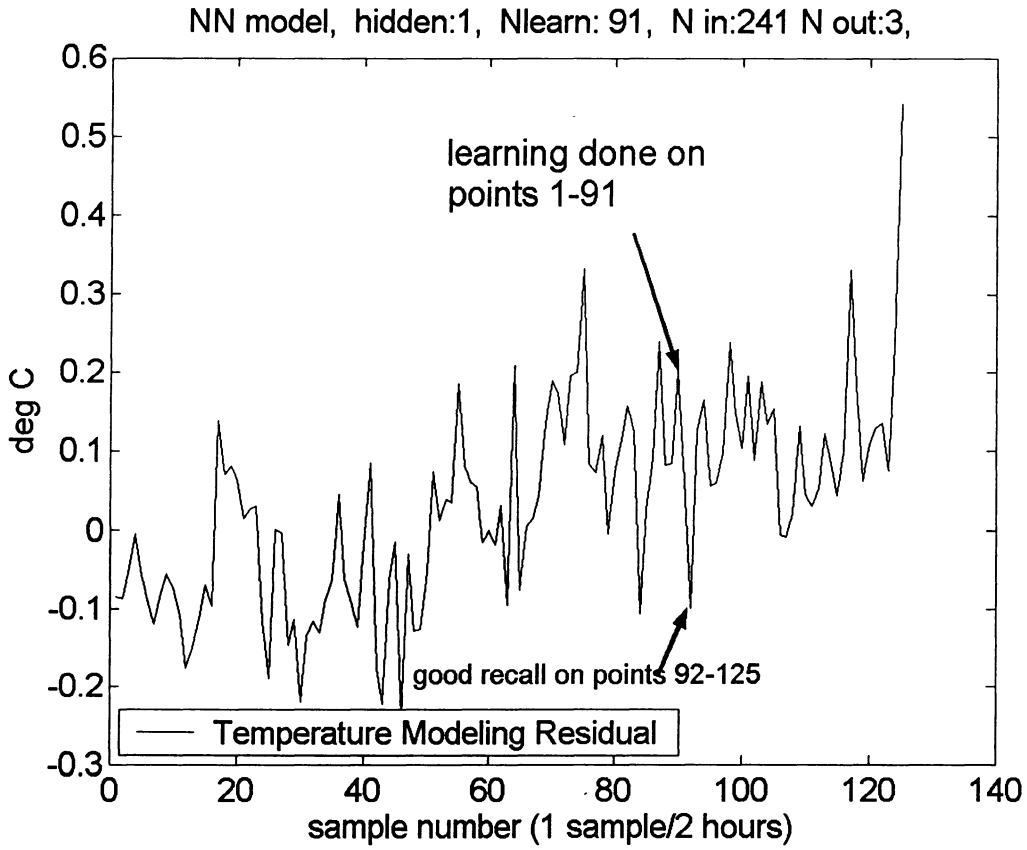


Figure 4-79 PCA/NN 91 point learning set residual.

4.3.2.1.3 Partial Least Squares Preprocessing and Modeling

As discussed in the previous section, PCA breaks the correlation between input variables with no regard for the usage of this transformation. PCA finds factors that capture the greatest amount of variance in the predictor variables. MLR seeks to find a single factor that best correlates predictor variables with predicted variables. Partial Least Squares (PLS) attempts to find factors which do both, i.e. capture variance and achieve correlation. [98]. Now, both compression and modeling are performed by one linear transformation. Figure 4-80 shows the output of the PLS algorithm while Figure 4-81 shows the disagreement between the measured quantity and its estimate.

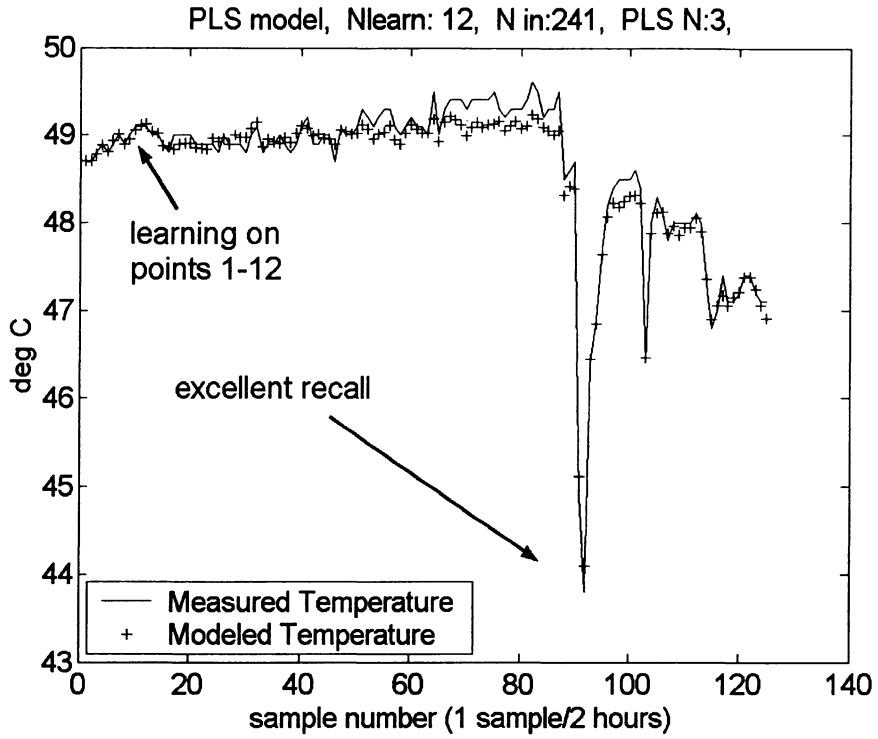


Figure 4-80 PLS 3 latent variables.

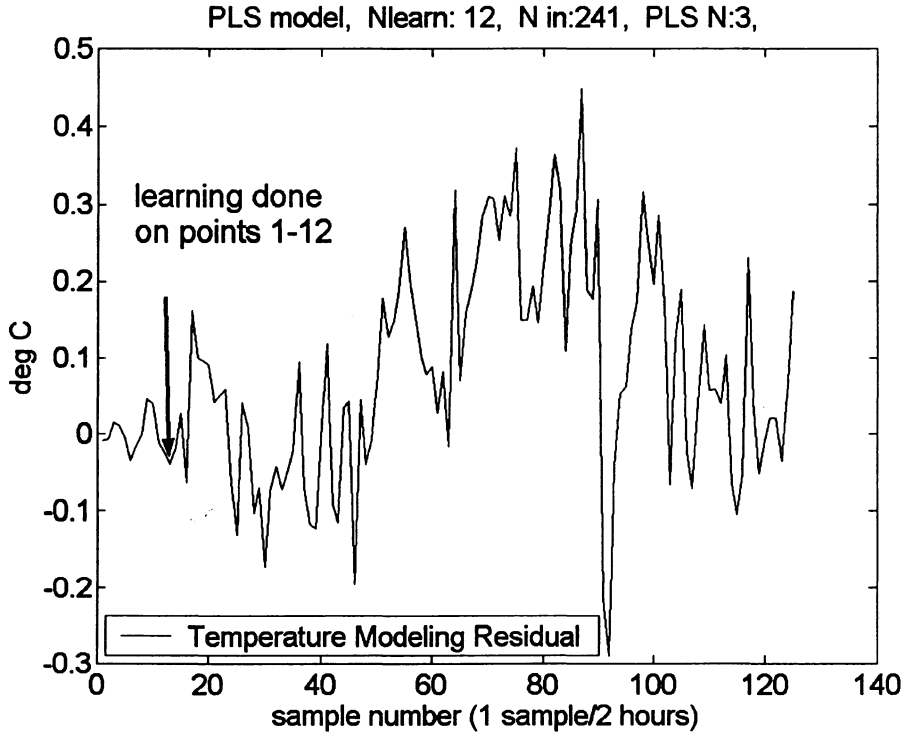


Figure 4-81 PLS output disagreement.

While the PLS disagreement over the training set was about half compared to the PCA/MLR algorithm, the performance over the whole set was comparable.

4.3.2.1.4 Neural Network - Partial Least Squares Preprocessing and Modeling

Finally, a PLS block which uses a non-linear Neural Network rather than a linear model is studied. The PLS algorithm determines first the latent variables and then establishes a nonlinear relation with the predicted quantity. Figure 4-82 shows the output of the NNPLS algorithm while Figure 4-83 shows the disagreement between the measured quantity and its estimate.

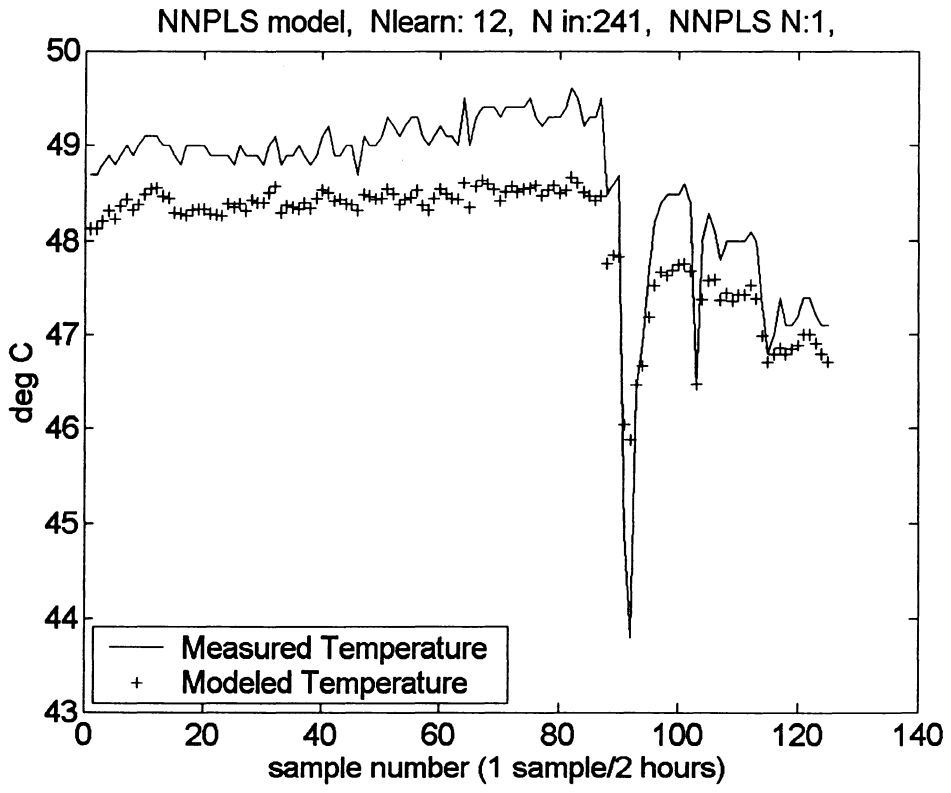


Figure 4-82 NNPLS, 1 latent variable, up to 6 hidden nodes.

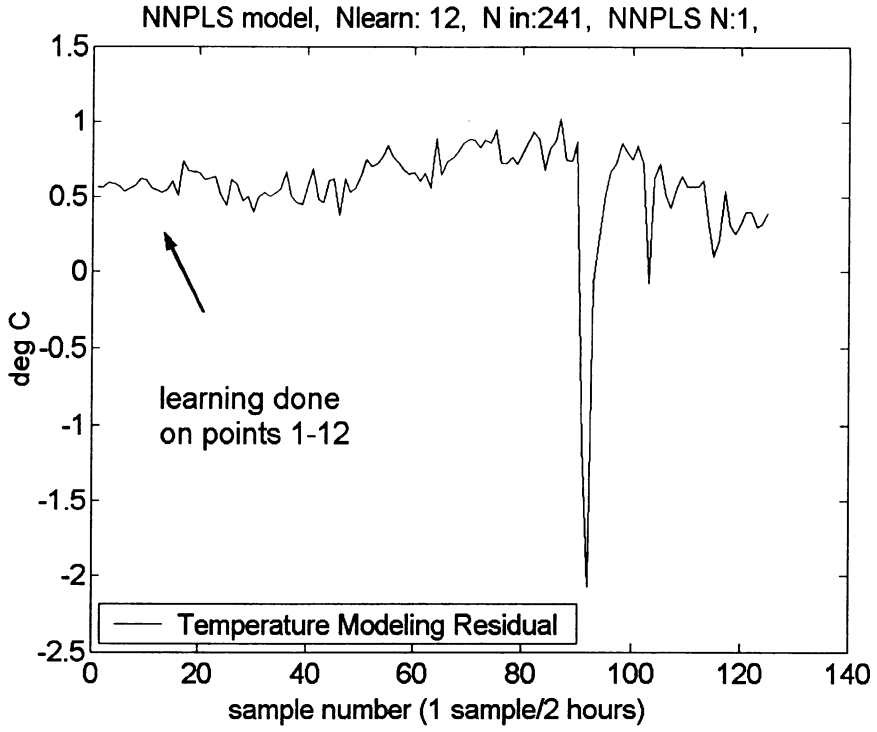


Figure 4-83 NNPLS residual, 1 latent variable, up to 6 hidden nodes.

Except for a bias in the output, the NNPLS has a performance virtually identical to the PCA/NN algorithm. This is to be expected since the PCA block was set to reduce the input dimensionality from 241 to 3. As it could be seen in Figure 4-73, even after this compression, the output from PCA could be contained into a plane, therefore the compression that could be achieved is down to 2. This means that PCA compressed the input data while maintaining most of its information content, allowing the Neural Network block to do the required modeling. No tangible benefit is achieved by using NNPLS which optimizes the compression keeping modeling in mind.

4.3.2.1.5 The On-Line Bayesian Adaptive Alarm Decision Block Alarm Decision Block

The new adaptive alarm generation techniques presented in Chapter 3.3.4 will now be applied to field data collected from a large electric power generator. Several hundred

thermocouples monitor the temperatures in the stator bars. To study the behaviour of the classifier, we are going to consider only one sensor.

The measured temperatures are from a clean generator. This study will demonstrate the ability of the alarm block to process signals having realistic noise statistics. To explore the system response in the presence of overheating, an artificial ramp temperature increase will be included in the signal.

The difference between the measured temperature and its estimate is computed and the values of the disagreement (residual) are used for diagnostic. Figure 4-84 and Figure 4-85 show the behaviour of the FDS system. Four plots are shown: the disagreement, the threshold, the alarm, and the prior. A noise spike of 0.4°C is artificially added to the disagreement in order to study the effect of unusual disturbances. The FDS prior for the OK state is initialized to 0.5. It can be seen that as a result of a low noise level, the prior increased its value from 0.5 to almost 1. As a result, the pdf(D|OK) is scaled up by the adaptive Bayes algorithm, while pdf(D|FAILED) is scaled down (Equation 3-13). The intersection between the two pdfs has shifted resulting in an increase of the threshold from 0.2°C to 0.45°C. The artificial spike at time point 5 has had as an effect, the lowering of the prior for the next point as well as the lowering of the threshold. Once low level disagreement values are observed, both the threshold and prior move back to their upper limits.

It should be pointed out that, technically, there is no upper limit for the threshold value. It is expected that a low noise disagreement could result in an asymptotically increasing prior for the OK state, while the prior for the FAILED state would asymptotically move towards zero. Because of this fact, while running the system under MATLAB, numeric underflows were experienced. The solution used was to limit the minimum value of the prior to 10^{-5} and its maximum value to 0.99999.

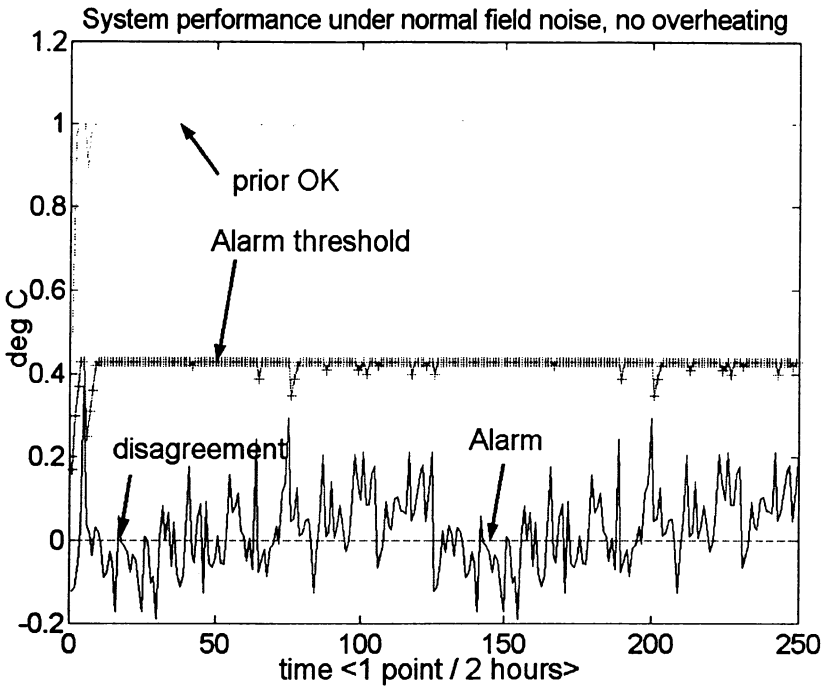


Figure 4-84 System performance under normal conditions, no overheating, field noise.

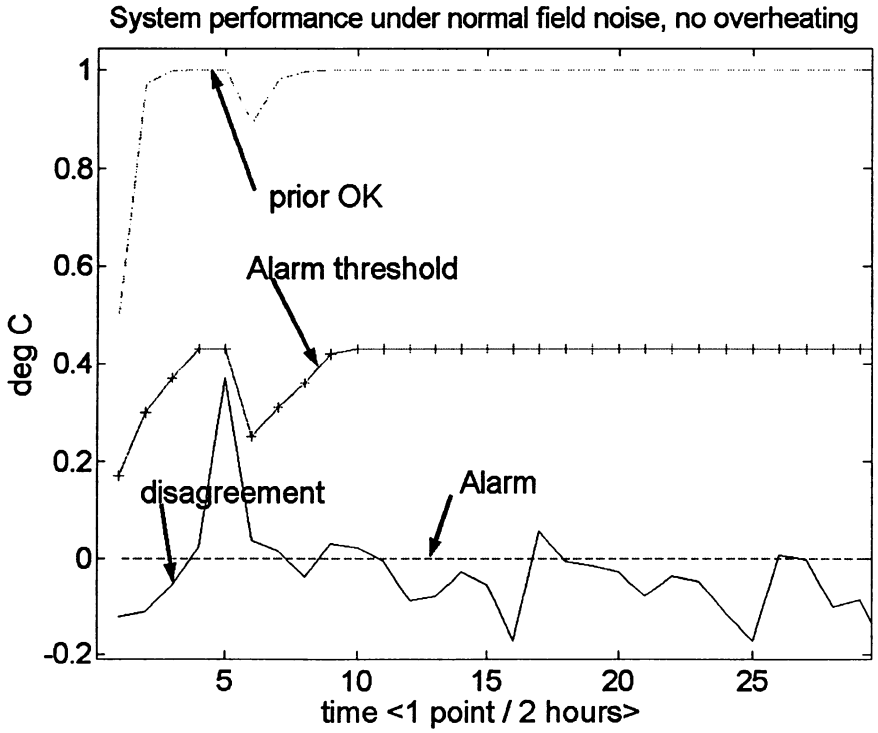


Figure 4-85 System performance expanded around the temperature spike area.

In order to test the FDS in the presence of an overheating condition, an artificial temperature increase is added to the observed disagreement values.
Figure 4-86 and Figure 4-87 show the result

The temperature increases linearly up to 1.2°C. The system alarms once the disagreement reaches 0.38°C. It should be pointed out that the noise spike that resulted in a disagreement of 0.41°C correctly did not alarm the system. The system has finally alarmed at a lower value because several residual values were large, and not just one of them. This shows the superior behaviour of the adaptive threshold scheme over a fixed value arrangement. The alarm is turned off once the disagreement value is below 0.12°C.

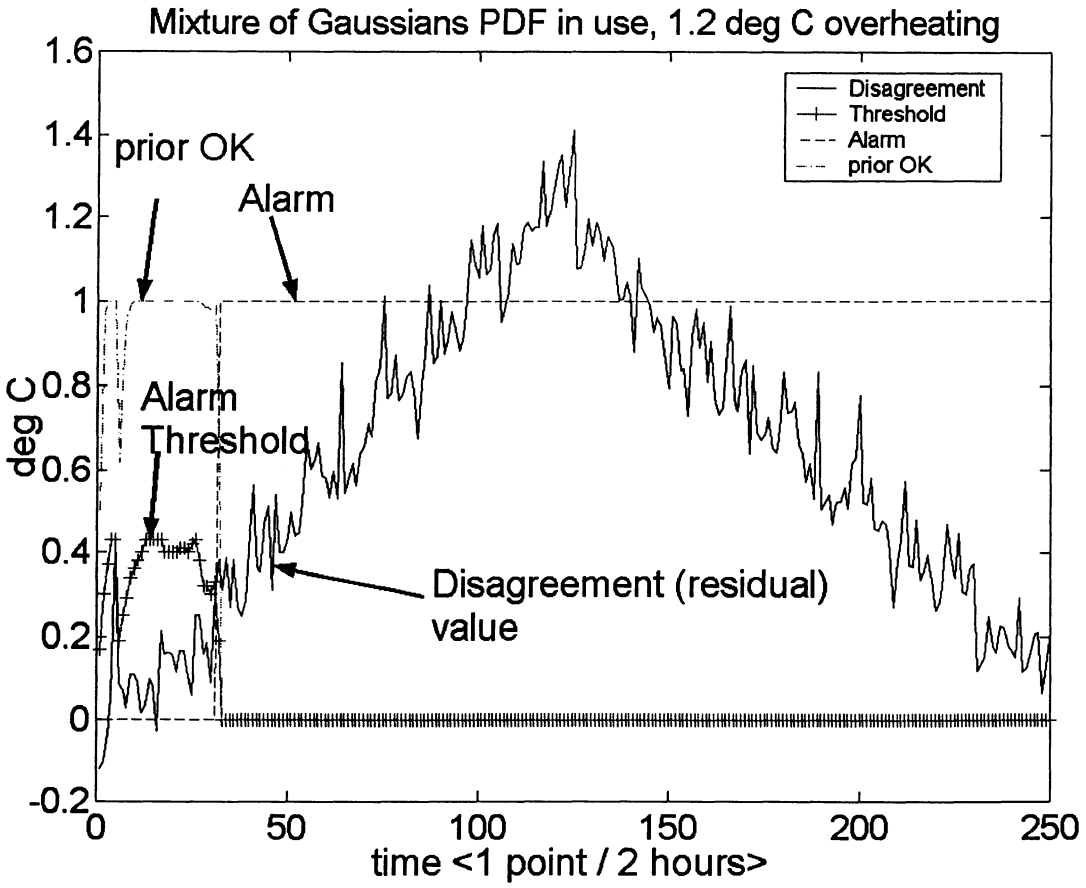


Figure 4-86 System performance, 1.2°C overheating, field noise.

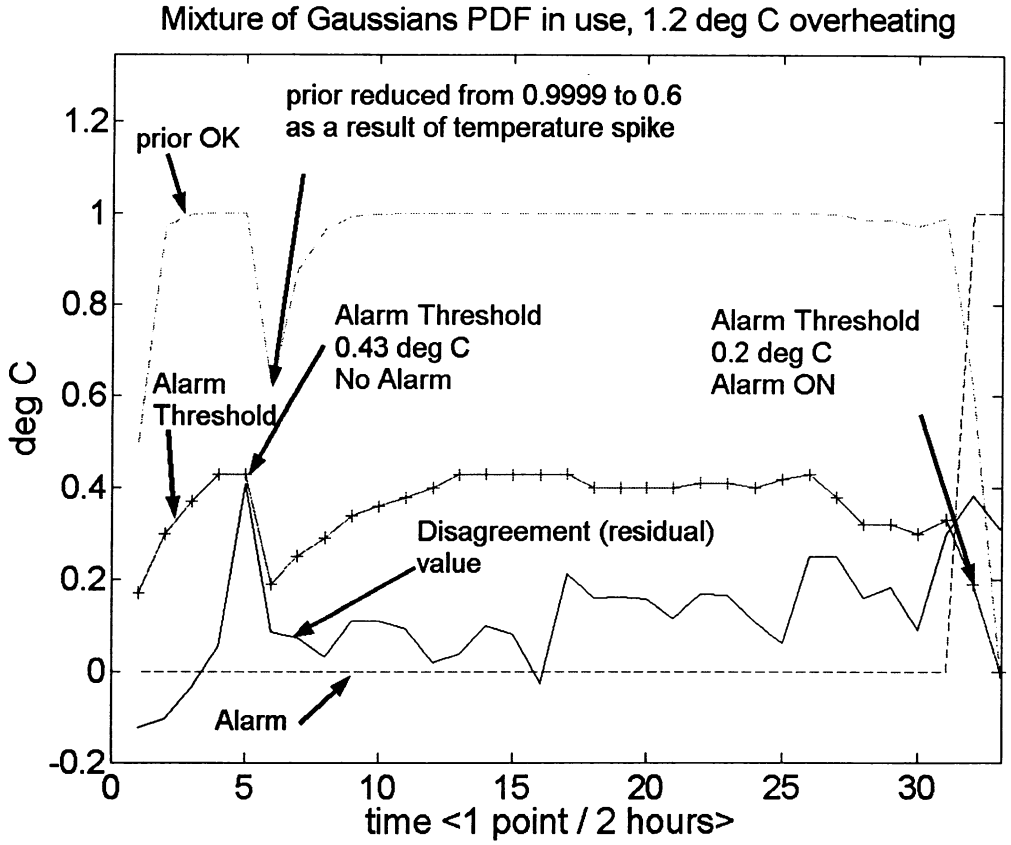


Figure 4-87 Expanded view of temperature spike area at time point 6.

As expected, the algorithm automatically estimates the value of the standard deviation of pdf(D|OK) and builds the pdf(D|FAILED) accordingly. As can be seen in Figure 3-6, lowering the value of P(FAILED) will not have a dramatic effect on the threshold position until the pdf(D|FAILED) will be actually below pdf(D|OK). When this happens, the intersection of the two curves will be to the right of their maximum points. Ironically, a low noise level (higher quality data), results in the two pdfs having very sharp roll offs. The result is that an intersection of the pdfs, which takes place to the right of their maximum, is an ill-conditioned problem; the two curves are almost parallel and horizontal. Better data result in a worst threshold localization. In order to alleviate this problem and also to manually increase the system threshold, and provide better immunity to spike noise, the standard deviation computed by the algorithm is artificially increased. This results in pdfs having smoother roll offs. Figure 4-88 shows the behaviour of the

system having a used standard deviation 4 times larger than the actual one computed from field data.

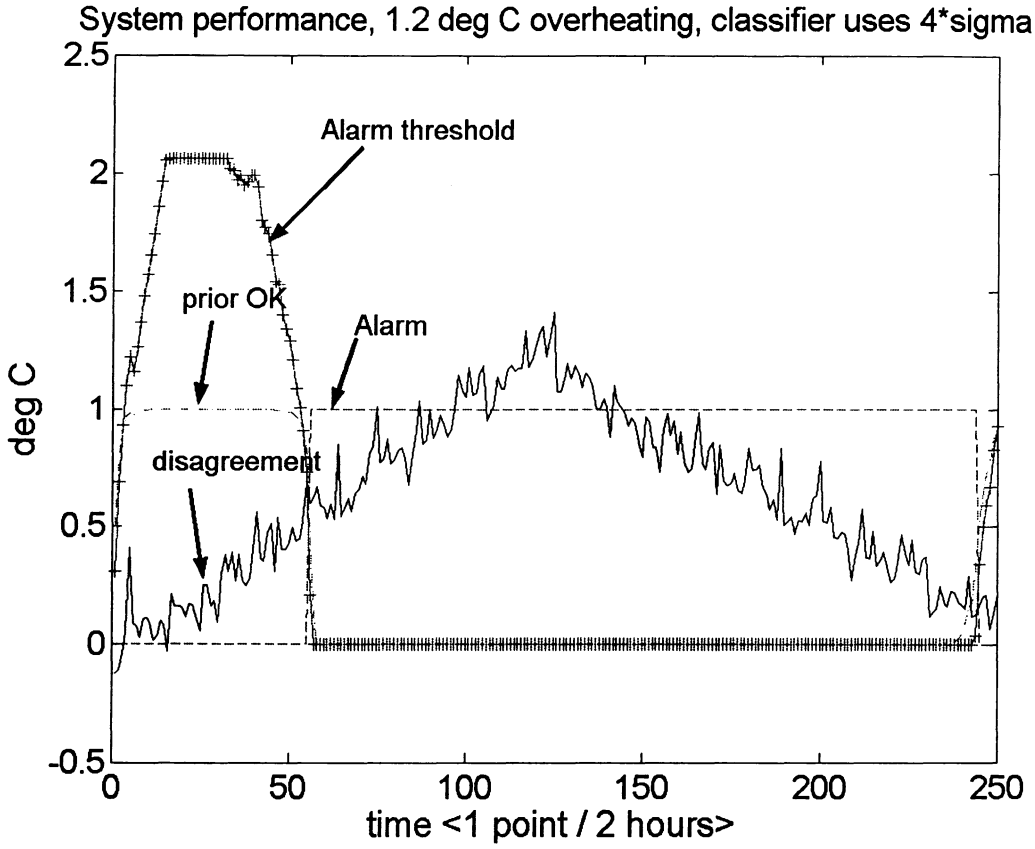


Figure 4-88 System performance, 1.2°C overheating, high alarm threshold.

As expected, the threshold value is larger providing better noise immunity. While the system now alarms at 0.6°C rather than the previous 0.38°C, the 0.22°C lost in late alarm is more than made up in an extra 1°C that can be tolerated in spike noise. Increasing the spike from 0.43°C to 1.4°C results in no false alarm.

Finally, the behaviour of the classifier is shown in Figure 4-89 where the pdf of the failed system is produced using the Mamdani Fuzzy Logic inference engine developed in Chapter 3.3.3. We made no attempt to fine tune the output fuzzy logic sets in order to compare the behaviour of the adaptive Bayes classifier using the Fuzzy Logic pdf and the

mixture of Gaussians pdf. Which approach is to be used, is application dependent. We are simply proposing several alternatives to solve this problem.

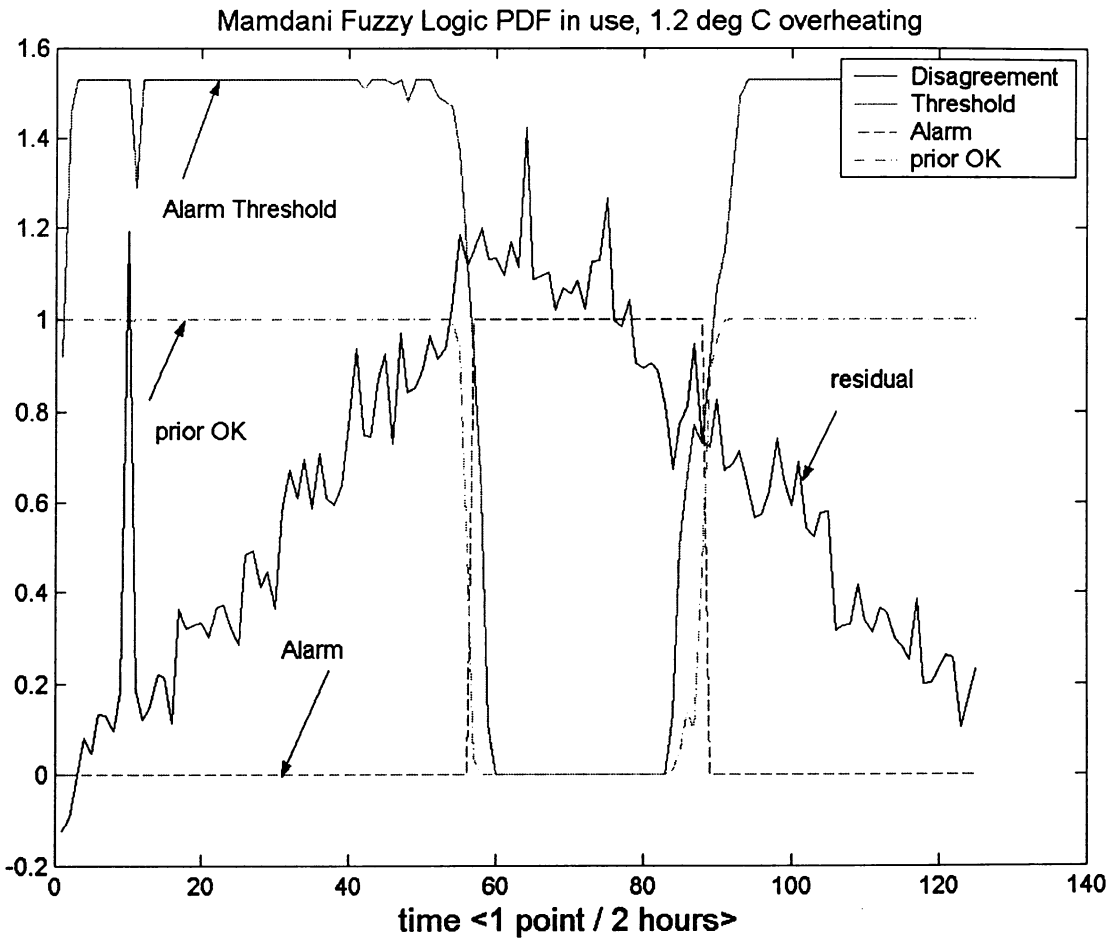


Figure 4-89 Fuzzy Logic produced pdf.

4.3.2.2 Real-Time Implementation

The previous sections have shown the different algorithms needed for the detection of abnormal temperature increases at different points on the generator's stator bars. We now present one possible real-time implementation of the FDS.

The different modules of the FDS run on several computers, all interconnected. The proposed communication approach is the use of Ethernet and the TCP/IP protocol. We have identified the need for: failure detection and alarm generation, data storing, and data viewing and analysis. Figure 4-90 depicts the different software modules and the location of the computer used to run a specific module.

The failure detection and alarm generation are expected to run on a computer located in the plant Control Room. This machine will interact with the data logger responsible for collecting the temperature measurements, and will produce alarm outputs to be presented to the operator. Since the data logger is located close to the generator, a modem to modem communication, using an RS232 port handles the exchange between the computer and the data logger. The RS232 Data Link Server module passes the information from the data logger to the Blockage Detection module, and when a result is available from this module, sends it to the Data Base Interface module. All three applications, the Blockage Detector, RS232 Data Link, and the Data Base Interface run on the Control Room computer.

The Data Base Interface module, passes the data to a data base application. We have implemented an SQL based data base module. This application can run on the same Control Room computer. However, given the fact that, for security purposes, the Control Room is a restricted area, and given the data base maintenance procedures (tape backups, etc.), it is desirable to have the data base running on an off-site location.

Finally, the data base is accessed by a number of users who view the data, and perform data analysis.

The FDS prototype has been implemented using a number of computer languages and environments: compiled MATLAB, C++, and objective Pascal.

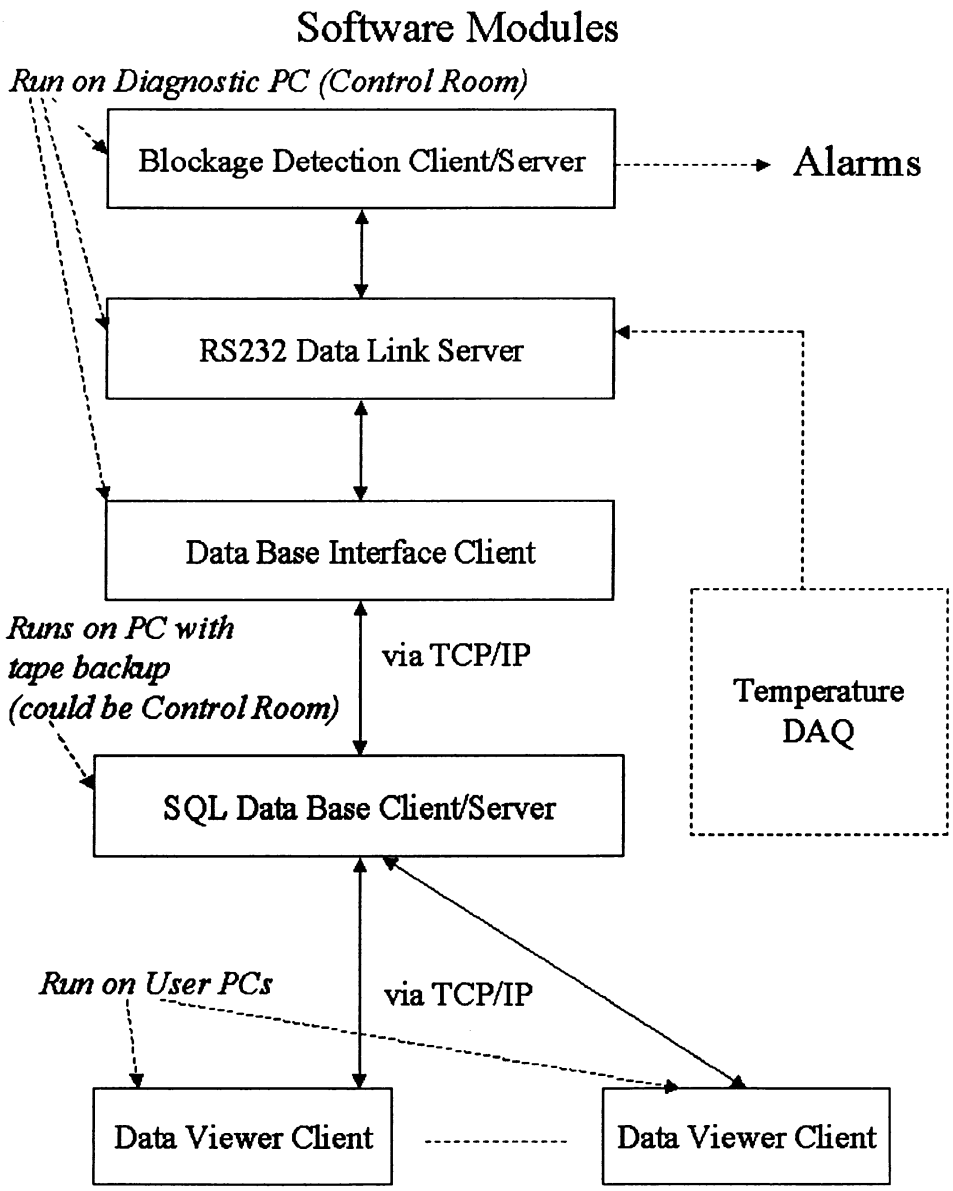


Figure 4-90 Real-time blockage FDS software block diagram.

4.3.3 Discussion

This section has described a Fault Detection System responsible for detecting flow restrictions in liquid cooled stator windings of electric power generators. This case study is relevant since it is a practical example of a FDS that must process a large number of highly correlated signals. Two major parts of the FDS are discussed: the modeling block and the alarm generator block.

Different pre-processing and modeling approaches have been discussed. It is determined that, in this example, a PCA block performing data compression followed by a MLR model achieves good performance. No tangible gain is obtained by using a Neural Network non-linear model, nor is there an advantage in using a targeted compression done by PLS. These facts suggest that the presented example is almost linear in nature and the observed disagreement is not caused by modeling errors but by signal noise.

The adaptive Bayes classifier developed in Chapter 3.3.5 has been successfully applied to this application. This approach has a number of advantages over other classical algorithms:

- determines an optimum alarm threshold (it minimizes the probability of making a mistake)
- threshold changes with the acquisition of more data (method is adaptive)

Careful consideration must be given to the following, previously mentioned, issues:

- positive feedback is used (compute Prob(OK|D), then feed it back). The system becomes more & more confident when the residual stays reasonably small.
- low noise data result in very narrow Gaussians, whose intersection point may be hard to compute accurately (the intersection point is the threshold of the alarm system)

We have solved the challenges mentioned above by:

- limiting the a priori probability of the system being OK to some maximum value
- limiting the alarm threshold to some maximum value
- forcing the Bayes classifier to compute its adaptable threshold as if the data contained more white noise than it actually did

CHAPTER V

5 Discussion

This work has presented several novel applications of Artificial Intelligence-based algorithms to Failure Detection Systems. It started by providing a framework for FDS design. Then it showed the benefits intelligent, adaptive blocks could provide along with potential pitfalls. A new fault detection structure was introduced which has desirable properties when dealing with missing data, or data corrupted by extraneous disturbances. The convergence of the learning algorithm developed for this detection structure, was proven.

A classical alarm generation procedure has been extended being transformed into an optimum, real-time, adaptive block. Artificial Intelligence techniques have been compared side by side with classical approaches and the results have been analyzed.

Three practical examples have been discussed: The Static Security Assessment of Electric Power Systems, the Oil Leak Detection in Underground Power Cables, and the Stator Overheating Detector. These case studies have been demonstrated since each one represents a class of failure detection problems.

We will now outline the main benefits and disadvantages of the Artificial Intelligence technologies. The Conclusions and Further Research will complete this work.

5.1 Benefits and Disadvantages of the Artificial Intelligence Technologies Compared with Classical Implementations

The ability of Neural Networks to learn non-linear functions makes them powerful model builders. Indeed, based on Kolmogorov's theorem [99], Feed Forward Back-propagation

Neural Networks are universal approximators needing at most 2 hidden layers and a suitable number of hidden nodes. This is a strong point supporting NN. However, one must be careful when adding nodes to a NN when trying to improve learning accuracy. While Kolmogorov states that a NN can learn any continuous function, it also says that a network with a large number of inputs, being trained on a small number of vectors, must be asked to approximate a target function which is simpler, in order to learn accurately from the given data. We specifically pointed out in the Static Security Assessment case study, that one must:

1. apply input preprocessing in order to reduce the number of (correlated) inputs
2. test if more data presented to the NN for learning actually increase the information content

Another comment to be made also refers to the non-linear modeling ability of the Back Propagation Neural Netowork. Care must be taken to insure compatibility between the preprocessing applied to data and the assumptions this preprocessing implies on the modeling the NN will perform. In the Oil Leak Detector case study, we argued for differentiating the data applied to all the inputs and output of the NN, for the learning process. We also pointed out that, in order to eliminate errors which would accumulate if we were to integrate the output of the NN during recall, we were going to use the same NN trained on differentiated data, with non-differentiated data, during recall. We pointed out that this use of the NN implies that it performs linear modeling. Because of this fact, we have forced the NN to use a linear transfer function in the hidden node, and not allow the default sigmoid. Of course, if we did not do this, the NN would have used its non-linear learning ability and would have reduced the error during learning. The network would have simply learned the noise present in the learning set and would have had a lower performance during recall, than would a linear model.

While, given enough data, a Back Propagation Neural Network can approximate any arbitrary function, we have found that, in practical cases, learning data are sparse. The only way to alleviate this problem is by reducing the number of weights the network must

compute. A reduction in the number of weights implies a reduction in the number of nodes. The implication is that the user must, as much as possible, pre-process the data and extract the features the Neural Network needs. Failing to do that, the Neural Network will need to do the feature extraction on its own.

Using expert knowledge is somewhat counter to the hope of neural network users, that the neural network will extract what it needs to solve the problem with no outside help. Reality shows that one must provide either expert knowledge, or a large amount of learning data.

A comment must be made regarding unsupervised learning neural networks. The Static Security Assessment case study has presented an unsupervised algorithm for grouping similar power system contingencies. Care was taken to ensure that the system will group contingencies because they have a similar effect on the power system, and not, for example, because they happen to coincide with some other unrelated feature that occurred on the power system. In other words, the reason for clustering was built into our algorithm. The user should be cautious when using a standard unsupervised network, for example a Kohonen network. It is true that the network will extract features from the presented vectors, however it will extract the features it wants, and, perhaps not the features needed for the application at hand. The user must ensure that the network's structure and learning process match the problem to be solved.

One of the case studies has shown that a Fuzzy Logic module can be used to synthesize a probability density function. The fuzzy sets and fuzzy rules were quite intuitive, and the controller behaved as expected. We have found in the literature, numerous papers showing fuzzy inference engines performing as function approximators, and being trained using a process similar to the Neural Network supervised learning. One of the accepted facts regarding neural networks, is that, in general, one cannot explain the values, a feed forward neural network, taught with back propagation, has assigned to its weights. The weight values do not correspond to parameter values that belong to the physical system. One of the appeals of Fuzzy Logic, is that it uses expert knowledge in

its fuzzy sets and rules. We feel that one should not use an automated algorithm to determine fuzzy sets and rules that have no physical meaning, just to produce yet another universal approximator.

5.2 Conclusions and Further Research

This work has showed a variety of techniques that lend themselves well to the Fault Detection System framework. A number of issues merit further research.

A large body of work was developed detailing the steps needed for the estimation of linear transfer functions coupled with stochastic disturbance models. Both the transfer function and the disturbance model are estimated simultaneously. The purpose of the disturbance model is to account for noise and result in a more accurate transfer function parameter estimates.

Neural Networks compute their weights during learning without the benefit of a disturbance model. We feel that further research is needed to justify this practice.

As we have pointed out in one of our case studies, the adaptive Bayes classifier we have developed, had the property that data with less noise has resulted in a behaviour which was more abrupt when the classification decision had to be made. Without accounting for this effect would have resulted in an algorithm which would have become singular if noise free data ware used. Our solution to "de-tune" the classifier by "adding" noise in the portion that was responsible for computing the decision threshold, is not elegant. More work needs to be spent in this area.

Finally, we have pointed out that using the innovation of a Kalman filter to point out the occurrence of a fault, as is sometimes indicated in the literature, is very dangerous and that a fast Kalman filter might even adapt to the fault, in effect masking it. We also showed that the integral of the innovation is a suitable signal for fault detection. Next we

raised the point of the integral integrating the innovation value along with noise, noise which will result in an unbounded variance. To solve this problem, we proposed to window the integral, thus limiting the amount of noise present. We found that this is the same approach performed by Sohlberg in [53]. A rectangular window is used. We feel that further work is required for determining the characteristics of the window for optimum failure detection. Some of these issues are discussed in a paper titled "Combining Neural Networks, Fuzzy Logic, and Kalman Filtering in an Oil Leak Detector For Underground Electric Power Cables" recently submitted to the IEEE Transactions on Power Systems.

5.3 Contributions

A number of papers have been published describing some of the developments of this work [25], [67], [85], and are covered in more detail in Chapter 1. The following is a summary of contributions made:

1. A clustering technique using the Partial Least Squares (PLS) approach is used to group contingencies in an Electric Power Systems. This is the first known usage of PLS in an electrical application. Previous applications of PLS have been in Chemical Engineering for process control. The clustering technique is applicable to any modeling problem where rather than using one input-output model for the whole input set, a number of models are constructed for parts of the input set.
2. A Neural Network based model capable of estimating the future state of a power system following a contingency is developed. While such a network has previously been reported in the literature, data preprocessing (PCA, PLS) results in a greatly reduced required learning data set and a faster learning, also improving generalization.

3. A number of new architectures for an Oil Leak Detection System in Underground Electric Power Cables are proposed. The proposed architectures have several advantages over other possible approaches.
4. For the first time, an Artificial Intelligence (AI) based system is developed for the oil leak detection application. The results from tests using real field data and the lessons learned from this implementation are presented. AI is used to perform feature extraction, modeling, as well as soft voting.
5. Custom data preprocessing leads to the possibility of a linear model to handle a strongly non-linear task. This leads to a vastly shortened learning time.
6. While a classical Neural Network supervised learning method is used to build a model in the Oil Leak Detector test case, no learning data set (a teacher) is available. A new approach shows how such a teacher can be constructed while learning takes place. The method is iterative and builds the model while constantly adjusting the training data set. It is similar to a student learning from a teacher, who then uses the student to fill in knowledge gaps, advancing the teacher's knowledge. This method provides, as a benefit, an elegant, generic, process for handling missing or corrupted data.
7. A new approach to the Electric Power Generator Stator Blockage Detection problem illustrates a way to exploit high input data correlations in order to achieve:
    - high resilience to hardware breakdowns
    - an almost linear input / output relationship resulting in fast learning

8. All Fault Detection Systems presented in this thesis use a Bayes classifier responsible for deciding whether or not an alarm should be initiated. A new, on-line, adaptive, extension to the classical Bayes classifier is presented.
9. All Fault Detection Systems presented in this thesis have to deal with the lack of data showing a system fault. They must learn using data produced by correctly operating systems but recognize when inconsistent data, produced by a fault, is presented. Two new Fuzzy Logic based implementations show how statistics of failed systems data can be generated using expert knowledge.

REFERENCES

[1] Ron J. Patton and J. Chen, "Robustness in Quantitative Model-Based Fault Diagnosis", *IEE Colloquium on Intelligent Fault Diagnosis - Part 2: Model-Based Techniques*, pp. 4/1-417, 26 Feb 1992

[2] Ron J. Patton, "Fault Detection and Diagnosis in Aerospace Systems Using Analytical Redundancy", *Computing & Control Engineering Journal*, vol. 2, no.3, pp. 127-136, May 1991

[3] Ron J. Patton, P. M. Frank, and R. N. Clark, *Fault Diagnosis in Dynamic Systems, Theory and Application*, Prentice Hall (Control Engineering Series), ISBN 0133082636, 1989

[4] Ron J. Patton and S. M. Kangethe, "Robust Fault Diagnosis Using the Model-Based Approach", *IEE Colloquium on Condition Monitoring and Failure Diagnosis- Part 1*, pp. 2/1-213, 2 Nov 1988

[5] Janos J. Gertler, "Survey of Model-Based Failure Detection and Isolation in Complex Plants", *IEEE Control Systems Magazine*, vol. 8, no. 6, pp. 3-11, Dec. 1988

[6] Ron J. Patton and J. Chen, "Advances in Fault Diagnosis Using Analytical Redundancy", *IEE Colloquium on Plant Optimisation for Profit (Integrated Operations Management and Control), (Digest No.1993/019)*, pp. 6/1 –612, 28 Jan 1993

[7] Marios M. Polycarpou and Arthur J. Helmicki, "Automated Fault Detection and Accommodation: A Learning System Approach; Systems", *IEEE Transactions on Man and Cybernetics*, vol. 25, no. 11, pp. 1447-1458, Nov. 1995

[8] Marios M. Polycarpou and Arun T. Vemuri, "Learning Methodology for Failure Detection and Accommodation", *IEEE Control Systems Magazine*, vol. 15, no. 3, pp. 16-24, June 1995

[9] Marios M. Polycarpou and Arun T. Vemuri, "Learning Approach to Fault Tolerant Control: An Overview", *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS)*, pp. 157-162, 14-17 Sept. 1998

[10] Marios M. Polycarpou and Alexander B. Trunov, "Learning Approach to Nonlinear Fault Diagnosis: Detectability Analysis", *IEEE Transactions on Automatic Control*, vol. 45, no. 4, pp. 806-812, April 2000

[11] Abbas Emami-Naeini, Muhamad M. Akhter, and Stephen M. Rock, "Effect of Model Uncertainty on Failure Detection: The Threshold Selector", *IEEE Transactions on Automatic Control*, vol. 33, no.12, pp. 1106-1115, Dec. 1988

[12] Rami Mangoubi, Brent Appleby, and Jay Farrell, "Robust Estimation in Fault Detection", *Proceedings of the 31st IEEE Conference on Decision and Control*, vol. 2, pp. 2317–2322, 16-18 Dec.1992

[13] Raman Mehra, Constantino Rago, and Sanjeev Seereeram, "Autonomous Failure Detection, Identification and Fault-tolerant Estimation with Aerospace Applications", *Proceedings of the 1998 IEEE Aerospace Conference*, vol. 2, pp. 133-138, 21-28 March 1998

[14] Deog Y. Oh and Hee C. No, "Instrument Failure Detection and Estimation Methodology for the Nuclear Power Plant", *IEEE Transactions on Nuclear Science*, vol. 37, no. 1, pp. 21-30, Feb. 1990

[15] P.D. Hanlon and P.S. Maybeck, "Equivalent Kalman filter bank structure for multiple model adaptive estimation (MMAE) and generalized likelihood ratio (GLR) failure detection"; *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 5, pp. 4312-4317, 10-12 Dec. 1997

[16] Marcello R. Napolitano, Dale A. Windon, Jose L. Casanova, Mario Innocenti, and Giovanni Silvestri, "Kalman Filters and Neural-Network Schemes for Sensor Validation in Flight Control Systems", *IEEE Transactions on Control Systems Technology*, vol. 6, no. 5, pp. 596 –611, Sept. 1998

[17] M. H. Petrick and B. Wigdorowitz, "A Priori Model Structure Selection for System Identification", *Control Eng. Practice*, vol. 5, pp. 1053-1062, 1997

[18] Prahba Kundur, *Power System Stability and Control*, McGraw-Hill, ISBN 0-07-035958-X, 1993

[19] Sami Repo, "On-Line Voltage Stability Assessment of Power System – An Approach of Black-box Modelling", Tampere University of Technology, Publications 344, September 2001

[20] J. Peter Lark, Chair, Robert B. Nelson, Commissioner, and Laura Chappelle, Commissioner, "Michigan Public Service Commission Report on August 14th Blackout", November 2003

[21] Helmut Muler, Robert Rehbold, and Horst Emshoff, "A Fuzzy Logic Expert System for Detecting Generator H_2 Leaks", *Third International Conference on Industrial Fuzzy Control and Intelligent Systems*, IFIS '93, pp. 185-190, 1-3 Dec. 1993

[22] H.W. Emshoff and M. Krug, "Water-Cooled Stator Windings of Turbogenerators. Computerized Monitoring of Stator Bar Water Temperature Rise for Early Detection of Overheating Due to Flow Restrictions in the Stator Winding", Kraftwerk Union Aktiengesellshaft, Federal Republic of Germany, *Cigre session paper. Ref. No: 11-09*, 1986

[23] M. Moliere, Y. Verdier, and C. Leymonie, "Oxidation of Copper in High Purity Water at 70 °C: Application to Electric Generator Operation", GEC Alshom, Centre d'Essais et de Recherches sur les Materiaux, unpublished, 1988

[24] M. Rioual, "A Thermohydraulic Modeling for the Stator Bars of Large Turbogenerators: Development, Validation by Laboratory and On-Site Tests", *IEEE Transactions on Energy Conversion*, vol. 12, no. 1, pp. 1-9, March 1997

[25] Lennart Ljung, *System Identification. Theory for the User*, Prentice Hall Information and System Sciences Series, ISBN 0-13-656695-2, 1999

[26] Paul Taylor, *Process Identification Notes: course notes*, McMaster University, 1993

[27] G.E.P. Box, G.M. Jenkins, and Holden-Day, *Time Series Analysis Forecasting and Control,* ISBN: 0-13-060774-6, 1976
[28] Lennart Ljung and Yorkel Glad, *Modeling of Dynamic Systems*, Prentice Hall Information and System Sciences Series, ISBN 0-13-597097-0, 1994
[29] Jonathan D. Cryer, *Time Series Analysis*, PWS Publishers, ISBN 0-87150-963-6, 1986
[30] O.D. Anderson, J.K. Ord, and E.A. Robinson, *Time Series Analysis: Theory and Practice*, ISBN 0444876839
[31] R. Christensen, *Linear Models for Multivariate, Time Series, and Spatial Data*, ISBN 0-387-97413-X, 1990
[32] Signal and Image Processing with Neural Networks; ISBN 0-471-04963-8, 1994
[33] Timothy Masters and Christopher M. Bishop, *Neural Networks for Pattern Recognition*, ISBN 019 853864, 1995
[34] Simon Haykin, *Neural Networks A Comprehensive Foundation*, ISBN 0-02-352761-7, 1994
[35] D.E. Rumelhard, J.L. McClelland, and the PDP Research Group, *Parallel Distributed Processing*, ISBN 0-262-18120-7, 1988
[36] T. Kohonen, *Self-Organizing Maps*, ISBN 0720-678X, 1997
[37] Geoffrey Hinton, *Neural Networks and Related Learning Algorithms*, seminar notes, unpublished, Department of Computer Science, University of Toronto, 1998
[38] S. Chen, C.F.N. Cowan, and P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, March 1991
[39] L. Zadeh, "Fuzzy Sets", *Information and Control*, Vol. 8, pp. 338-353, 1965
[40] E. Cox, *The Fuzzy Systems Handbook*, Second Edition; AP Professional, ISBN: 0121944557, 1999
[41] V. Novak, *Fuzzy Sets and Their Applications*, ISBN 0852745834, January1989
[42] H.J. Zimmermann, *Fuzzy Set Theory and Its Applications*, Kluwer Academic Publishes, ISBN: 0792374355, 1991
[43] Lotfi A. Zadeh, *Fuzzy Sets, Fuzzy Logic, and Fuzzy System: Selected Papers by Lotfi A. Zadeh*, World Scientific Pub Co., ISBN: 9810224214, 1996
[44] H.T. Nguyen and A. Walker, *First Course in Fuzzy Logic*, ISBN: 0849316596, 1999
[45] K.M. Passino and S. Yurkovich, *Fuzzy Control*, Addison-Wesley, ISBN: 020118074X, 1998
[46] B. Kosko, *Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence; Prentice Hall*, ASIN: 0136114350, 1992
[47] B.W. Leach, "An Introduction to Kalman Filtering", National Research Council Canada, National Aeronautical Establishment, NAE MISC 57, March1984
[48] Greg Welch and Gary Bishop, "An Introduction to the Kalman Filter", Department of Compute Science, University of North Carolina at Chapel Hill, TR 95-041, 2003
[49] Julius T. Tou and Rafael C. Gonzales, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, ISBN 0-201-07586-5, 1974

[50] Richard O. Duda and Peter E. Hart, *Pattern Classification and Scene Analysis*, A Wiley-Interscience Publication, John Wiley & Sons, ISBN 0-471-22361-1, 1973

[51] Gudmund R. Iversen, *Bayesian Statistical Inference*, Sage Publications, 0-8039-2328-7, 1984

[52] J. Louis Tylee, "On-Line Failure Detection in Nuclear Power Plant Instrumentation", *IEEE Transactions on Automatic Control*, vol. 28, no. 3, pp. 406-415, Mar 1983

[53] B. Sohlberg; "Monitoring and Failure Diagnosis of a Steel Strip Process", *IEEE Transactions on Control Systems Technology*, vol. 6, no. 2, pp 294–303, March 1998

[54] Herve Abdi, "Partial Least Squares (PLS) Regression", The University of Texas at Dallas, http://www.utdallas.edu/~herve/Abdi-PLS-pretty.pdf

[55] P. Geladi and B. Kowalski, "Partial Least Square Regression: A Tutorial", *Analytica Chimia Acta*, pp. 1-17, 1986

[56] T. Naes and H.Martens, "Comparison of Prediction Methods for Multicolinear Data", *Communications in Statistics, Simulation and Computation*, pp: 545-576, 1985

[57] Barry M. Wise and Neal B. Gallagher, *PLS_Toolbox 2.1 User Manual*

[58] Harald Martens and Tormod Naes, *Multivariate Calibration*, ISBN 0 471 90979 3, 1989

[59] Barry Lennox, Gary Montague, and Ognjen Marjanovic, "Detection of Faults in Batch Processes: Application to an Industrial Fermentation and a Steel Making Process", Control Technology Centre Ltd, School of Engineering, University of Manchester, UK, Dept. of Chemical and Process Engineering, University of Newcastle-upon-Tyne, UK, http://www.forum-ira.com/pdf/4conf3.pdf

[60] B. Lennox, H.G. Hiden, G.A. Montague, G. Kornfeld, and P.R. Goulding, "Application of Multivariate Statistical Process Control to Batch Operations", Elsevier, *Computers and Chemical Engineering*, vol 24, pp. 291-296, 2000

[61] Randall D. Tobias, "An Introduction to Partial Least Squares Regression", SAS Institute Inc., Cary, NC, http://support.sas.com/rnd/app/papers/pls.pdf

[62] Josef Kittler and Pierre A. Devijver, *Pattern Recognition: A Statistical Approach*Prentice, Hall International, ISBN 0-13-654236-0, 1982

[63] Keinosuke Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, ISBN: 0122698517, 1972

[64] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood From Incomplete Data via EM Algorithm", *Journal of Royal Statistical Society Series B* vol. 39, pp.1-38, 1997

[65] Kamalaldin Morad, William Y. Svrcek, and Ian McKay, "Probability Density Estimation Using Incomplete Data", *ISA Transactions,* vol. 39, pp. 379-399, 2000

[66] B.W. Silverman, "Density Estimation for Statistics and Data Analysis", *Monographs on Statistics and Applied Probability*, Chapman & Hall, 1986

[67] D. Fischer, B. Szabados, and W.F.S. Poehlman, "Using a Bayes Classifier to Optimize Alarm Generation to Electric Power Generator Stator Overheating", *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no.3, pp. 703–709, June 2003

[68] L. Chen, K. Tomsovic, A. Bose, and R. Stuart; "Estimating Reactive Margin for Determining Transfer Limits", *IEEE Power Engineering Society Summer Meeting*, vol. 1, pp. 490-495, 16-20 July 2000

[69] M. La Scala, M. Trovato, and F. Torelli, "A Neural Network-Based Method for Voltage Security Monitoring", *IEEE Transactions on Power Systems*, vol. 11, no 3, pp. 1332 –1341, Aug. 1996

[70] V. Brandwajn, A. Kumar, A. Ipakchi, A. Bose and S.D. Kuo, "Severity Indices for Contingency Screening in Dynamic Security Assessment", *IEEE Transactions on Power Systems*, vol. 12, no.3, pp. 1136-1142, Aug. 1997

[71] Y. Mansour, E. Vaahedi and M.A. El-Sharkawi, "Large Scale Dynamic Security Screening and Ranking Using Neural Networks", *IEEE Transactions on Power Systems*, vol. 12, no. 2, pp. 954–960, May 1997

[72] Y. Mansour, E. Vaahedi and M.A. El-Sharkawi, "Dynamic Security Contingency Screening and Ranking Using Neural Networks", *IEEE Transactions on Neural Networks*, vol. 8, no. 4, pp. 942-950, July 1997

[73] K.L. Lo, L.J. Peng, J.F. Maqueen, A.O. Ekwue, and D.T.Y. Cheng, "Application of Kohonen Self-Organizing Neural Network to Static Security Assessment", *Forth International Conference on Artificial Neural Networks*, pp. 387-392, 1995

[74] M.A. El-Sharkawib and R. Atteri, "Static Security Assessment of Power System Using Kohonen Neural Network", *Proceedings of the Second International Forum on Applications of Neural Networks to Power Systems*, ANNPS '93, pp. 373-377, 19-22 April 1993

[75] M. El-Sharkawi and D. Niebur, "Security Assessment and Enhancement", *A Tutorial Course on Artificial Neural Networks with Applications to Power Systems*, IEEE Catalog Number: 96 TP 112-0, pp. 104-127, 1996

[76] K.S. Swarup and P.B. Corthis, "ANN Approach Assesses System Security", *IEEE Computer Applications in Power*, vol. 15, no. 3, pp. 32–38, July 2002

[77] K. Shanti Swarup and P. Britto Corthis, "ANN Assesses System Security", *IEEE Computer Applications in Power*, ISBN 0895-0156/02, 2002

[78] W.P. Luan, K.L. Lo and Y.X. Yu, "ANN-Based Pattern Recognition Technique for Power System Security Assessment", *Proceedings International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*, pp. 197-202, 4-7 April 2000

[79] K.L. Lo and L.J. Peng, "Design of Artificial Neural Networks for On-Line Static Security Assessment Problems", *Fourth International Conference on Advances in Power System Control, Operation and Management*, APSCOM-97. (Conf. Publ. No. 450), vol. 1, pp. 288-293, 11-14 Nov. 1997

[80] S. Weerasooriya and M.A. El-Sharkawi, "Feature Selection for Static Security Assessment Using Neural Networks", *IEEE International Symposium on Circuits and Systems, ISCAS '92. Proceedings*, vol. 4, pp. 1693-1696, 3-6 May 1992

[81] S. Weerasooriya and M.A. El-Sharkawi, "Use of Karhunen-Loe've Expansion in Training Neural Networks for Static Security Assessment", Proceedings of the First International Forum on Applications of Neural Networks to Power Systems, pp. 59-64, 23-26 July 1991

[82] B. Jeyasurya, "Artificial Neural Networks for On-Line Voltage Stability Assessment", *IEEE Power Engineering Society Summer Meeting*", vol. 4, pp. 2014-2018, 16-20 July 2000

[83] T.S. Sidhu and L. Cui, "Contingency Screening for Steady-State Security Analysis by Using FFT and Artificial Neural Networks", *IEEE Transactions on Power Systems,* vol. 15, no. 1, pp. 421-426, Feb. 2000

[84] C.A. Jensen, M.A. El-Sharkawi and R.J. Marks, "Power System Security Assessment Using Neural Networks: Feature Selection Using Fisher Discrimination", *IEEE Transactions on Power Systems*, vol. 16, no. 4, pp. 757–763, Nov. 2001

[85] D. Fischer, B. Szabados, and W.F.S. PoehIman, "Automatic Contingency Grouping Using Partial Least Squares and Feed Forward Neural Network Technologies Applied to the Static Security Assessment Problem", Large Engineering Systems Conference on Power Engineering, pp. 84–89, May 7-9, 2003

[86] T.W. Anderson, *An Introduction to Multivariable Statistical Analysis*, John Wiley & Sons, sec. edition, ISBN: 0-471-88987-3, 1984

[87] "Underground Transmission Systems Reference Book", Electric Power Research Institute (EPRI), 1992

[88] "Leak Detectives: Brookhaven", *Science Daily press release*, Con Edison, EPRI Report New Method for Finding Underground Pipe Leaks, 1999

[89] "Underground Pipeline Leak Detection and Location Technology Application Guide", User Guide UG-2028-ENV, Naval Facilities Engineering Service Center, 1998

[90] J. Grzan, E.I. Hahn, R.V. Casalaina, and J.O.C. Kansog, "The 345 KV Underground/Underwater Long Island Sound Cable Project", *IEEE Transactions on Power Delivery*, vol. 8, no. 3, pp. 750-760, July 1993

[91] M. Rioual, "Presentation of a System for the Improvement of the On-Line Thermal Monitoring on 900 MW Turbogenerators for Predictive Maintenance Purposes", *IEEE Transactions on Energy Conversion*, vol. 12, no. 2, pp. 157-165, June 1997

[92] V. Poljakov and V. Tsvetkov, "Methods of Stator Winding On-Line Diagnostics for Large Turbine Generator Preventive Maintenance", *IEEE Transactions on Energy Conversion*, vol. 14, no. 4, pp. 1646–1650, Dec. 1999

[93] Li Junquing and Li Heming, "A Thermal Model for the Water-Cooled Stator Bars of the Synchronous Generator", *International Conference on Power System Technology*, PowerCon 2002, vol. 2, pp. 756 -760, 13-17 Oct. 2002

[94] D. Fischer, B. Szabados, and W.F.S. Poehlman, "Combining Partial Least Squares and Feed Forward Neural Network Technologies in a Fault Detection System with Large Number of Correlated Sensors", *Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference*, IMTC/2002, vol.1, pp. 829-834, 21-23 May 2002

[95] S. Simani, C. Fantuzzi, and S. Beghelli, "Diagnosis Techniques for Sensor Faults of Industrial Processes", *IEEE Transactions on Control Systems Technology*, vol. 8, no. 5, pp. 848-855, Sept. 2000

[96] S. deJong and H. Fiers, "Principal Covariance Regression", *Chemometrics Intelligent Laboratory Systems*, vol. 14, pp. 155-164, 1992

[97] K.Karhunen, "Über Lineare Methoden in der Wahrscheinlichkeitsrechnung", *Annales Academiae Scientiarum Fennicae, Seried A1: Mathematica-Physica*, 37:3-79, 1947

[98] T.W. Anderson, *An Introduction to Multivariable Statistical Analysis*; John Wiley & Sons, 2nd edition, ISBN: 0-471-88987-3, 1984

[99] A.N. Kolmogorov and Dokl. Akad. Nauk, "On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition", USSR, vol. 114, pp. 953-56, 1957