Aspects of total system fault detection and diagnosis using neural networks applied to the Pebble Bed Modular Reactor

L. S. Madlopha B. Eng. Electronic Engineering

Dissertation submitted in partial fulfilment of the requirements for the degree Magister in Electronic and Computer Engineering at the North-west University (Potchefstroom Campus)

Supervisor: **Prof. C. P. Bodenstein**

November 2005

Potchefstroom

Acknowledgement

First of all, I would like to thank God for giving me strength and perseverance needed to complete this project. I would also like to thank my family especially my mom for standing by me through good and bad times. This project, besides my laboratory and computer work, owes much to the effort of many, that either helped me, pushed me in one way or another. Thanks to all of them.

In addition, I would extend my thanks to my project leader Prof C.P Bodenstein for his continued support. His guidance and support helped keep me pointed in the right direction and always moving forward.

Lastly my thanks go out to my friends. With these friends, I knew I could accomplish anything. The friends I've made will stay with me, reminding me where ever I go.

Abstract

The objective with this thesis is to investigate the potential of model-based diagnosis, especially when combined with neural networks as modelling tool. The diagnosis system has been applied to a model of the Pebble Bed Micro Model. The neural network was mainly used as tool to simulate the normal behaviour of the plant.

The discrepancy between the two models (actual model and neural network) which becomes larger when a fault is present is used to form residuals. The generation of residuals needs to be followed by residual evaluation, in order to arrive at detection and isolation decisions.

This thesis considers the design of fault detection and diagnosis for linear and nonlinear systems. It consists of different sections. Firstly, an overview of the ideas and theory behind the model-based approach of fault detection and diagnosis is given. Initially, a fourth-order linear system is simulated and a number of faults are simulated, detected and diagnosed. The knowledge gained with the first system is then refined and applied to a nonlinear water level control system which is used as a benchmark. The calculations and application results are presented in detail to illustrate the principles.

The principles are then applied to simulation as well as experimental results on the Pebble Bed Micro Model. Flownex simulation software was used to generate the data, where experimental data was not practical or safe to obtain.

Typical faults that were diagnosed are plant and instrumentation faults. Since the full-scale Pebble Bed Modular Reactor plant is not yet in operation, the principles applied in this thesis can be used to design and implement fault detection and diagnosis on a real system.

Table of contents

ACKNOWLEDGEMENT i

ABSTRACT ii

LIST OF FIGURES vi

LIST OF TABLES viii

LIST OF ABBREVIATIONS ix

1 INTRODUNCTION 1

1.1 BACKGROUND 2

1.2 PROBLEM STATEMENT 3

1.3 PROPOSED SOLUTION 5

1.4 OBJECTIVES 6

1.5 RESEARCH METHODOLOGY 7

1.6 OVERVIEW OF RESEARCH 8

2 FAULTS DIAGNOSIS METHODS 10

2.1 OVERVIEW 11

2.2 FAULTS MODEL 12

2.2.1 Multiplicative changes in parameters 12

2.2.2 Additive changes in parameters 13

2.2.3 Combined additive and multiplicative changes in parameters 13

2.3 CLASSIFICATION OF DIAGNOSIS ALGORITHMS. 13

2.4 MODEL BASED DIAGNOSIS. 14

2.5 MODELLING FAULTS BY MEANS OF RESIDUAL GENERATION. 16

2.6 RESIDUALS EVALUATION 18

2.7 THRESHOLD DEFINITION 20

2.8 NEURAL NETWORK. 21
2.8.1 Introdunction 21
2.8.2 Introductory theory of Neural networks 23
2.8.3 Analysing the Problem 24
2.8.4 Training of neural networks 25
2.8.5 MLP 27
2.9 SUMMARY 28

3 MODEL BASED FID 29

3.1 OVERVIEW 30
3.2 MODEL CONSTRUCTION 30
3.3 CREATING A NEURAL NETWORK FOR GENERATING RESIDUAL ... 32
3.4 IMPLEMENTING OF DIAGNOSIS SYSTEM 34
3.5 SIMULATION RESULTS. 40
3.6 A CASCADED PLANT MODELLED WITH SISO NETWORKS 45
3.7 SUMMARY 48

4 FAULT DIAGNOSIS ON A DRUM LEVEL SYSTEM 49

4.1 GOAL 50
4.2 BACKGROUND 50
4.3 METHODOLOGY 54
4.3.1 Simulink model 54
4.3.2 Simulated results 58
4.4 FAULT DIAGNOSIS. 63
4.4.1 Introdunction 63
4.4.2 Faults description 65
4.4.3 Faults modelled as arbitrary fault signals 66
4.4.4 Faults isolation 68
4.5 SUMMARY 70

5 FLOWNEX MODEL OF PBMM 72

5.1 BACKGROUND 73

5.2 PBMR POWER CONVERSION CYCLE 74

5.3 PBMM 76

5.4 MODELLING PBMM USING FLOWNEX 77

5.5 MODELLING LPC AND HPC. 79

5.6 MODELLING IC AND PC. 82

5.7 SIMULATION AND FAULTS DIAGNOSIS. 82

5.7.1 NN Model 84

5.7.2 Training the chosen Neural network 85

5.7.3 Fault identification and diagnosis 88

5.8 SUMMARY 92

6 CONCLUSION 93

6.1 SUMMARY OF EXPERIMENTAL RESULTS 94

6.2 CONTRIBUTIONS OF THE STUDY 94

6.3 AREAS FOR IMPROVEMENT AND FUTURE WORK 95

7 APPENDIX 96

LIST OF REFERENCES 104

List of Figures

Figure 2.3.1: Diagnosis family tree....14
Figure 2.4.1: General scheme of process model based FDD....16
Figure 2.5.1: Model to generate residual....18
Figure 2.6.1: Decision logic....19
Figure 2.6.1: Model to diagnose faults....21
Figure 2.8.1: Supervised learning....26
Figure 2.8.2: Unsupervised learning....26
Figure 3.2.1: Plant model of four cascaded first order sections....31
Figure 3.4.1: Process of constructing diagnosis system....34
Figure 3.5.1: Normalised plant response compared to NN....41
Figure 3.5.2: Setting threshold for residual....43
Figure 3.5.3: Impact of threshold on residual....44
Figure 3.5.4: Faults detection using Minimum and Maximum threshold....45
Figure 3.6.1: Four cascaded first-order sections....46
Figure 3.6.2: Detection of Plant fault....46
Figure 4.2.1: Feed water PID regulator....51
Figure 4.2.2: Closed loop model for feed water PID regulator....52
Figure 4.3.1.1: Simulink model of water level PID regulator....55
Figure 4.3.1.2: Simulink model of PID....55
Figure 4.3.1.3: Simulink model of controller in detail....56
Figure 4.3.1.4: Simulink model of modulator....56
Figure 4.3.1.5: Simulink model of relay histeresis....57
Figure 4.3.1.6: Simulink model of actuator....57
Figure 4.3.1.7: Simulink model of valve....57
Figure 4.3.1.8: Simulink model of water level....58
Figure 4.3.2.2.1: Memorisation of NN....61
Figure 4.3.2.2.2: Actual response compared to NN....62
Figure 4.3.2.2.3: Residuals of actual and NN response....63
Figure 4.4.3.1: Residuals properties of Figure 4.3.2.2.3....67

Figure 4.4.4.1: Detecting faults using fixed threshold 69
Figure 5.1.1: PBMM plant 74
Figure 5.2.1: Proposed PBMR schematic layout... 74
Figure 5.3.1: Schematic layout of the PBMM recuperative Brayton cycle 76
Figure 5.3.2: Schematic layout of the PBMM with the location of simulated faults.....77
Figure 5.4.1: Flownex Simulink interface 78
Figure 5.4.2: Valve opening during injection 79
Figure 5.5.1: Part of Flownex model schematically layout 80
Figure 5.5.2: Flownex response of pressure variation during injection 81
Figure 5.5.3: Experimental response of pressure variation during injection 81
Figure 5.6.1: Flownex response of pressure variation during injection 82
Figure 5.7.2.1: Modelling capabilities of NN with both input and hidden layer delay ...87
Figure 5.7.2.2: Modelling capabilities of NN with both input delay 88
Figure 5.7.3.1: Impact of adaptive threshold 90
Figure 5.7.3.2: Detection of faults using adaptive threshold.....90
Figure 5.7.3.3: Setting of threshold for ddetection of plant fault.....91
Figure 5.7.3.4: Detection of plant fault using adaptive threshold.....92
Figure A1: Schematic illustration of water level control system with faults.....97
Figure A2: Schematic layout of PBMM plant with sensors for collection of data.....98
Figure A3: Zoomed schematic layout of PBMM plant 99
Figure A4: Part of Flownex Model showing the collection of input and output data....100
Figure A5: Part of Flownex Model showing the collection of input and output data....101

List of Tables

Table 3.4.1: Maximum and minimum threshold values.....37
Table 3.4.2: Faults and their notations.....38
Table 3.4.3: Decision structure plant1.....39
Table 3.4.4: Decision structure plant2.....39
Table 3.4.5: Decision structure plant3.....39
Table 3.4.6: Decision structure plant4.....40
Table 3.6.1: Decision logic plant1.....46
Table 3.6.2: Decision logic plant2.....46
Table 3.5.3: Decision logic plant3.....47
Table 3.6.4: Decision logic plant4.....47
Table 4.2.1: Symbols definition.....53
Table 4.4.4.1: Strongly isolability of residuals.....69
Table 5.7.2.1: Comparisons of neural network training functions.....86

List of Abbreviations

ANN	Artificial Neural Network
FDD	Fault Detection and Diagnosis
FDDE	Fault Detection, Diagnosis and Evaluation
FID	Fault Identification and Diagnosis
HPC	High Pressure Compressor
IC	Inter-Cooler
LPC	Low Pressure Compressor
MSE	Mean Squared Error
MIMO	Multi-Input Multi-Output
MLP	Multi-Layer Perceptron
NN	Neural Network
PBMM	Pebble Bed Micro Model
PBMR	Pebble Bed Modular Reactor
PC	Pre-Cooler
PID	Proportional Integration Differentiation
PWR	Pressurised Water Reactor
SIMO	Single-Input Multi-Output
SISO	Single-Input Single-Output

1. INTRODUCTION

This chapter focuses on the background and motivation for the research. Then the problem statement with the proposed solution is discussed. The research problem is subsequently broken into sub-problems which are separately addressed. A brief description of the methodology applied to this research is presented. Finally an overview of the dissertation's chapters is given.

1.1 Background

A major challenge to product manufacturers today is how to economically manufacture high quality goods. The same applies to electricity generation. One important way to consistently achieve high-quality products is to utilise in-process machine monitoring and control. Equipment reliability and maintenance drastically affect the three key elements of competitiveness, namely quality, cost, and production lead time. With proactive maintenance, a company can shorten lead times by reducing the machine downtime in the case of discrete products or batch processes. Likewise, in the case of continuous production such as chemical processes or power plant, higher profitability is directly linked to plant availability.

Occurrence of faults or equipment failure is a major cause of sub-optimal plant operation. There is a growing realisation that maintenance of the equipment and the control loops in the face of faults is the key to achieving long-term economic success. An overall advisory system that has the ability to quickly detect abnormal plant operation and initiate remedial measures to bring the plant back to normal operating region is undoubtedly very useful in the context of overall plant optimisation and safety.

Early detection and diagnosis of process faults while the plant is still operating in a safe and controllable region can help avoid abnormal event progression to breakdown and so reduce productivity loss. If incipient faults are allowed to progress to full-fledged faults, damage may be incurred or life may be endangered. With software systems for detection and diagnosis of faults, it is possible to identify many minor faults before they significantly impact the performance of the system.

As early as the 1960's, it was realised that faults of critical systems, such as nuclear power plants, space exploration, and weapons systems can have grave consequences. Even a minor malfunction may cause the failure of the whole system resulting in loss of time, money and even life. Such considerations led to research into automated and (even on-line) *Fault Detection, Diagnosis and Evaluation* (FDDE) supervisory systems. The objectives of these critical system supervisors were to identify even relatively minor

malfunctions as early as possible so that they could be attended to before damage occurred or lives were endangered (Jia, 2002).

When equipment is referred to as faulty it is implied that some abnormality exists in the operating conditions. A more general definition of a fault is that there is a substantial degradation in system performance. This may be due to gradual or abrupt changes in the parameters of some system or process parameter or malfunction of equipment causing uncertainties in measured values.

To detect malfunction of a process a monitoring system is required. Such a monitoring system should, amongst other, have the following functions: Fault detection, fault diagnosis, fault location and fault correction.

Thus, human operators and automatic controllers need to be advised by intelligent supervision, control, and decision-support systems. These intelligent systems must have the ability to detect faults. They are proposed to serve as tools that may help improve the decision-making process of human operators or automatic controllers alike. Their basic task must be to prevent the (human or automated) decision makers from committing errors or from misjudging the current situation, by providing them with additional quantitative and qualitative information that can be used in the decision-making process, for detecting and discriminating faults at an as early time as possible, and for dealing with developing emergencies in an informed fashion (Jia, 2002).

1.2 Problem statement

Although good design practice tries to minimise the occurrence of faults and failures, recognition that such events do occur, enables system designers to develop strategies by which their effect is minimised.

In order to do fault detection and diagnostics on a plant, some means of detecting deviation from the usual normal operation of the plant is required. The model describing

normal behaviour can be seated in the experience of a plant operator, but this is severely limited by the attention span of a human in the presence of a large number of signals monitored in a modern plant. In many cases, certain plant signals, for instance pressure and temperature, have physical constraints that may not be exceeded to prevent destruction of the plant. In such cases, normal behaviour is considered operation in the safe region. By considering safety limits only, less damaging degradation is not detected. Subtle degradation could cause less economical operation. It could also point to incipient failure. Both these cases, if instantaneously detected and identified, can (hopefully) be rectified during planned maintenance. In order to detect subtle degradation it is necessary to have a model of normal operation so that relatively small deviations of plant behaviour from the norm may be detected. This model is more involved than the fixed limits for gross deviation to trigger alarms.

The model for normal behaviour from which small deviations of actual behaviour may be detected, can be done in a number of ways as will be outlined in chapter 2. For the purposes of this thesis a plant consists of combined electrical and mechanical systems that need to be modelled in some way to find a baseline for fault-free operation against which the physical plant will be compared to detect faulty behaviour. These methods are broadly categorised as process model-based and process history based.

In this thesis, process history will be used to train neural networks which will act as reference models in fault detection. In a plant which consists of a single section, selecting a neural net is rather straightforward. In a plant with many sections having multiple inputs and outputs, the question of model topology becomes important. Should one design a single multi-input multi-output neural net, or will it be better to partition the problem so that a number of single-input single-output neural nets can be used. This thesis will address some of the principles and issues to be considered in such choices.

The aim of this work is to investigate principles that can be used to design a system for fault detection and diagnosis for the proposed *Pebble Bed Modular Reactor* PBMR. Such a system will promote plant availability and help to increase safety. The aim of diagnosis

is the identification and isolation of faults. The implementation takes place in the following way: Detection of faults and malfunction using the deviation between measured values and calculated values from a model. The deviation or residual is analysed to find the probable location of the plant fault. The proposed diagnosis system is subdivided into two components, a part for fault detection and a part for fault isolation.

In this research four methods will be used to model the behaviour of the plant. Those methods are covered later in this report.

1.3 Proposed solution

In this report we will investigate model-based fault detection and diagnosis. A real system will be modelled with neural networks which have been trained from process history data. The difference between the real system and its model is used to generate residuals. Such residuals are the key elements to evaluate the occurrence of faults.

The neural networks will be used to simulate the normal behaviour of the plant after having been trained on the fault-free behaviour of the plant. The difference between the actual plant and neural network plant model will generate residuals. Once a fault has been introduced the response of the real system will differ to the one of the neural network, resulting in residuals which indicate probable faults. By means of statistical approaches the faulty residuals will be evaluated to form a vector matrix that will classify each occurred fault uniquely. Such classification will be used to identify and diagnose faults on the system.

A simple cascade of four first order transfer functions will be considered first. Once an understanding of the diagnosis system has been gained, it will be implemented on the benchmark model. The benchmark model will be used to show that the system will work on a real system. Finally the insight gained on the benchmark model will be applied to simulations of the PBMR which will be constructed in a few years as well as on a pilot plant.

1.4 Objectives

Two research objectives have been identified for this thesis. These objectives consist of finding the optimum method for fault detection together with increasing the accuracy of fault identification, and evaluating the efficiency of the diagnoses. The objectives are described below.

i. Determine the optimal method

The primary objective of this study is to identify those neural network configurations and preprocessing methods that yield the best results for a multiple-input multiple-output plant which is constructed in separable sections.

The effectiveness of the approach is based on the choice of neural network parameters, including the number of neurons in each layer, the learning rate, and the momentum. If there are too few neurons in the hidden layer, the network will not be trainable. If there are too many hidden neurons, the network won't be able to generalise and will perform well on data included in the training set, but not on other data.

The learning rate controls how quickly the network weights are adjusted. If the weights are adjusted too slowly, the network may train too slowly to be practical. If the weights are adjusted too quickly, the network may not converge to an acceptable error level. The choice of the neural network architecture will be based on which architecture will fit the data accurately.

Firstly a survey on different networks will be conducted. Based on this survey a choice will be made. The chosen network will be described in the next chapters of this report.

ii. Evaluate the accuracy of the diagnoses

The accuracy of the chosen method is evaluated with a number of experiments, progressing from relatively simple to more complex. The experiments will start with a linear plant, then a plant model followed by simulation models of the PBMM as well as data from physical experiments.

1.5 Research Methodology

This research focuses on neural networks to model the normal or fault-free operation of the plant. Plant history is obtained in a number of ways:

- Modelling of a linear system using Simulink
- Modelling of a benchmark plant using Simulink
- Physical modelling using Flownex
- Data from physical experiments

In order to compare various diagnostic approaches, it is useful to identify a set of desirable characteristics that a diagnostic system should possess. Then the different approaches may be evaluated against such a common set of requirements or standards.

Though these characteristics will not usually be met by any single diagnostic method, they are useful to benchmark various methods in terms of the a priori information that needs to be provided, reliability of solution, generality and efficiency in computation etc. In this context, one needs to understand the important concepts, completeness and resolution, before proceeding to the characteristics of a good diagnostic classifier. Whenever an abnormality occurs in a process, a general diagnostic classifier would come up with a set of hypotheses or faults that explains the abnormality.

Simulink and Flownex are tools which will be used to simulate the behaviour of plant. Neural networks will be used as a model to simulate the normal behaviour of the plant.

1.6 Overview of research

The dissertation will be divided up into the chapters described below and follow the sequence as presented:

Chapter 2: Fault detection and diagnosis methods

This chapter covers a theoretical background on fault detection and diagnosis. It briefly gives an overview on what has been done by other researchers. A short overview on model-based diagnosis is presented.

Chapter 3: Model-based fault detection and diagnosis

Some results from the literature investigation on model-based diagnosis are covered. Furthermore, guidelines on how to generate residuals using models are given in this chapter. The use of neural networks as models to simulate plant behaviour is given. The chapter is concluded with some results of model-based fault detection and diagnosis.

Chapter 4: Fault detection and diagnosis on a drum level control system

A description of a PID water regulator, used as a benchmark plant, is given. Subsequently fault detection and diagnosis on the model is done using Simulink as an actual process and a neural network as a reference model of fault-free operation. Since control may mask the effect of faulty behaviour (within limits), a system with control is investigated.

Chapter 5: Flownex model of the PBMM

A description of the Pebble Bed Modular Reactor and Micro Model will be given. The subsystems that are of interest for modelling are highlighted. The transient behaviour of these systems is derived using Flownex software.

Chapter 6: Conclusion

A summary of the research results is given. The contributions of the study as well as some areas for improvement are discussed. Some suggestions for future research are given.

2. FAULT DIAGNOSIS METHODS

This chapter introduces the theory and methods used in this thesis. The background and motivation of fault detection and diagnosis are presented. Some of the terminology used in the area of fault detection and diagnosis is described in order to simplify both the understanding and the reading. An overview of related work in the field of fault identification and diagnosis is given. The challenges that are faced when designing a fault identification and diagnosis system are discussed.

2.1 Overview

A fault model is a formal representation of the knowledge of possible faults and how they influence the process. More specific, the term *fault* means that component behaviour has deviated from its normal behaviour. It does not mean that the component has stopped working altogether. The situation where a component has stopped working is, in the diagnosis community, called a failure (Frick, 2001). So, one goal is to detect faults before they cause failure.

Faults may be modelled as deviations of normally constant parameters from their nominal values. These deviations can be modelled as multiplicative or additive, or a combination thereof. In the case of multiplicative faults, the value of a system parameter changes without an offset. In the case of an additive fault, an offset is introduced without changing the slope of the relation. Typical faults that are modelled in this way are gain-errors and bias errors in sensors. Process faults modelled as a deviation of physical parameters.

Other more elaborate faults may be modelled by time-varying or non-linear relations. In this thesis relatively simple fault models are considered. An advantage of using simple fault models is the simplicity and relatively few assumptions made in modelling. A disadvantage with such fault models is that fault isolability may be lost compared to more detailed fault models.

Another important factor is the choice of residuals as well as functions used for residual generation since residuals are fundamental components in a diagnosis system. A residual is a, often time-varying, signal that is used as a fault detector. Normally, the residual is designed to be zero (or small in a realistic case where the process is subject to noise and the model is uncertain) in the fault-free case and deviate significantly from zero when a fault occurs. Note, however, that other approaches exist. In case of a likelihood function based residual generator where the residual indicates how likely it is that the observed data is generated by a fault-free process, the residual is large in the fault-free case and

small in a faulty case. For the remainder of this text it is assumed, without loss of generality, that a residual is zero in the fault-free case.

2.2 Fault models

In a diagnosis system not only has the system to be modelled, but also the faults need to be modelled in order to be detected. A system fault model is a representation of possible faults and how they affect the system. If a novel or unmodelled fault occurs, the diagnosis system will not be able to give a correct diagnosis. It may not be possible to model all faults. Which ones to model require good system knowledge. There are several ways to model a fault, but the most common fault models will now be considered (Olsson, 2002).

2.2.1 Multiplicative changes in parameters

A fault can also be modelled as a deviation of a normally constant parameter, typically:

$$y_{obs}(t) = y_{corr}(t) \times (f(t)) \quad \text{..... 2.1}$$

where

$y_{obs}(t)$ = observed value

$y_{corr}(t)$ = correct value

$f(t)$ = fault signal, one in the fault free case.

Sensor faults are often modelled this way if they are of the type "gain errors". This fault model is also useful when the signal in the fault free case has a low and constant variance, i.e. the deviations from the mean value of the signal are small. When a fault is present the variance is still constant but higher, i.e. the deviations are bigger. There are also some faults that consist of a deviation of a physical parameter; these faults are also suited for this kind of fault model (Olsson, 2002).

2.2.2 Additive changes in parameters

A fault can be modelled as an additive signal, typically:

$$y_{obs}(t) = y_{corr}(t) + f(t) \quad \text{..... 2.2}$$

where

$y_{obs}(t)$ = observed value

$y_{corr}(t)$ = correct value

$f(t)$ = fault signal

This equation describes sensor faults of the type "offsets".

2.2.3 Combined additive and multiplicative changes in parameters

A combination of the previous cases may also be postulated.

2.3 Classification of diagnosis algorithms

In a dynamic system, any kind of malfunction that leads to an unacceptable anomaly in the overall system performance is defined as a fault. The first concern in the design of a fault detection algorithm is detection performance *i.e.*, the ability to detect and identify faults correctly with minimal delay and a minimum of false alarms.

In theory there are various types of techniques, which broadly fall into three categories (Clark, Patton and Frank, 1989), namely, (i) statistical approach, (ii) model-based approach and (iii) model-free approach. It is worth to mention that, irrespective of their implementation, all the techniques perform similar tasks that mainly involve three stages such as, detection, isolation and identification.

The most commonly used fault diagnosis approach is based on building a model of the real system to provide estimates of certain measured signals. Then, in the most usual case, the estimates of the measured signals are compared with the actual signals, that is,

the difference between the actual signal and its estimate is used to form the residual. The residual is later employed for fault identification and diagnosis.

However, numerous methods may be suitable for a given plant. In this study model-based diagnosis algorithm will be considered. Figure 2.3.1 below shows a diagnosis family tree where other methods are classified.

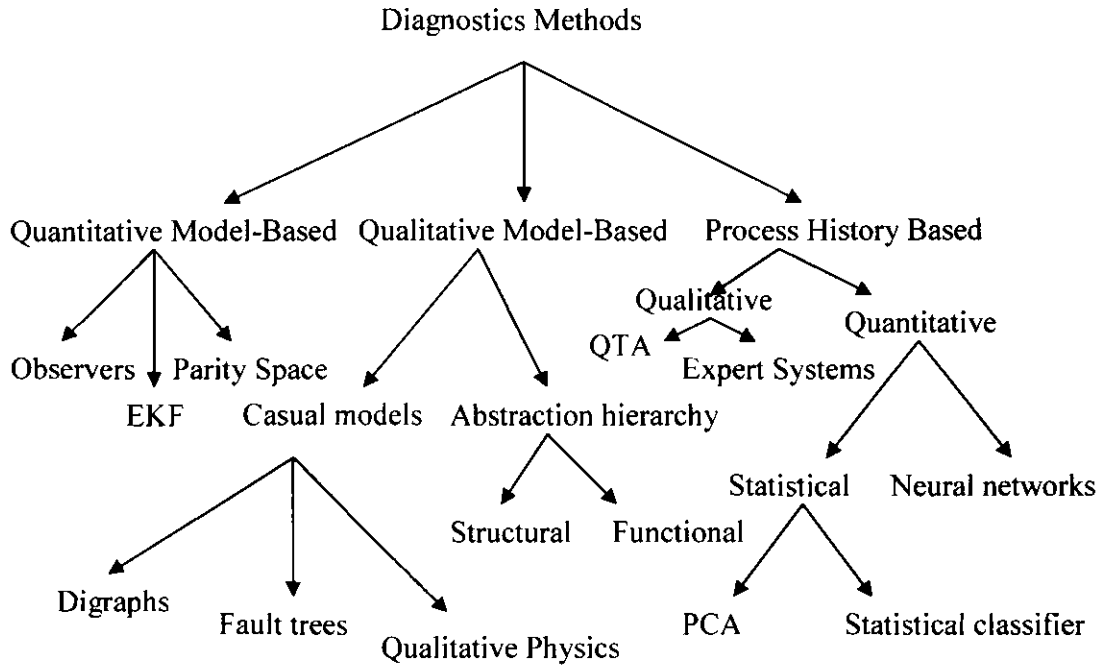

Figure2.3.1: Diagnosis family tree (Surya and Kavuri, 2002)

2.4 Model-based diagnosis

Why is there a need for a mathematical model to achieve diagnosis? It is easy to imagine a scheme where important entities of the dynamic process are measured and tested against predefined limits. The model-based approach instead performs consistency checks of the process against a model of the process (Isermann, 2004).

Methods that rely on a quantitative mathematical relation between the input and output are called model-based techniques. Model-based fault detection depends on the availability of a mathematical model of the plant.

This approach might be used on its own or as a complement to other methods. In model-based diagnosis a software model of the system is built and the system is compared with the model, see Figure 2.4.1. If the model is correct the system's output should be equal, or close to, the output from the model, given the same input. These values can then be compared and faults can be detected and in some cases also isolated and identified.

There are several important advantages with the model-based approach:-

- Outputs are compared to their expected values on the basis of process state, therefore the thresholds can be set much tighter and the probability to identify faults in an early stage is increased dramatically.
- A single fault in the process often propagates to several outputs and therefore causes more than one limit check to fire. This makes it hard to isolate faults without a mathematical model.
- With a mathematical model of the process the *Fault Identification and Diagnosis* (FID) scheme can be made insensitive to unmeasured disturbances.

There is of course a price to pay for these advantages in increased complexity in the diagnosis scheme and a need for a mathematical model. Different approaches for fault detection using mathematical models have been developed in the last 20 years (Willsky, 1976). The task consists of the detection of faults in the processes, actuators and sensors by using the dependencies between different measurable signals.

In general model-based algorithms are very different from one another in terms of how they generate residuals. In many cases the algorithms derive fault information from an optimal estimation of state variables (Isermann, 2004). Some other model-based methodologies rely on the construction of parity-space. Figure 2.4.1 illustrates the general structure of the model-based technique in the context of information processing.

The dependencies are expressed by mathematical process models. Based on measured input signals **U** and output signals **Y**, the detection methods generate residuals **r**, parameter estimates □ or state estimates $\hat{x}$ which are called features. By comparison

with the normal features, changes of features are detected, leading to analytical symptoms.

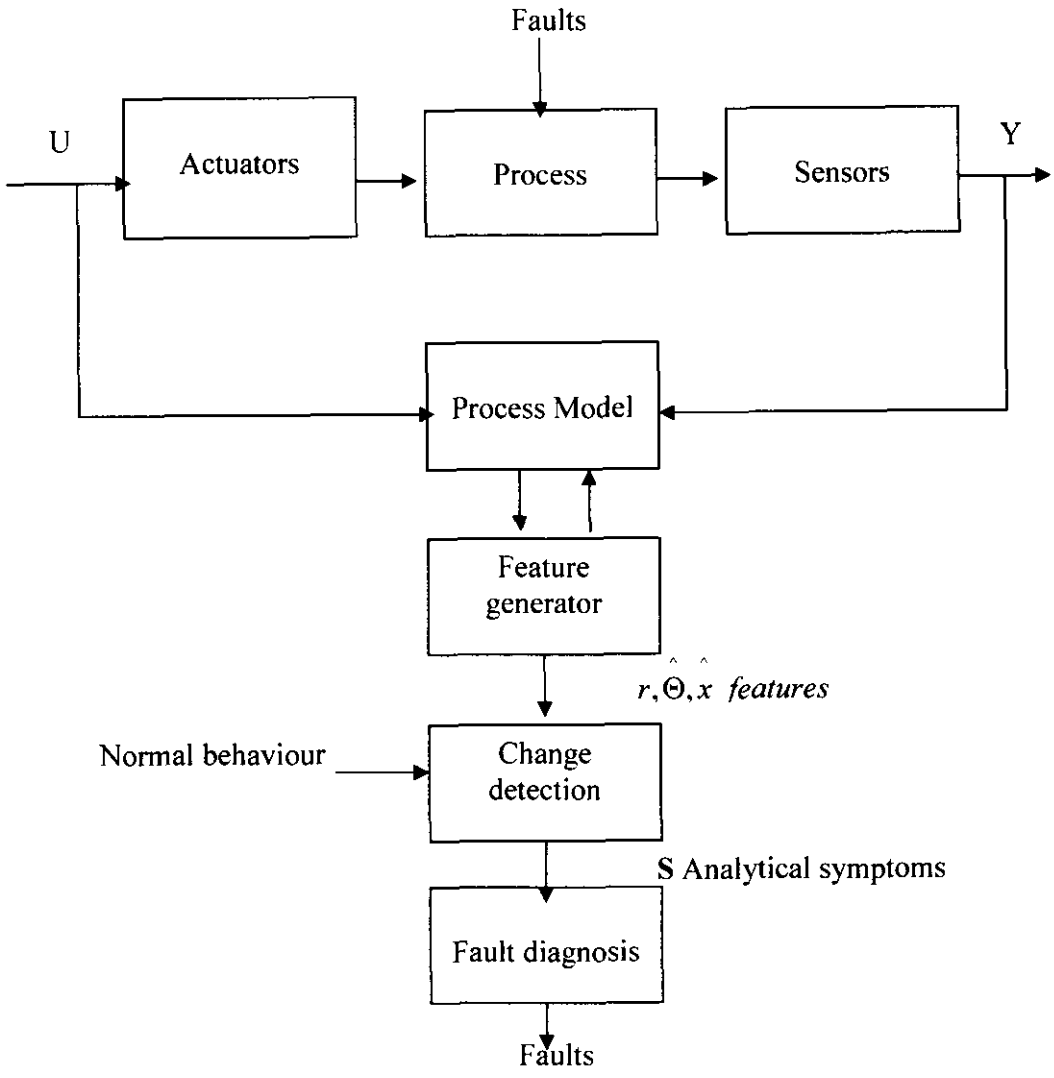

Figure 2.4.1: General scheme of process model-based fault detection and diagnosis (Isermann, 2004)

2.5 Modelling of faults by means of residual generation

These methods generally consist of two basic steps: Residual generation and a decision process to identify the cause. When faults occur, model and process differ and a residual $r \neq 0$ occurs, where broadly residuals represent the differences between various outputs and the expected values of these outputs.

The task of fault diagnosis is to, from the observations and a-priori knowledge, generate a diagnosis statement, i.e. to decide whether there is a fault or not and also to identify the fault. Thus the basic problems in the area of fault diagnosis is how the procedure for generating the diagnosis statement should look like, what parameters or behaviour that are relevant to study, and how to derive and represent the knowledge of what is expected or normal. This thesis focuses on principles of diagnosis that can be applied to the proposed PBMR plant. Typical faults to be considered are for example 'offset' and 'gain' faults in sensors, and physical faults in the plant. The observations are mainly signals obtained from the sensors, but can also be observations made by a human, such as level of noise and vibrations. The knowledge of what is expected or normal is derived from selected inputs together with models of the system.

To construct a model-based diagnosis system, a model of the system is needed as well as fault models which describe the effects of different faults. A fault model is the formal representation of the knowledge of possible faults and how they influence the process. In general, better models imply better diagnosis performance, e.g. smaller faults can be detected and more different types of faults can be isolated. In this section a general framework for fault modelling using residuals will be described. In this framework, practically all existing fault modelling techniques fit in naturally.

One of the ways in which faults can be detected is by using a plant and trained neural network to generate residuals as shown in Figure 2.5.1 below. A neural network (*ANN*) is created and trained to model the plant's fault-free behaviour. The residuals between plant and trained neural net are used to identify the presence or absence of a fault or faults. The residuals are then used to diagnose the faults.

Residual properties are firstly evaluated under normal conditions (with no faults). The reason is to determine threshold values that will be used to detect faults in the system. A fault is then introduced and again the properties of residuals are evaluated. There are a number of ways in which the generated residuals can be processed to diagnose faults. Some of these methods will be outlined in this report.

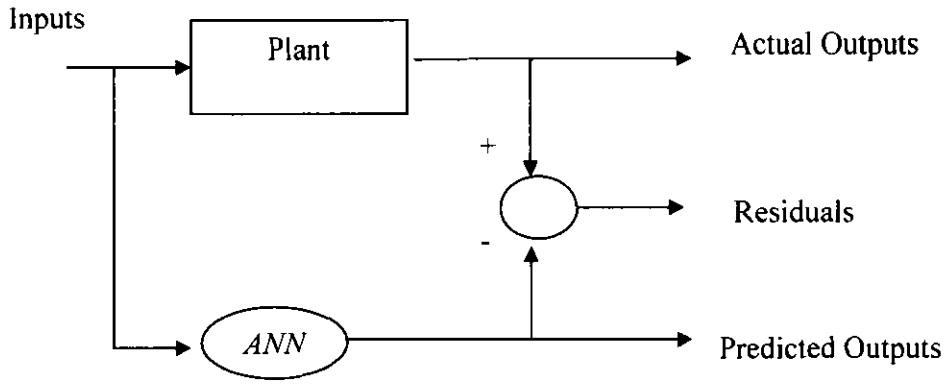

Figure 2.5.1: Model to generate residuals (Olsson, 2002)

2.6 Residual evaluation

With this approach, faults are modelled by signals $f(t)$. Central is the residual $r(t)$ which is a scalar or vector signal of which the elements are zero or small in the fault-free case when $f(t) = 0$, and is nonzero when a fault occurs, i.e. $f(t) \neq 0$.

Diagnosis can be considered as detecting and isolating faults in processes. The diagnosis system is then separated into two parts: residual generation and residual evaluation. This view of how to design diagnosis systems is well established on research conducted by (Karlsson, 2001). Thus (Karlsson, 2001) defines the model-based FID as a two-stage process: (1) residual generation, (2) decision making (including residual evaluation). This two-stage process is accepted as a standard procedure for model-based FID nowadays. Residual evaluation can be done using decision logic or a neural net, amongst others. These two methods will now be further discussed.

Residual evaluation by decision logic is an established procedure. The method is often called structured residuals and is primarily an isolation method (Karlsson, 2001). A diagnosis system using structured logic can be illustrated as in Figure 2.6.1. In this method, the first step of the residual evaluation is essentially to check if each residual is responding to the fault or not, often achieved via simple thresholding. By using residuals

that are sensitive to different subsets of faults, isolation can be achieved. What residuals that are sensitive to what faults is often illustrated with a residual structure. An example of a residual structure is shown in Figure 2.6.1 below.

	f_1	f_2	f_3
r_1	0	1	0
r_2	0	1	1
r_3	1	0	1

Figure 2.6.1: Decision logic (Karlsson, 2001)

The *1's* indicate which residuals that are sensitive to each fault. For this residual structure, assume for example that residuals r_2 and r_3 are responding, and r_1 is not. Then the conclusion is that fault f_3 has occurred. A large part of all fault-diagnosis research has been to find methods to design residual generators. Of this large part, most results are concerned with linear systems. A characteristic of this approach to fault diagnosis is that faults are modelled as signals. This is very general and might therefore seem to be a good solution.

However, the generality of this fault model is actually its drawback (Frisk, 2001). Many faults can be modelled by less general models, and we will see in this thesis that to facilitate isolation, this is necessary in many situations. Another limitation is that the residual structure, with its 0's and 1's, places quite strong requirements on the residual generators. A 1 more or less means that the corresponding residual must respond to the fault. It can be understood that for small faults in real systems, with noise and model uncertainties present, this requirement is often violated. A third limitation, related to the previous limitation, is that the decision procedure, of how the diagnosis statement is formed from the real-valued residuals, does not have a solid theoretical motivation. For example, in the context of deciding the diagnosis statement, what are the meanings of the 0's and the 1's, and what does it mean that a residual is above the threshold? It would be desirable to use a decision procedure for which we can find an intuitive formalism based on existing well-established theory, preferably mathematics if possible.

One way of evaluating a fault by means of a generated residual is shown in Figure 2.6.2 below. In this case the plant represents a transfer function where a fault can be introduced. The neural network ANN_1 will simulate the behaviour of the plant under normal conditions. The residuals generated as a result of deviations between the two outputs will indicate the presence or absence of faults. One way of diagnosing the faults is to use a second neural network (ANN_2) to evaluate the residuals. However, in this report, faults are diagnosed by evaluating the properties of the residuals. Details on how to evaluate the properties of residuals will be covered in the next section.

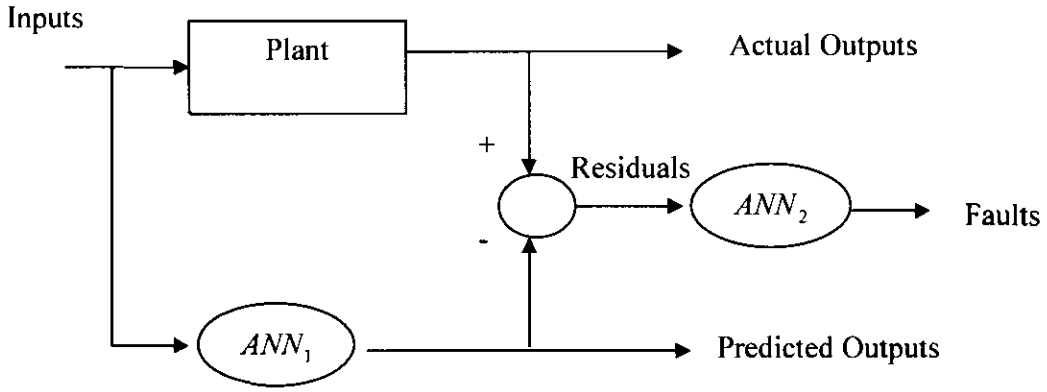

Figure 2.6.2: Model to diagnose faults (Olsson, 2002)

2.7 Threshold definition

In order to detect fault quantitatively, the thresholds have to be defined for the residuals. It is very important that the definition of thresholds for the residuals is independent of disturbances. The disturbances come from unknown input noise signals, modelling errors, etc.

Because of the presence of noise disturbances and other unknown signals acting upon the monitored variable, the residuals are usually stochastic variables with *mean value* and *standard deviation* for fault free processes.

If the distribution and variance of the noise is known, it is easy to determine the threshold. This method employs a fixed threshold and is therefore easy to implement.

Analytic symptoms are obtained as changes of residual signals with reference to the normal values. To separate normal from faulty behaviour, usually a fixed threshold has to be selected. By this means, a compromise has to be made between the detection of small faults and false alarms. The start of the fault can be easily detected by the positive peak (maximum) and the end of the fault can be detected by the negative peak (minimum). This means that when a fault occurs, one or more components of the residual vector will change in magnitude and make it possible to recognise that some change has taken place.

The problem with a fixed threshold is that some part of the signal is ignored. Fixed thresholds are only concerned with the maximum and minimum peak of the signal. However, the basic idea of adaptive thresholds is that since disturbances and other uncontrolled effects vary with time, the thresholds should also vary with time instead of being fixed at a constant value. The adaptive threshold adapts to the disturbances and therefore follows the test quantity as long as there are no faults.

One way of setting the thresholds is to perform a large number of simulations. No two simulations will give exactly the same result since noise is present. The threshold is then set according to a worst case scenario. This will give a system that is unlikely to fire false alarms but unfortunately there is a risk for missed detection instead. The thresholds might be set so high that an alarm is not even generated when a fault is present (Olsson, 2002). This report will demonstrate the ideas used to limit missed detections and false alarms. The impact of fixed and adaptive thresholds will be investigated in this report.

2.8 Neural networks

2.8.1 Introduction

The operation of any industrial plant is based on the readings of a set of sensors. The ability to identify the state of operation, or the events that are occurring, from the time evolution of these readings is essential for the satisfactory execution of the appropriate control actions.

In supervisory control, detection and diagnosis of faults, adaptive control, process quality control, and recovery from operational deviations, determining the correct mapping from process trends to operational conditions is the pivotal task. Reasoning in time, however, is very demanding, because time introduces a new dimension with significant levels of additional freedom and complexity. The real-time history of scores of variables can be displayed and monitored in most computerised process monitoring and control systems.

However, whereas a simple visual inspection of displayed trends is sufficient to allow the operator to confirm the process status during normal, steady-state operations, when the process is in significant transience or crises have occurred, the displayed trends of interacting variables and alarms can easily overwhelm an operator. When process variables change at different rates, or are affected by varying lags, it is very difficult for a human operator to carry out routine tasks, such as distinguishing normal from abnormal conditions, identify the causes of process trends, evaluate current process trends and anticipate future states, etc.

In order to carry out fault diagnostics, some representation (or reference) of correct or normal behavior has to be developed. This reference is the most important part of a fault diagnosis system. The consequences of a poorly defined reference are a failure to detect faults or the generation of false alarms. A model-based approach to diagnostics involves using a mathematical description of the system as a reference of correct behaviour. A diagnostics scheme can use various types of models, such as first-principles models, neural networks, fuzzy rules, characteristic curves, etc. This report advocates the use of neural network models, which is briefly described in this section.

2.8.2 Introductory theory of neural networks

An *Artificial Neural Network* (ANN) is a network of many very simple processors ("units"), each possibly having a small amount of local memory. The units are connected by unidirectional communication channels ("connections"), which carry numeric (as opposed to symbolic) data. The units operate only on their local data and on the inputs they receive via the connections.

The design motivation is what distinguishes neural networks from other mathematical techniques. A neural network is a processing device, either an algorithm, or actual hardware, whose design was motivated by the design and functioning of human brains and components thereof (Haykin, 1994).

There are many different types of neural networks, each of which has different strengths particular to their applications. The abilities of different networks can be related to their structure, dynamics and learning methods.

ANNs are particularly suited to deal with the problem of system identification in dynamic processes for several reasons (for a general reference on neural networks see (Hassoun, 1995)). First of all, ANNs can approximate any well-behaved function with an arbitrary accuracy, which is an essential advantage on methods based on regression when the problem at hand presents essential nonlinearities (Hunt, 1992; Willis, 1991). One should stress that, in some applications, ANNs do not outperform other system identification methods. The biggest advantage of ANNs manifests itself when dealing with hard problems, e.g. in the case of significantly overlapping patterns, high background noise, and dynamically changing environments. The ANN's characteristics of adaptive learning generalisation ability, fault tolerance, robustness to noisy data and parallel processing makes it a very interesting candidate for approaching the identification of dynamic events.

The success of neural networks' abnormality detection depends strongly on the ability to create a model for normality. On first sight, this may seem an impossible task as for

several reasons the classification will never be accurate. Such is true but only in a numerical sense. Because of the non-linear curvature in the error space and the incomplete, irreproducible and noisy character of the learning data, a specific sample will almost never be 100% correctly reproduced (Spaanenburg).

2.8.3 Analysing the problem

Where processes to be modelled are complex enough to be described mathematically, neural networks are considered to outperform the conventional, deterministic models most of the time. However, one should be aware of the applicability of neural networks to a specific problem and the basic conditions for getting the best performance out of it. In many cases neural networks for research are used 'blindly' by choosing all the possible input variables and without considering much of the possibilities to maximise the performance.

In general, neural networks are suitable for problems where the underlying process is not known in detail and the solution can be learned form the input-output data set. Nevertheless, the following points have to be stressed:

- It has to be made sure that the problem is difficult to be solved by conventional methods and that neural networks can be used as a good alternative.
- If there are logical non-chaotic relationships or structural properties that similar initial configurations indicate mapping to the similar solutions, one can expect a generalisation by neural network. It simply means the same input should always result in the same output.
- If the data set to train the network is impossible to be represented or coded numerically, the problem cannot be solved by a neural network approach
- Non-linearity and the change of variables in time are possible to be dealt with by neural networks.

Training the network has to be started by defining the topology of the neural network. The best topology is found by adjusting the parameters by trial and error, therefore it is better to start with a small network which learns fast and is easy to change the parameters. Initial weights are also defined by trial and error. When the appropriate network topology is defined, it is possible to speed up or slow down the process by changing the learning rate and fine-tuning.

This is one of the most important stages of any neural network application because the accuracy of the solution for most of the networks depends on the quality and quantity of the training data set. Although neural networks can accept a wide range of inputs, they work with data of certain format encoded numerically.

2.8.4 Training of neural networks

Artificial neural networks are designed to operate in a similar manner to their biological counterparts. Biological neural networks in the brain have neurons that receive input stimuli, which are amplified or attenuated by other neurons based on past learning experience, and the outputs are passed to other neurons through synapses. The final output is based on a combination of the output of other neurons.

Artificial neural networks use a similar method by training the network using known inputs and expected outputs. The network continuously adjusts a series of weights associated with each neuron as the network is trained.

A neural network is required to go through training before it is actually being applied. Training involves feeding the network with data so that it would be able to learn the knowledge among inputs through its learning rule. There are three types of training algorithms - initialisation algorithms, supervised learning and unsupervised learning. Initialisation algorithms are not really training algorithms at all, but methods to initialise weights prior to training proper. They do not require any training data.

In supervised learning, the network is shown a series of input and expected output examples. The expected output is compared with the actual output from the network. The network will adjust its weights to accommodate each training example. The purpose of adjusting the weight here is to minimise the difference between the two outputs. The learning rule is used to adjust the weights and biases of the network in order to move the network outputs closer to the targets. The perceptron multilayer learning rule falls in this supervised learning category.

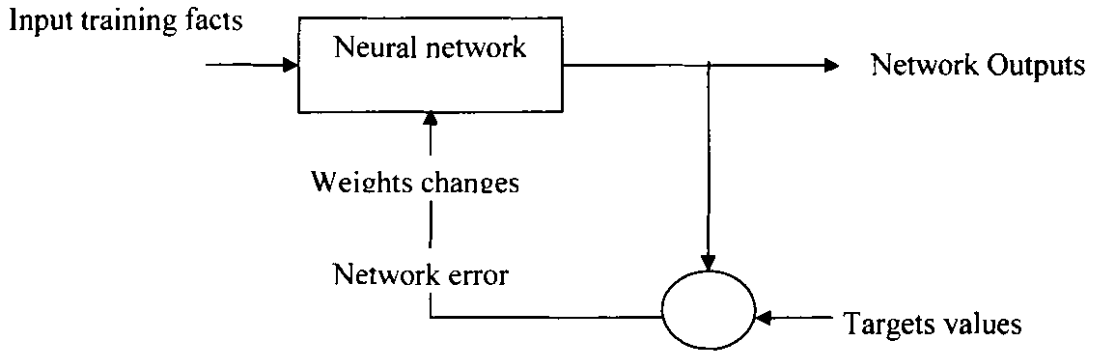

Figure 2.8.4.1: Supervised learning (Howard, 1996)

For unsupervised learning, the network is only presented with the inputs but not the output. The network in response to the input patterns updates the weights. That implies that there are no training data like supervised learning.

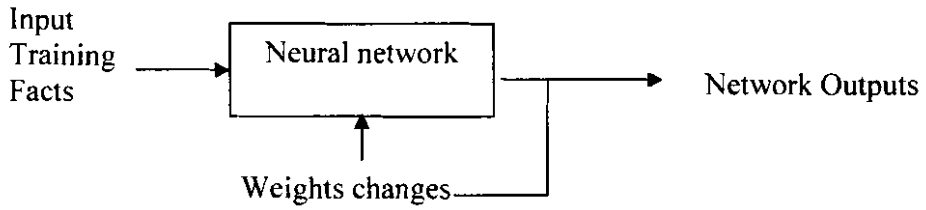

Figure 2.8.4.2: Unsupervised learning (Howard, 1996)

2.8.5 Multi layer perceptrons

There are many network models, or architectures, of neural networks. The type of neural network normally used for fault identification and diagnosis is the multilayer feed-forward neural network. The term "multilayer" signifies that the neurons are arranged in multiple layers. A "feed-forward" neural network indicates that information always flows through the network in a forward direction, from inputs to outputs; that is, there is no feedback to previous layers. This type of neural network can be trained using sets of known inputs and expected outputs. This method of training is known as supervised learning.

Multi Layer Perceptrons (MLP) can be trained with the back-propagation algorithm that has proven to be very successful in many diverse applications. The back-propagation algorithm is based on an error-correction learning rule. The algorithm searches for the minimum in the multidimensional error-surface by following the steepest descent. Learning of the MLP consists in adjusting all weights such that the error measure between the desired output signals d_{jp}, and the actual output signals y_{jp} averaged over all learning examples p will be minimal (possibly zero). The standard back-propagation learning algorithm uses the steepest-descent gradient approach to minimize the mean-square error function as shown in equation 2.3 – 2.4 below.

$$E_p = \frac{1}{2}\sum_{j=1}\left(d_{jp} - y_{jp}\right)^2 = \frac{1}{2}\sum_{j=1} e_{jp}^2 \quad \text{..... 2.3}$$

The total error function is $E = \sum_p E_p = \frac{1}{2}\sum_p \sum_j \left(d_{jp} - y_{jp}\right)^2$ 2.4

Where d_{jp} and y_{jp} are desired and actual output signal of the j^{th} output neuron for the p^{th} pattern, respectively. MLP neural networks are very flexible mathematical functions of their inputs, making it very easy to overfit the data. To avoid overfit, it is therefore necessary to somehow constrain the modelling process.

Typically, a MLP begins training with a poor fit to the data (due to its random weight initialisation). As training progresses the neural networks fit to the data improves. At some point, however, the neural network begins to overfit the data, meaning that its performance on the training data continues to improve, but only because it is beginning to memorise the peculiarities of the training cases, not because it is learning more about the underlying process. Remember, the object is to have the neural network generalise usefully to new cases, not memorise the training cases.

It is obvious that the model performance will be overly optimistic if any of the test data is included in the training set. It is less obvious that if you are tempted to look at the results on the testing set, and then return to the training to improve the performance, you are actually cooking the model. This problem can be solved by setting aside a group of data to be used as a *validation* set. This *validation* set is used as a final test of the model performance.

To improve the performance of a neural network the following steps needs to be done:

1. Elimination of weights which don't contribute to accuracy
2. Limiting the number of nodes
3. Start with few hidden nodes and increase the number by testing at each epoch
4. Preventing overtraining (to stop when the mean squared error stops improving)

2.9 Summary

In this chapter a survey on fault detection and diagnosis methods have been done. It was found that model-based diagnosis using neural networks is adequate for the problem. A theory on how to apply model-based diagnosis was covered.

Furthermore, this chapter has illustrated the use of residuals in fault detection and diagnosis. Most of the theory applied in this chapter followed from (Olsson, 2002). A theoretical survey of neural networks has been covered in this section. The next section focuses on the application of model-based diagnosis on four cascaded first-order transfer functions.

3. MODEL BASED FAULT IDENTIFICATON AND DIAGNOSIS

The goal of this chapter is to identify and diagnose faults by means of model-based diagnosis. A neural network is used as a model to mimic the normal behaviour of the plant. The intention of using neural networks is to generate residuals which will be evaluated to diagnose faults. This chapter illustrates the concepts of model-based diagnosis.

3.1. Overview

After a residual signal is derived, the evaluation of the residual to distinguish a particular fault from other possibilities follows. Faults can be classified by evaluating properties of the residuals together with a matrix that contains the decision logic. By using test quantities that decouple different sets of faults and performing hypothesis tests on these, the fault can be detected and hopefully also isolated. Each test quantity has a corresponding hypothesis test. When a fault is decoupled in a test quantity this means that the hypothesis test will not be sensitive to that particular fault.

Fault isolation can be performed using several different principles. The approach used here is a structure of hypothesis tests. This makes it possible to diagnose a large variety of different types of faults within the same framework and the same diagnosis system. A number of hypothesis test are performed individually, each one coming up with a statement. The statement from each test is a list of possible fault modes.

3.2 Model construction

For the purpose of fault diagnosis, a simple and accurate model is desirable. In this work, the simple transfer function plant system is modelled by evaluating the residuals' properties; that is, mean value and standard deviation of the residuals. The model shown in a previous section (figure 2.5.1) will now be developed as four cascaded first order systems.

The objective is to identify and diagnose faults on the entire system. To determine to what extent this can be achieved, a few plant models will be tested to determine whether is possible to identify and diagnose faults on the entire system. In addition, investigations will be done, amongst others, to determine whether faults propagate among the subsystems. The potential of fault detection in the case of multi-input multi-output systems will further be investigated in this section.

A model is first developed for the case when no fault is present. The model for the transfer function plant system is shown in Figure 3.2.1 below. Simple first order transfer function sections of form $\frac{a_i}{b_i s+1}$ were considered. The variables a_i and b_i are the key elements to evaluate the occurrence of faults. Faults can be classified as gain, offset, and change in time constant. These types of faults are typical instrument faults that can happen on sensors and may also be used to model some plant faults. Figure 3.2.1 depicts a plant model as well as a fault detection system, using neural networks as a tool to simulate the fault-free behaviour of the plant. Once a fault is induced in any of the plants, a discrepancy between the two systems (neural network and actual plant) will emerge. Those discrepancies are called residuals, and are the key to diagnose the system.

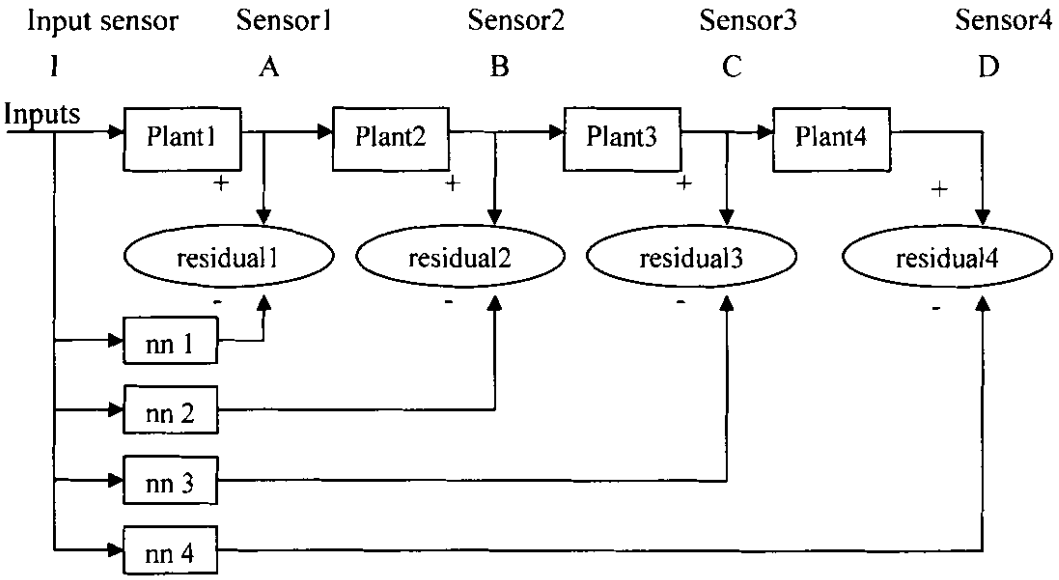

Figure 3.2.1: Plant model of four cascaded first order sections

The part I to A is modelled by neural network nn1. The two parts in series from I to B are modelled by neural network nn2; the three parts in series from I to C are modelled by neural network nn3; the four parts in series from I to D are modelled by neural network nn4. This model simulates the time response of a system of four cascaded first order transfer functions in which the following faults can occur:

- Change in gain of sensors at A, B, C and D.

- Change in offset of sensors at A, B, C and D.
- Change in plant offset disturbance at input, A, B and C.
- Change in plant time constant.
- Change in plant gain.
- Change in offset and gain of input sensor to all neural network.

A fault diagnosis system consists of a classification system that can distinguish between different faults based on observed symptoms of the process under investigation. Since the fault symptom relationships are not always known beforehand, a system is required which can be trained on experimental or simulated data. A neural network based process model simulator is advantageous. It allows for easy incorporation of a-priori rules and enables the user to understand the inference of the system.

Four *neural networks* (nn) are created and trained to model the fault-free behaviour. The residuals between plant and trained neural nets are used to identify the presence or absence of a fault or faults. The residuals are then used to diagnose the faults.

3.3 Creating a neural network for generating residuals

A three layered feed-forward neural network was used to model the plant behaviour. The neural network mimics the plant behaviour under normal conditions. Should any discrepancy emerge between the output of the neural network, and the output of the plant, residuals will be generated. The residual is designed as the difference between the real process output and neural network output.

There is no exact available formula to decide what architecture of ANN and which training algorithm will solve a given problem. The best solution is obtained by trial and error. Different nets were tried and the following works satisfactorily.

A three layered feed-forward network utilising resilient back-propagation, which institutes supervised learning, was created, using Matlab®. The input layer is composed

of tansig transfer function with 6 neurons; see neural network toolbox design and simulation (Howard, 1996). The hidden layer is composed of three neurons with purelin transfer function (Howard, 1996), and the output layer is composed of one neuron with purelin transfer function. This is a standard set-up for multi-layer perceptrons which worked for this design. The hidden layer determines the network's complexity, and hence determines the number of training epochs needed to achieve the desired result or output.

Data presented to the neural network were normalised to remove problems with scaling and signal units (such as say temperature and voltage) and filtered to remove spikes and noise since the performance of the neural network depends on the training data presented to it. Poor input and output data may cause a neural network to fail to converge to the desired level of accuracy.

Network learning pertains to training an untrained network. Input patterns are exposed to the network and the network output is compared to the target values to calculate the error. which is corrected in the next pass by adjusting the synaptic weights.

The training accuracy was set to within 0.001 of the target data. The target data is used to measure the *Mean Squared Error* (MSE) of the output, which is obtained from the difference between the network outputs and the target outputs. The weights and biases calculated during this phase are saved for use in the simulation of the network. The network stops training if an error goal has been reached, or when maximum number of epochs has been reached.

One of the most important factors to construct a neural network depends on what the network will learn. A neural network must be trained on some input data. The two major problems in implementing the training are: defining the set of input data to be used (the learning environment) and deciding on an algorithm. However, there are many different types of neural network algorithms in use. Some are optimised for fast training, others for fast recall of stored memories, others for computing the best possible answer regardless

of training or recall time. But the best model for a given application function depends on the data and the function required. Network training means adjusting neural network weights. During training the network analyses the data provided to it and changes weights between network units to reflect dependencies found in your data.

The neural network was trained using 40% of the data set presented. The network was simulated using the entire data set. Remember, the object is to have the neural network generalise usefully to new cases, not memorise the training cases.

After completing several simulations for predicting the plant response with the back-propagation learning algorithm, it is concluded that the average error for simulations using lots of data is smaller than that using less amount of data. That is, the more data for training the neural network, the better prediction it gives. If the training error is low, predicted response are close to the real response.

The results of how the neural network performed will be shown in this chapter. The intention was to be able to use neural networks to generate residuals. The feed-forward network utilising resilient back-propagation algorithm proved adequate for the problem.

3.4 Implementation of a diagnosis system

Once residuals have been determined, the next step is to evaluate the symptoms of such residuals. In order to diagnose faults, the behaviour of each fault candidate must be determined.

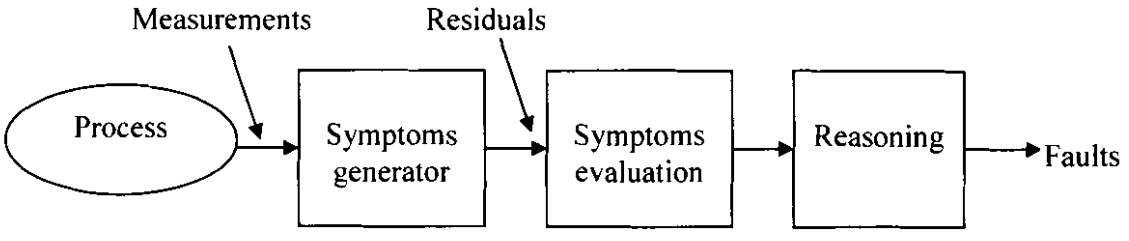

Figure 3.4.1: Process of constructing a diagnosis system (Rakar)

In the evaluation stage one has to detect if a residual is significantly different from zero. The evaluation must be robust in order to avoid false alarms, but also sensitive enough to detect even smaller faults which cause only small residual changes. Many approaches make use of thresholds to separate deviations caused by faults from deviations caused by other reasons, such as noise and modelling errors.

The design of the threshold represents a compromise between robust evaluation (high threshold) and sensitive detection (low threshold). The decision logic with proper reasoning tries to isolate faulty components on basis of the generated matrix patterns of residuals. As a result, the mapping from a set of residuals into a distribution of beliefs for each fault candidate is done.

The primary goal of a diagnostic system is to detect anomalous system behaviour and then to isolate the cause for such a behaviour. There is no universal method, which would be able to cope with all kinds of system faults. Therefore it is important to set clear diagnostic statements beforehand.

Inserting a single fault in the plant and recording its effects provides a signature of the faulty process. Having a signature for the fault-free plant and for the plant with a known fault the decision logic allows characterising the process for the potential presence of such faults. Unknown faults that cannot be handled should be analysed off-line and added to the structured logic for later usage.

The system can only diagnose faults which are clearly classified in the decision logic matrix. It means that the behaviour of faults under different circumstances must first be identified. Therefore a matrix that illustrates different combinations of faults is needed to diagnose the faults.

Table 3.4.1 shows a maximum and a minimum threshold. These values were found by studying the no fault behaviour of the plant. The reason for using minimum and

maximum thresholds is that, some faults occur on the negative margin, especially the offset error. At some stage when using one threshold; there was missed detection.

As can be seen in Table 3.4.1, the maximum threshold for the residuals is 0.16474, and the minimum threshold is -0.049184. In order to isolate each occurred fault uniquely, decision logic is needed. Firstly, residuals are generated as explained previously. The properties of each occurred residual is evaluated using a standard deviation and mean value. The maximum and minimum thresholds for residuals were used to separate the occurrence of faults from normal operation. Faults were introduced into the plant as constant values for fixed times, chosen between 0.1 and 00.1, which cause changes in residuals' amplitudes. These residuals are constantly monitored by fixed threshold values for any deviation.

If a negative or positive peak of the faulty signal exceeds its corresponded normal residual threshold, a binary value 1 will be assigned to represent the occurrence of a fault. If a faulty signal does not exceed its corresponding normal threshold, a binary value 0 will be assigned for that particular residual, to symbolise fault-free operation. Decision logic will be formed to represent the occurrence of faults. This decision logic will further be used to identify and isolate the faults that occurred. Thus the basic problems in the area of fault diagnosis is how the procedure for generating the diagnosis statement should look like, what parameters or behaviour that are relevant to study, and how to derive and represent the knowledge of what is expected or normal.

Residuals which are generally noisy can be used in raw form or in processed form. Firstly, properties of residuals were evaluated using a statistical approach; that is mean value, median value, standard deviation, variance, and residual squared were calculated. It was found that the mean and the median value respond in the same way. After a couple of experiments, it was concluded that the contribution of the standard deviation, and the mean value can be used to classify faults.

The threshold was set to distinguish between no faults and faulty conditions. Table 3.4.3 shows decision logic in matrix format, as determined experimentally. First a threshold under normal condition was determined. That threshold was then used to detect faults. It was further assumed that only one fault occurred at a time. Multiple faults can also be diagnosed, for the purposes of this section only a single fault was considered.

One can see that the decision logic is represented by a unique matrix, except decision logic for plant 4. The problem was that there were two types of residuals from plant 4 to isolate 4 different types of faults. Faults from preceding sections may propagate forward to plant 4, but faults cannot propagate backwards from plant 4 since the system is an open loop.

The faults in the other 3 plants were easily isolated, the decision logic representing the faults are unique. One can see that faults propagate from one plant to another, especially the plant faults. The propagation of faults does not have an impact on the isolation of occurred faults. If a fault is inserted in plant 1 it will affect the residuals of the other plants.

Faults Logic based on residuals or properties of residuals	**Faults maximum threshold**	**Faults minimum threshold**
r_1 mean (r_1)	0.015677	-0.0268
r_1 std (r_2)	0.068139	0.022273
r_2 mean (r_3)	0.013745	-0.022428
r_2 std (r_4)	0.0562	0.018604
r_3 mean (r_5)	0.0219	-0.0463
r_3 std (r_6)	0.1005	0.022304
r_4 mean (r_7)	0.04248	-0.0492
r_4 std (r_8)	0.16474	0.021782

Table 3.4.1: Maximum and minimum threshold values

Fault type	**Plant/ Sensor**	**Notation**
No faults	All 4 plants	nf
Offset	Sensor 1	f_1
Change in gain	Sensor1	f_2
Offset disturbance	Plant 1	f_3
Change in gain	Plant 1	f_4
Change in time constant	Plant 1	f_5
Offset	Input sensor (to all nn`s)	f_6
Change in gain	Input sensor (to all nn`s)	f_7
Offset	Sensor 2	f_8
Change in gain	Sensor 2	f_9
Offset disturbance	Plant 2	f_{10}
Change in gain	Plant 2	f_{11}
Change in time constant	Plant 2	f_{12}
Offset	Sensor 3	f_{13}
Change in gain	Sensor 3	f_{14}
Offset disturbance	Plant 3	f_{15}
Change in gain	Plant 3	f_{16}
Change in time constant	Plant 3	f_{17}
Offset	Sensor 4	f_{18}
Change in gain	Sensor 4	f_{19}
Offset disturbance	Plant4	f_{20}
Change in gain	Plant 4	f_{21}
Change in time constant	Plant4	f_{22}

Table 3.4.2: Faults and their notations

Decision structures:

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
nf	0	0	0	0	0	0	0	0
f_1	0	1	0	0	0	0	0	0
f_2	1	1	0	0	0	0	0	0
f_3	0	1	0	1	1	1	1	1
f_4	1	1	1	1	1	1	1	0
f_5	0	1	1	1	1	1	1	0
f_6	1	0	1	0	1	0	1	0
f_7	0	1	1	1	1	1	1	1

Table 3.4.3: Decision structure plant1

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
f_8	0	0	0	1	0	0	0	0
f_9	0	0	1	1	0	0	0	0
f_{10}	0	0	0	1	1	1	1	1
f_{11}	0	0	1	1	1	1	1	1
f_{12}	0	0	1	1	1	1	0	0

Table 3.4.4: Decision structure plant2

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
f_{13}	0	0	0	0	1	0	0	0
f_{14}	0	0	0	0	1	1	0	0
f_{15}	0	0	0	0	0	1	0	1
f_{16}	0	0	0	0	1	1	1	0
f_{17}	0	0	0	0	1	1	1	1

Table 3.4.5: Decision structure plant3

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
f_{18}	0	0	0	0	0	0	1	1
f_{19}	0	0	0	0	0	0	1	0
f_{20}	0	0	0	0	0	0	1	1
f_{21}	0	0	0	0	0	0	1	0
f_{22}	0	0	0	0	0	0	0	0

Table 3.4.6: Decision structure plant4

The above tables, shows that it is possible to isolate different faults by using decision logic. The results serve as proof that different plant faults can be detected and isolated. As explained, the system consists of four cascaded plant sections, in which faults can occur from plant 1 to plant 4. The faulty signal from plant 1 can cause a deviation of the normal signal on the other plants. However the impact of such deviation does not affect the identification and isolation of faults accurately, as all residuals contribute to the classification of faults.

3.5 Simulation results

There are number of factors that affect the ability of the diagnosis system to detect and isolate faults. The most important one is the model itself, because in order to use a model for model-based diagnosis the model has to be accurate at least under the circumstances that the diagnosis system is supposed to work. It follows that an inaccurate model may cause the difference between non-faulty residuals, and faulty residuals to be either too small or large to detect.

It is also important that the thresholds are well adapted to the model faults so that there are few false alarms and so that even small faults can be detected. The residuals and decision structure also have to be correct; otherwise there is a risk of isolating the wrong components.

Figure 3.5.1 shows the response of the plant compared to a neural network. These results are for the plant model shown in Figure 3.2.1. The output data were normalised. The blue graph is the actual response and the red graph is the neural network response.

One can see that the neural network correctly mimic the plant response. This shows that the model-based algorithm is implemented correctly, with the neural network mapping the plant response.

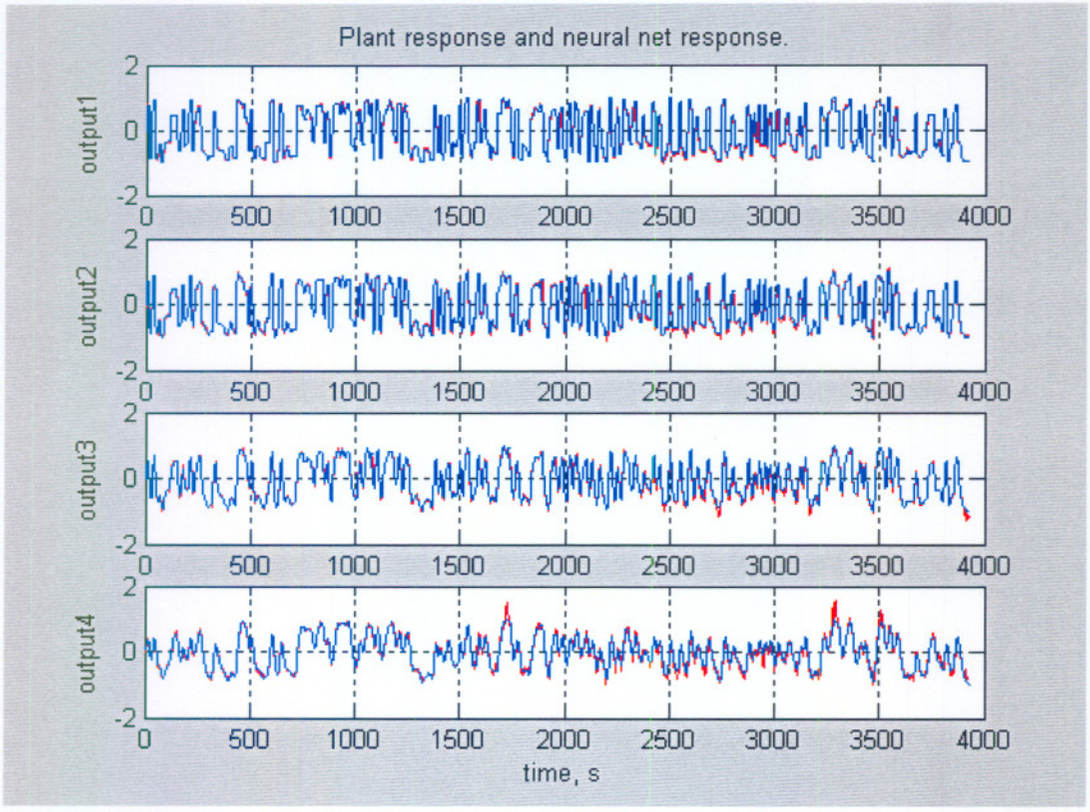

Figure 3.5.1: Normalised plant response compared to neural network response

In model-based diagnosis model building is essential. The results from the diagnosis system are directly dependent on how accurate the model is. Since the values from the model will be compared with the values from the physical system they must behave in the same way if not unacceptably large thresholds need to be used. There are several ways of building a software model and one common way will be presented here. For a full description of different model designs, see (Glad and Ljung, 1991).

If the system's physical behaviour is easy to understand and the system is not too big or complex it might be a good idea to build a unique model. In this kind of model building every physical relationship is modelled as equations in some software language, for example Simulink® in Matlab®. Naturally this demands good system knowledge and good understanding of how each element within the system works. It has the advantage that the model does not waste any parameters on estimating redundant information, which might be the case with a parametric model.

A unique model also makes it easier to estimate whether the results from the model are accurate or not. Since every physical component is considered it is also easier to understand how a fault influences the system and the fault is also easier to model.

Failing to get a good representative system model may result in difficulties to diagnose faults. The residual formed depends on the accuracy of the system and its model.

Since this master thesis uses a "model of a model" to build the diagnosis system the focus has not been on optimising neither the model nor the thresholds. The main objective was to explore the principles of model-based diagnosis, not to build an optimal diagnosis system for the model.

However one key element of diagnosing faults is to set a threshold. Figure 3.5.2 shows how the threshold can be used to separate the occurrence of faults. Once the signal exceeds the threshold (red line), it is assumed that there is a fault in the system.

One can see the consequences of using one threshold. Some of the faults can occur on the negative margin. This can result in missed detection of the fault; therefore it is advisable to use both a maximum and minimum threshold. Figure 3.5.3 below depicts a missed detection of a fault. As can be seen the faulty signal has shifted into the negative margin of the y-axis. This resulted in missed detection of the fault.

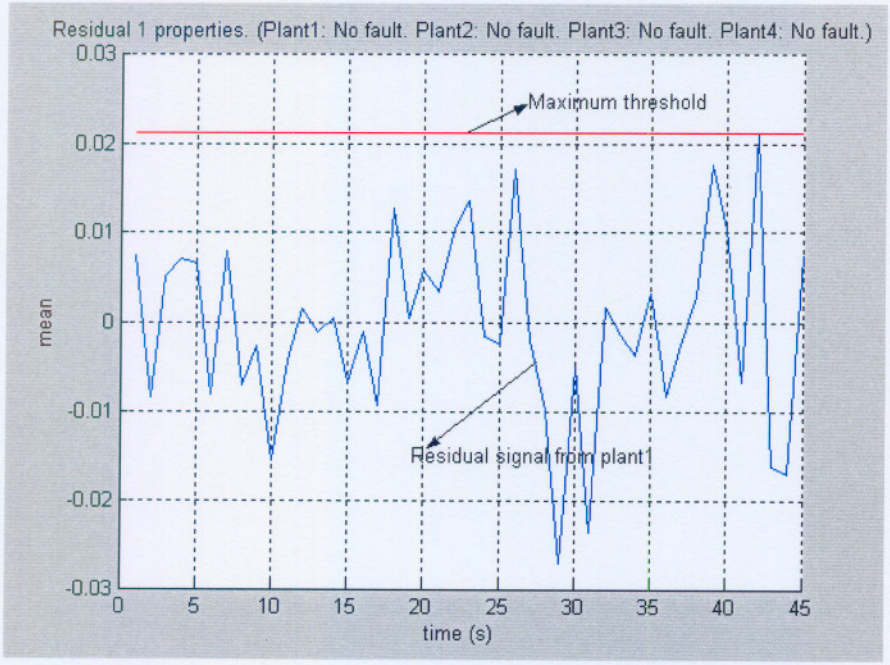

Figure 3.5.2: Setting threshold for residual

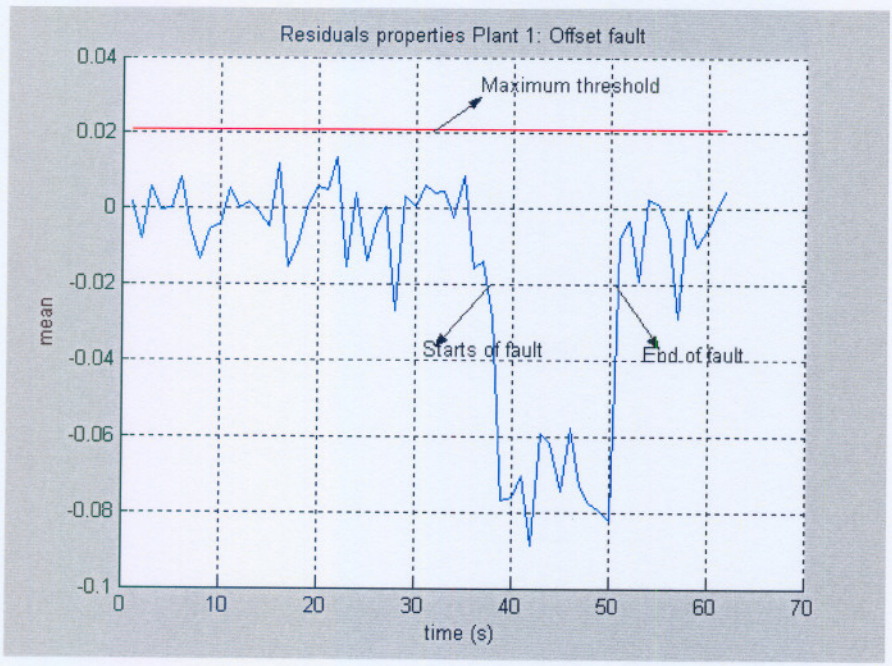

Figure 3.5.3: Impact of threshold on residual

Figure 3.5.4 below depicts the impact of both maximum and minimum threshold. As one can see, the occurred fault is on the negative margin, and was accurately detected by the minimum threshold. Due to the induced offset fault on plant 1, the residual properties (mean) shifted to negative margin. The simulated run was performed as follows: First the plant operates in fault free-state, after 27seconds a fault is inserted in plant 1, after 38 seconds the plant return to normal operation.

It is important to distinguish between the change in signal due to noise or other disturbances, and the one caused by the induced faults. However the impact of noise in this simulation was insignificant, as the neural network correctly fits into the plant response. The goal of the residual was to extract symptoms to identify and isolate faults and was successfully accomplished. Diagnosing faults on a real system might differ since a real system will need data on-line which may be affected by other factors such as noise and error of operators. One needs to differentiate between a fault signal and a noise signal. For this experiment it was assumed that the data produced is independed of noise and other disturbances, expect the intentionally inserted faults.

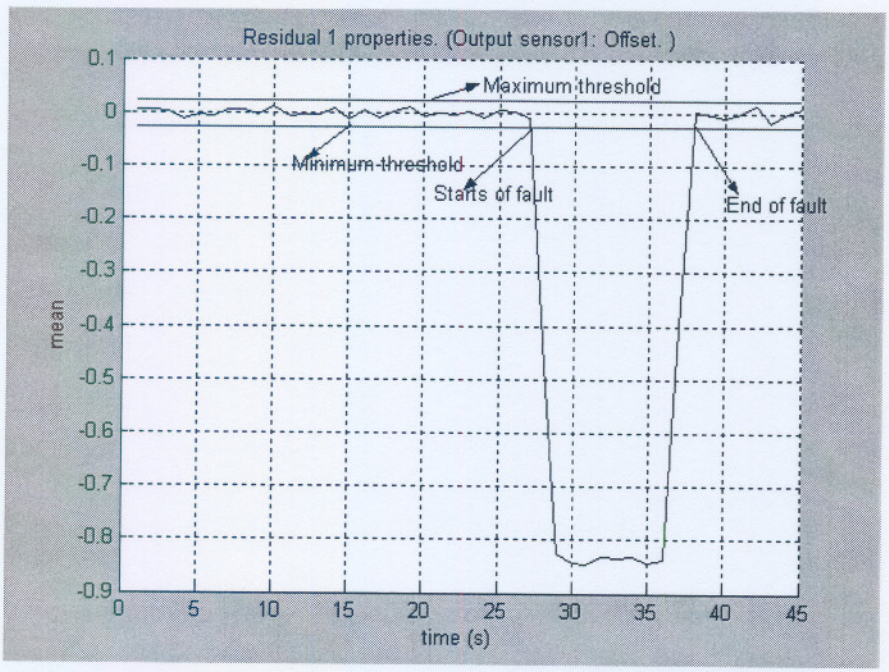

Figure 3.5.4: Fault detection using minimum and maximum threshold

3.6 A cascaded plant modelled with single-input single output neural networks

The same experiment was repeated with single neural networks input for each section. The intention, amongst others, was to investigate the potential of using different inputs, instead of using one input to all neural networks. In addition, to get an alternative way to incorporate the model for fault detection and diagnosis purposes.

Table 3.6.1 to 3.6.4 below shows decision logic used to isolate the detected faults. Note that the meaning of symbols was defined in Table 3.4.2. The location of the faults is the same as the one shown in Figure 3.2.1.

The experiment proved that training the network using single sections trained faster than training higher order sections using a single-input. It needs to be emphasized that the time taken to train the network does not have an impact on fault detection and diagnosis.

Figure 3.6.1 below depicts a single input single output cascaded network. Input **I** is used as an input to both plant 1 and neural network nn1. The output of plant 1 **A** is used as an input to neural network nn2 and plant 2. The output from plant 2 **B** is used as an input to neural network nn3 and plant 3. The output from plant 3 **C** is used as an input to neural network nn4 and plant 4.

The propagation of faults from the other plants still continues. This is because the faulty output from the previous plant is propagated to the next plant as shown in Figure 3.6.1. Figure 3.6.2 shows the detection of gain fault from plant1. Note that the fault was inserted in plant1 and propagated to other plants as shown in Table 3.6.1 (f_4).

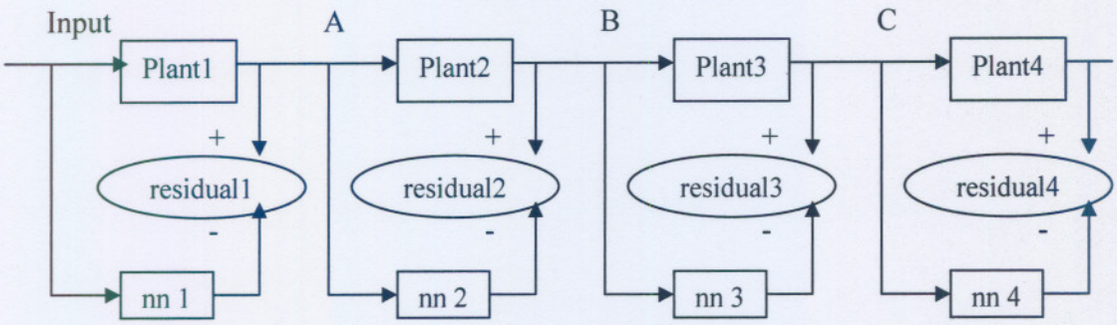

Figure 3.6.1: Four cascaded first-order sections

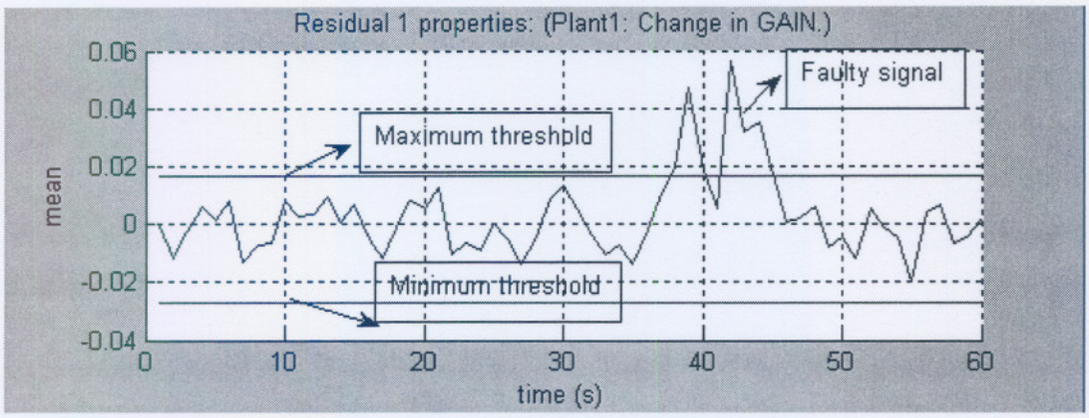

Figure 3.6.2: Detection of Plant fault

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
nf	0	0	0	0	0	0	0	0
f_1	0	1	1	1	0	0	0	0
f_2	1	1	1	1	0	0	0	0
f_3	0	1	1	1	1	1	1	1
f_4	1	1	1	1	1	1	0	0
f_5	0	1	1	0	0	0	0	0
f_6	1	0	0	0	0	0	0	0
f_7	0	1	0	0	0	0	0	0

Table 3.6.1: Decision logic plant1

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
f_8	0	0	0	1	1	1	0	0
f_9	0	0	1	1	0	1	0	0
f_{10}	0	0	0	1	1	1	1	1
f_{11}	0	0	1	1	1	1	1	1
f_{12}	0	0	1	1	0	0	0	0

Table 3.6.2: Decision logic plant2

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
f_{13}	0	0	0	0	1	1	1	1
f_{14}	0	0	0	0	1	1	0	0
f_{15}	0	0	0	0	0	1	1	1
f_{16}	0	0	1	1	1	1	1	1
f_{17}	0	0	0	0	1	1	0	1

Table 3.6.3: Decision logic plant3

	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
f_{18}	0	0	0	0	0	0	0	1
f_{19}	0	0	0	0	0	0	1	1
f_{20}	0	0	0	0	0	0	0	1
f_{21}	0	0	0	0	0	0	1	1
f_{22}	0	0	0	0	0	0	0	0

Table 3.6.4: Decision logic plant4

The design of a cascaded system gives an alternative way of modelling by using neural networks. This experiment has proved that in a complex multi-input multi-output plant and neural net model can be used for fault detection and diagnosis.

3.7 Summary

The methodology for building a system model with a neural network was described, and then criteria for evaluating model effectiveness were given. The neural network was mainly used to simulate the fault-free behaviour of the plant, in order to generate residuals.

The theory presented in chapter 2 was implemented in this section. This chapter was concluded with results to support the theory. The results for fault identification and diagnosis on four cascaded first order plant sections were described.

4. FAULT DETECTION AND DIAGNOSTICS ON A DRUM LEVEL CONTROL SYSTEM

To test the concepts on an industrial problem, a literature survey was done. A system for the drum level control of a boiler was selected (Kaliberda, 1999), see figure 4.2.1. In this plant the feedwater flow rate is controlled to match the steam flow rate. Since there can never be a perfect balance between these two flow rates, the level is measured and controlled around the set point by trimming the feedwater flowrate.

Traditional control methods were utilised to develop a control law to stabilise the water level in the drum. One important consideration is to keep the level of water in the drum at a desired set point because there is a risk that the water could completely evaporate or overflow into the steam system. The most dominant parameters that affect the water level are the flow rate of feedwater as well as the steam flow from the boiler. To simulate the control of the water level, a simple *Proportional, Integral and Differential* (PID) control has been adopted. The response of the plant was simulated using a Simulink® model (Kaliberda, 1999).

A feed-forward neural network was trained to mimic the response of the plant. The ability of a neural network to model a non-linear dynamic system is investigated in this chapter. These results serve as a verification of the ability of a neural network to model the actual process. This chapter is concluded by investigating fault detection and diagnosis methods for the drum level control system.

4.1 Goal

The goal of this chapter is to investigate, amongst others, the ability of neural network to model non-linear dynamic systems. The results to be obtained in this chapter will help to test whether is possible for the neural network to mimic the response of a controlled dynamic system accurately.

It is important to control the level of water in the boiler drum, because a fault in the control of the water level could cause serious malfunction with legal implications. A full description of the implementation of the controller will be covered.

Furthermore, the purpose of this section is to use Simulink® to model the dynamic behaviour of water level in the drum. Modelled data will be used to train neural networks for the dynamic behaviour of the process. The mathematical model that describes the behaviour of the water level was developed by Oleg Kaliberda (Kaliberda, 1999). Once the neural network has proved that it can model the dynamic behaviour of the system, fault identification and diagnosis will be carried out on the system.

4.2 Background

The water level regulator system is a classic control problem that is used in thermal power plant around the world. It is a suitable process to test prototype controllers due to its high non-linearity, transport time and potential instability. The system consists of measured physical variables to control the level of water in the drum.

In this chapter, the dynamical equations of the system will be used, the model will be developed in Simulink® and basic controllers will be developed. The aim of the Simulink® model is that the developed model will have the same characteristics as the actual process. It will be possible to test each of the prototype controllers in the Simulink® environment, obtain the fault-free response, introduce faults, and to obtain the response with faults. As before, neural networks will be trained on data when no faults are present. The residuals will be used for fault detection and diagnosis.

Figure 4.2.1 below depicts a schematic model for the feed water PID control system to be implemented in Simulink® to generate data for fault detection and diagnosis. The critical path of this process is to be able to control the level of water, given outflow of steam from the boiler and inflow of water to the drum. The numerical object characteristics are taken from the data for a 200 MW Thermal Power Plant. Derivation is out of the scope of this project, for references consider (Kaliberda, 1999).

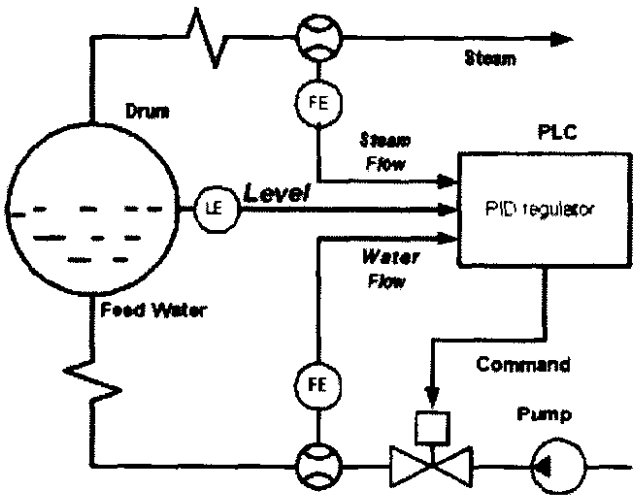

Figure 4.2.1: Feed water PID regulator (Kaliberda, 1999)

Feed water enters the drum through a valve. The heat energy converts the water in the drum into steam. The steam drives the turbine and the generator coupled to the turbine produces electricity. One important consideration is to keep the level of water in the drum at a desired set point because there is a risk that the water could completely evaporate. The water level in the drum should be maintained within a desired range in order to avoid a possible efflux of water droplets from the drum and the development of abnormal thermal stress in the drum wall. The most dominant paramcter that affects the water level is the flow rate of feed water. To simulate the control of water level, a simple PID control has been adopted. The standard three-element PID control works well under steady-state conditions. Since we are concerned with fault detection and diagnosis, the intricacies how the PID controller works in real thermal power plant is out of the scope of this project. An important feature of this feed water PID regulator is the method of calculating the

process variable derivative. Level derivative dL/dt is calculated not by numerical differentiation but as a difference of the measured *water flow* and *steam flow*:

$$dL/dt = K_{ob}\,(F_{water} - F_{steam}) \quad \text{..... 4.1}$$

Since there is no delay in these measurements, it raises the accuracy of the regulator in transient mode.

Object transfer function (Kaliberda, 1999)

$$W(s) = K_{ob}\left(\frac{1}{s}\right)\left(\frac{1}{Ts+1}\right)\left(e^{-sT_{delay}}\right) \quad \text{..... 4.2}$$

Regulator transfer function (Kaliberda, 1999)

$$W_{reg}(s) = K_p\left[1 + \frac{1}{T_i} + \frac{K_d s}{T_{fd}(s+1)}\right] \quad \text{..... 4.3}$$

The controller output from the PID is represented by (Kaliberda, 1999)

$$CO(t) = K_p\left[e(t) + \frac{1}{T_i}\left(\int_0 e(t)\,dt\right)K_d\,\frac{dx(t)}{dt}\right] \quad \text{..... 4.4}$$

The above equations are represented as a closed loop by the following figure

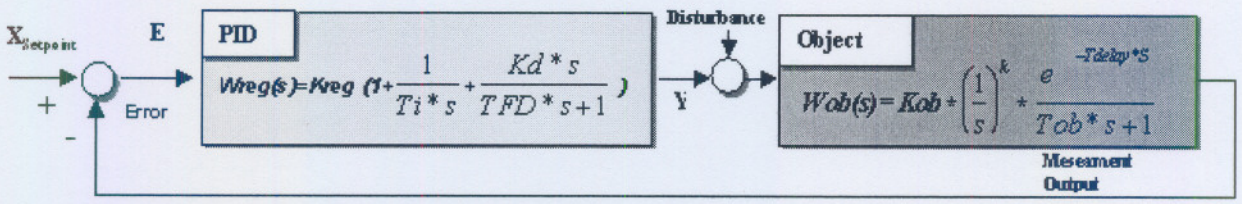

Figure 4.2.2: Closed loop model for feed water PID regulator (Kaliberda, 1999)

This closed loop model was further developed in Simulink® to model the dynamic behaviour of the water level inside the drum. The disturbance for the model is the change

in steam flowrate. The level of water depends on the steam flowrate from the boiler and the water flowrate to the drum. Thus, the measured variables are steam flowrate from the boiler, the water flowrate to the drum, and water level in the drum. The controller should be able to control the level of water inside the drum to the desired level, given a disturbance. The dynamic equation that describes the process behaviour is shown in the Simulink® model.

Name	Description
K_{ob}	Object gain
$W(s)$	Object transfer function
T	Derivative filter time
T_{delay}	Pure time delay
$W_{reg}(s)$	The regulator transfer function
K_p	The regulator proportional gain
T_i	The integral time
T_{FD}	Derivative filter time
K_d	The derivative gain
$e(t)$	a control error

Table 4.2.1: Symbol definitions

4.3 Methodology

4.3.1 Simulink Model

A set of equations (linear & non-linear) describing the PID regulator, and plant have been implemented. The next stage is constructing a Simulink model of the water level controller. The diagram below is the feed water PID regulator model. This model is constructed using integrators, gain blocks, etc. For details on deriving the model and implementation it in Simulink® refers to (Kaliberda, 1999).

The equations shown in equation 1-3 will now be further developed in Simulink® model. Note that only the mathematically equation describing the control of the water level was given by Kaliberda, therefore details description of every part of the Simulink® model will not be covered as information supplied by the source was insufficient.

The water level control system consists of PID controller, actuator, valve, and drum as shown in Figure 4.3.1.1. Figure 4.3.1.2 and Figure 4.3.1.3 show a Simulink® model for PID controller.

The purpose of the PID controller is to control the level of water to the desired system response. The controller uses three terms. The proportional term, Kreg, will push the system in the right direction. The derivative term, KD will respond quickly to changes. The integral term, Ti will respond to long-term errors. These terms are used to tune the controller to the desired level of water.

Furthermore, the PID controller consists of modulator with a relay histeresis as shown in Figure 4.3.1.4 and Figure 4.3.1.5 below. The purpose of the modulator is to maintain the level of water at desired set point by modulating the varying level of water.

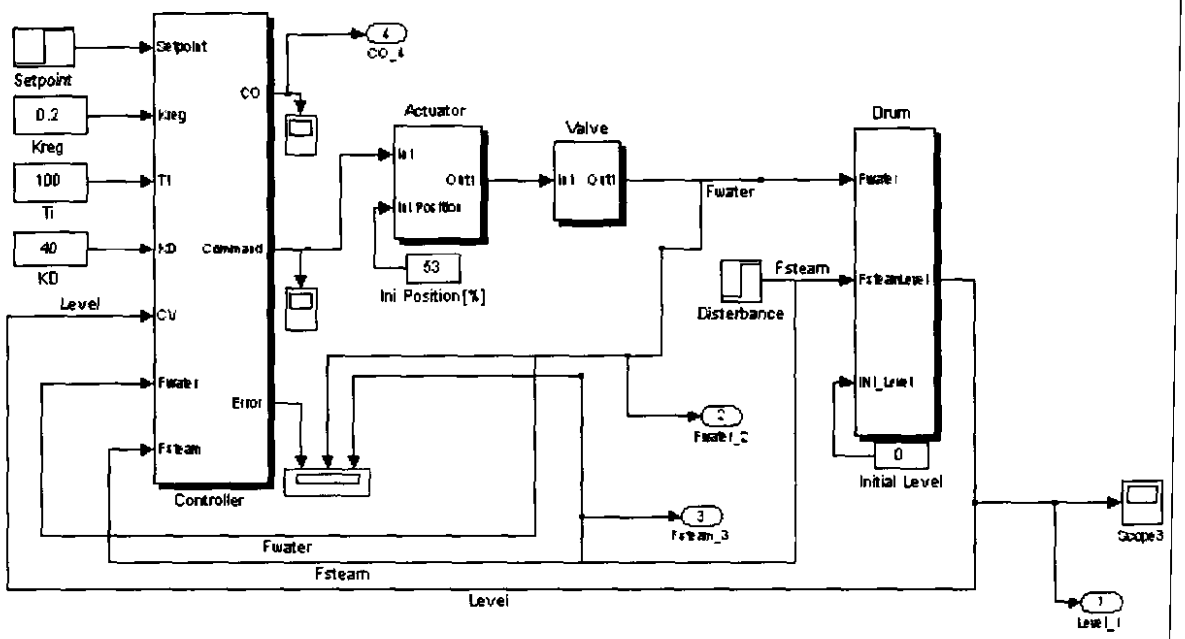

Figure 4.3.1.1: Simulink model of the water level PID regulator (Kaliberda, 1999)

$$dL/dt = K_{ob}(F_{water} - F_{steam}) \quad \text{..... 4.5}$$

Equation 4.5 is represented by Figure 4.3.1.2 in Simulink®.

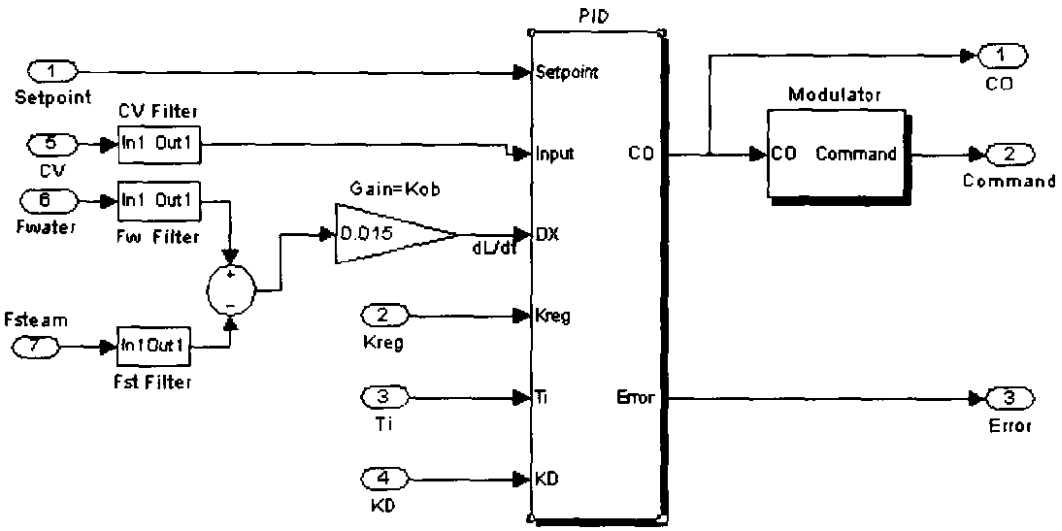

Figure 4.3.1.2: Simulink model of the PID

$$CO(t) = K_{reg}\left[e(t) + \frac{1}{T_i}\left(\int e(t)\,dt\right)K_d\,\frac{dx(t)}{dt}\right] \quad \text{..... 4.6}$$

The above equation is represented by Figure 4.3.1.3 in Simulink®.

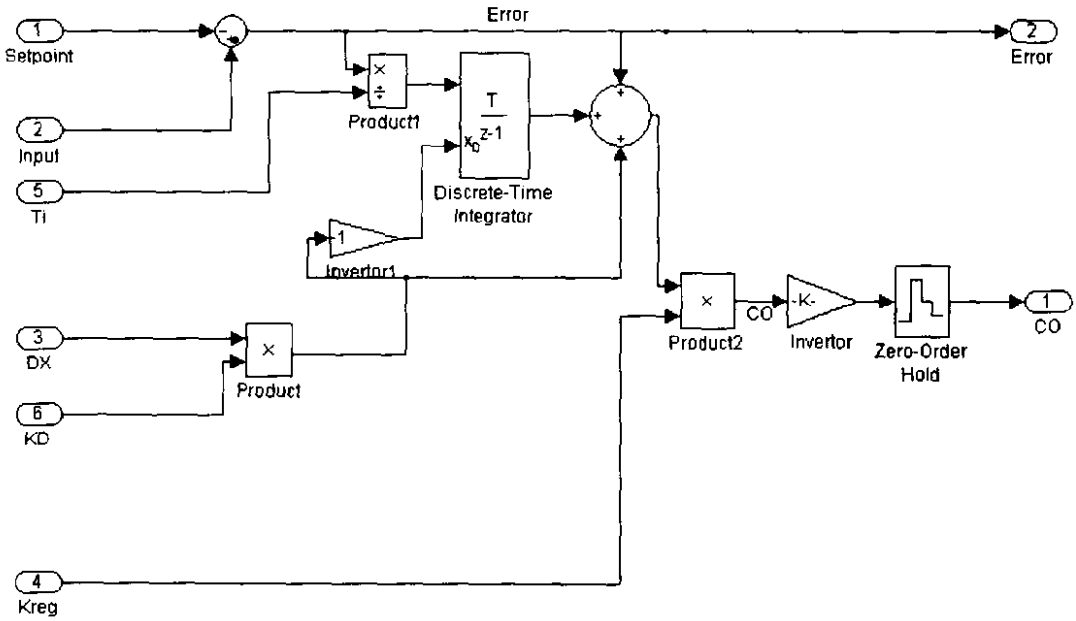

Figure 4.3.1.3: Simulink model of the PID Controller in detail

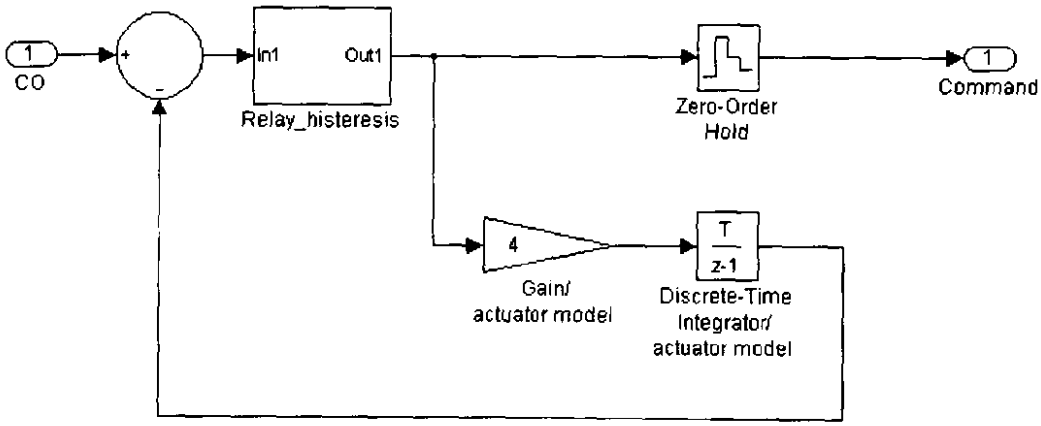

Figure 4.3.1.4: Simulink model of the Modulator

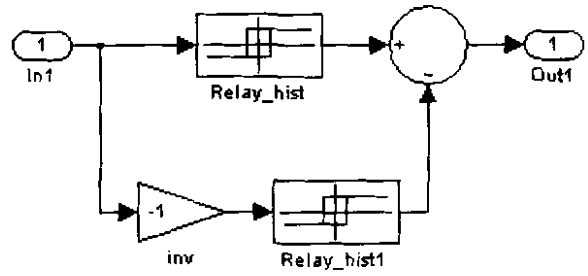

Figure 4.3.1.5: Simulink model of the relay histeresis

Figure 4.3.1.6 shows a Simulink® model for actuator. The purpose of the actuator is to drive the plant at desired level of water inside the drum.

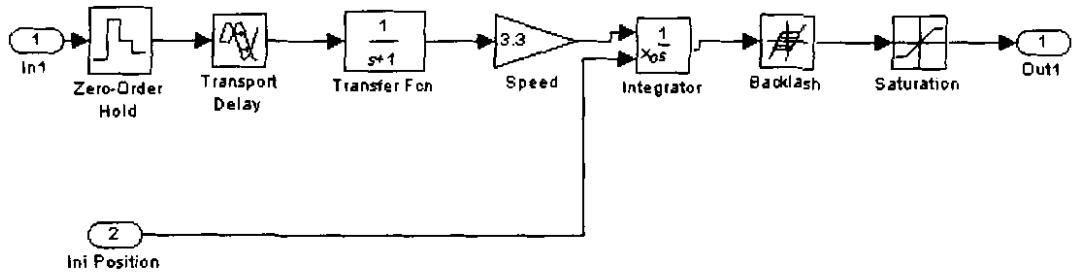

Figure 4.3.1.6: Simulink model of the actuator [17]

Figure 4.3.1.7 shows a Simulink® model for the valve. The purpose of the valve is control the flow of water into the drum.

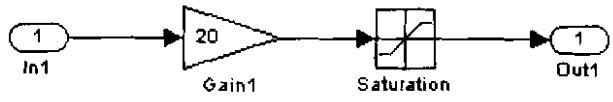

Figure 4.3.1.7: Simulink model of the valve

The Simulink® model for the drum is shown in figure 4.3.1.8. Its purpose is to give the level of water in the drum as the difference between flow of water and steam.

$$W(s) = K_{ob}\left(\frac{1}{s}\right)\left(\frac{1}{Ts+1}\right)\left(e^{-sT_{delay}}\right) \quad \text{.....4.7}$$

Equation 4.7 is represented by a Simulink® model in Figure 4.3.1.8 below.

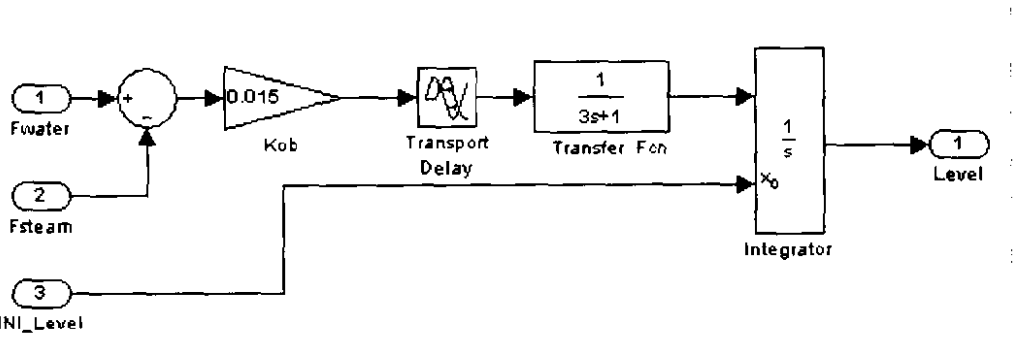

Figure 4.3.1.8: Simulink model of the water level (object)

Simulated response for a 20% disturbance of the steam flow to water level setpoint was generated to observe the time to reach its first peak, the overshoot, and the period of oscillation. The intention was to tune the PID values to fall into the setpoint requirements. It was found that the process reaches settling time at approximately 650s.

The system output that is used to train the neural network is the output from the *drum*, which is the level of the water on the drum.

4.3.2 Simulation results

4.3.2.1 Introduction

Using residuals for fault diagnosis relies on the accuracy of the predictive model. A residual is a signal generated from computation based on measured variables and reference model. It is ideally zero in the fault-free case and different from zero in the faulty case. In practice the generated residuals are not identically zero, due to various errors (measurement noise, modelling uncertainties).

Therefore the model generated by the neural network should be as close as possible to the actual model. The question is, will a neural network be able to generalise to the required accuracy of the system response, given ripple and step on the system? The description of how to build an accurate neural network model is next covered.

4.3.2.2 Building a neural network model

The selection of an appropriate number of hidden layers and the number of units in a layer is problem-dependent and typically requires considerable engineering judgment. For instance, by adding more hidden units and layers to a network, the agreement between the actual and target outputs may be improved but at the cost of increased training time and memory requirements. In addition, if too many hidden units are used over-fitting of the training data may occur and the generalisation to new input patterns may be poor. This is similar to the effect seen when curve fitting with too many free parameters.

The network was built by starting with a few number of neurons, and see how the network performed. In most cases a network that performs well with few training data points, turns out to perform the best with large amounts of data. At first the network seems to learn the behaviour of the process, only to find out it memorised the behaviour of the system, and failed to generalise to new sets of data. This process is normally called over-fitting, were the error on the training set is driven to a very small value, but when new data is presented to the network the error is large.

One of the reasons might be the data was taken directly from the Simulink® model and was not normalised. Figure 4.3.2.2.1 below shows actual response compared to the neural network response. One can see that the neural network seems to learn the dynamic behaviour of the system, except that there are some small ripples. However that is not the case for the network failed completely to map the behaviour of system. The number of epochs was increased to give the neural network enough time to learn the data presented on it. The network failed again to give good results of the underlying data.

One method for improving network generalisation is to use a network that is just large enough to provide an adequate fit. The larger a network you use, the more complex the functions the network can create. If we use a small enough network, it will not have enough power to overfit the data.

Unfortunately, it is difficult to know beforehand how large a network should be for a specific application. There are two other methods for improving generalisation that are implemented in the Neural Network Toolbox: regularisation and early stopping. Early stopping is to avoid over-learning (not the same as over-fitting but related) (Howard, 1996).

The data were normalised, and then the network started to generalise well. Normalisation of data helps the network not to overfit data. Normalising the data (subtracting the mean and dividing by the standard deviation) is important to ensure that the distance measure accords equal weight to each variable, without normalisation, the variable with the largest scale will dominate the measure.

The data was normalised using the neural network toolbox function PREMNMX, which preprocesses the network training set by normalising the inputs and targets so that they fall in the interval [-1,1].

Many researchers in the field of fault identification and diagnosis use the back-propagation training algorithm, which simplifies the effort of investigating different neural network topologies. It was found that the feed-forward network utilising trainbfg as training method is adequate for the problem.

The 'newff' function allows a user to specify the number of layers, the number of neurons in the hidden layers and the activation functions used. The hidden layer contains tan-sigmoid activation functions and the output layer contains a linear function. This is the standard set-up of activation functions in multi layer perceptron.

The number of neurons was 15-8-1 in the input, hidden and output layer respectively. The time delay on the hidden and output layer was fixed at 60 delays. In this case the number of delays has no effect on the performance of the network to fit the data correctly.

How would one know if a neural network generalise well? There are number of ways to determine the performance of the neural network depending on the problem in hand. One can calculate the mean squared error of the neural network to test the quality of the neural network. The MSE gives a good indication of the accuracy of the model. The MSE between the model and the process should be low. This will result in small residuals close to zero.

In this case it means that it will be possible to identify a small change in the residual. Figure 4.3.2.2.3 below shows the residuals between the actual output and neural network output. The residual is close to zero, which shows that the error between the actual output and the neural network output is very low.

The discrepancy between the two models has major impact on fault detection. Due statistically variations when training neural network, the neural network will not fit the data 100% accurately.

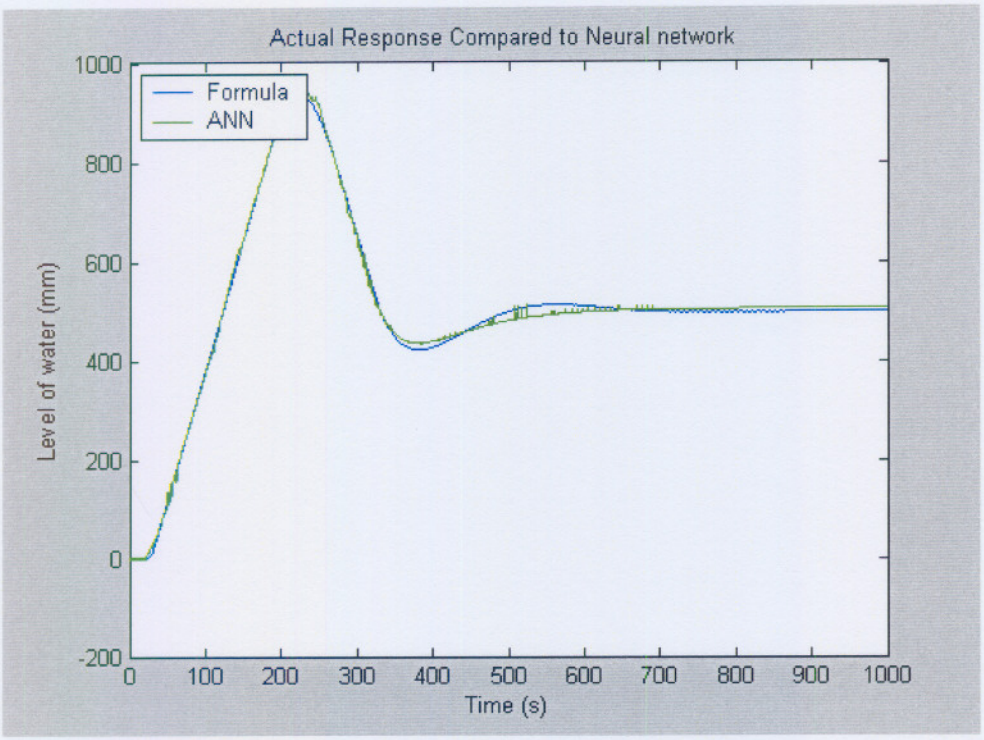

Figure 4.3.2.2.1: Memorisation of neural network

Figure 4.3.2.2.2 shows a response of a neural network compared to the Simulink® response. One can see that the neural network correctly maps the Simulink® model. This shows that small faults can be easily detected. Figure 4.3.2.2.3 show the discrepancy between the actual response and the neural network. If you can look at the output of the actual model and neural network, you can see that both outputs look the same. However the discrepancy between the two models is not zero.

The generation of residuals needs to be followed by residual evaluation, in order to arrive at detection and isolation decisions. Because of the presence of noise and model errors, the residuals are never zero, even if there is no fault. Therefore the detection decision requires testing the residuals against thresholds, obtained empirically or by theoretical considerations.

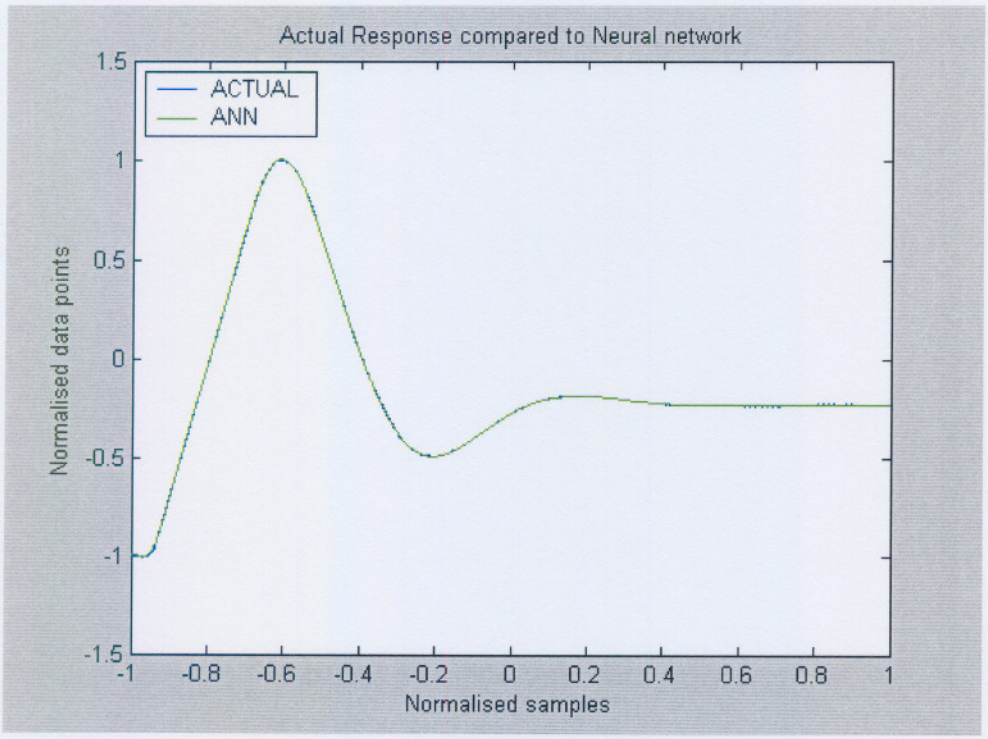

Figure 4.3.2.2.2: Actual response compared to neural network

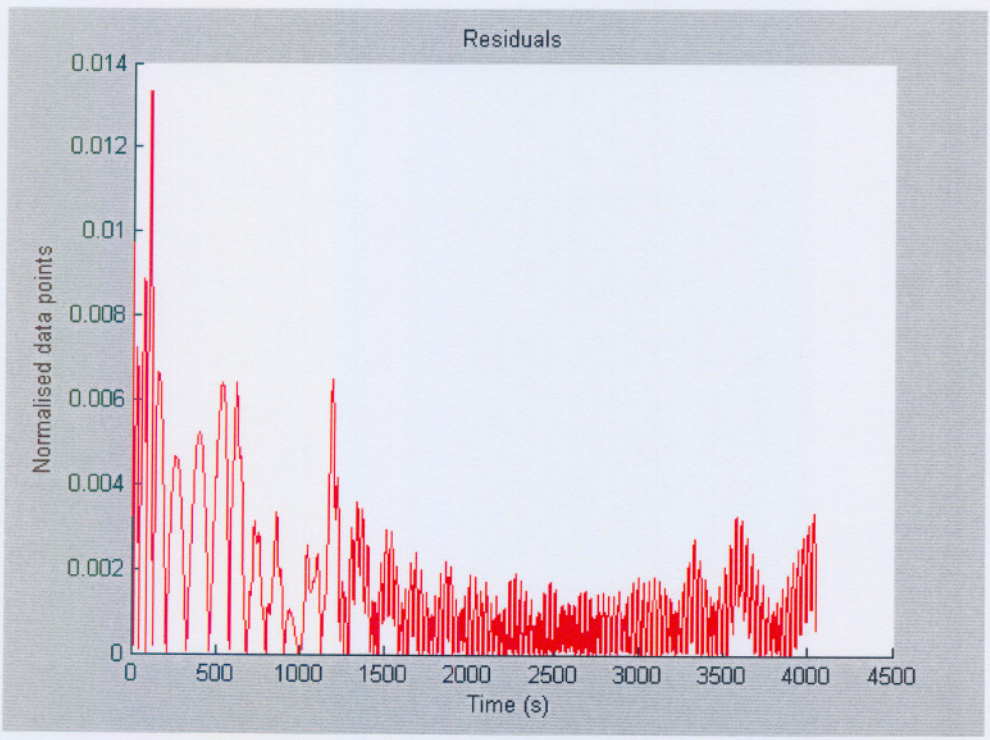

Figure 4.3.2.2.3: Residuals of the actual response and neural network

4.4 Fault diagnosis

4.4.1 Introduction

In the previous chapter a model-based FID on a system of four cascaded transfer functions was shown. This section uses the same procedure to diagnose faults on the drum level control system described in this section. Typical faults to be considered are instrumental faults and faults in the actuator and valve. Typical faults that often occur on sensors are gain and offset faults. Therefore gain and offset faults will be considered again in this section.

The water level in the drum with boiling water and steam mixture is a process variable with safety relevance. Redundantly designed hydrostatic measuring systems are often used for continuous monitoring and diagnosis of the water level. Indication errors or malfunction of the hydrostatic water level measurement, caused by faults, could endanger the operating state, the process flow and the safety of the whole plant.

Due to transient processes, malfunctions in measurement or stochastic noise deviations occur between redundant measuring systems. The task of the fault diagnosis is to assign occurring deviations to the causes in order to minimise the potential of endangerment.

In order to ensure reliable operations of an industrial process and safety of the plant, it is necessary to use correct measurements from actual system inputs and outputs. This requires the use of *Fault Detection and Diagnosis* (FDD) techniques for the recognition of the failures regarding the sensors of the system under investigation (Isermann and Balle, 1997).

Recently, different methods based on analytical redundancy have been developed to detect and diagnose faults in nonlinear, time-invariant, dynamic systems and a wide variety of model-based approaches has been proposed (Nyberg, 1999).

There are different model-based approaches to the FDD problem, namely parameter identification, parity equations, methods in frequency or in state-space domain, such as diagnosis observers, and residuals (Nyberg, 1999). However each method has its own drawback. Investigation of different diagnosis methods is considered to be out of scope of this project.

All model-based methods use a model of the monitored system to produce the so-called symptom generator (residuals). If the system is not complex and can be described accurately by the mathematical model, FDD is directly performed by using a simple geometrical analysis of residuals. In real industrial systems however, the modelling uncertainty is unavoidable. The design of an effective and reliable FDD scheme should take into account the modelling uncertainty with respect to the sensitivity of the faults. Several papers addressed this problem. For example, model-based residual evaluation was proposed by (Olsson, 2002) and (Glad and Ljung, 1991).

However a theory conducted by (Nyberg, 1999) found that there are some drawbacks with model-based fault diagnosis methods. Some of the drawbacks will be highlighted in this section.

The model-based methods assume that an accurate mathematical model is available. The model-based methods use residuals as features, where the residuals are the outcomes of consistency checks between the sensed measurements of a real system (Simulink® model) and the outputs of a mathematical model (neural network). The premise is that the residuals are large in the presence of malfunctions (faults), and small in the presence of normal disturbances, noise and modelling errors. Statistical techniques are used to define thresholds to detect the presence of faults.

The same principle used in the previous chapter will be applied in this chapter. The only difference is, in chapter 3 robustness of fault isolation was not strongly considered. Induced faults where isolated using all generated residuals. This isolate faults weakly. Refer to table 3.4.3 to 3.4.6. Extra 1's in columns can result in isolating faults wrongly.

However this type of isolating faults depends on the problem in hand. In the previous section output from plant 1 was an input to plant 2, and output from plant 2 was an input to plant 3 etc. This caused faults to propagate amongst the plants. That is why all residuals were considered to isolate occurred faults. The meaning of the 1's and 0's was explained in the previous section.

4.4.2 Fault description

Controlling the level of water in the drum requires measured variables which can be used as input to the drum software object. In this case steam flow from and water flow to the drum was considered as inputs. The water flow to the drum is controlled by the valve. Any change in the valve gain results in a change in water level. Therefore the first fault to be considered is change in gain of the valve. This fault was simulated by changing the valve's gain by 5% of the actual value.

The second fault is change in valve offset. This fault was simulated by adding a constant value to the actual system. These types of faults are additive faults that occur because of actuator failure.

The actuator also affects the controlling of water level in the drum. Therefore the impact of the actuator will be considered as a third fault that affects the level of water. This type of fault was considered to be of gain type. The objective of this section is to show a few ways to detect common sensor faults and thus, enhance sensor reliability by using instrument signals to better advantage. Figure A1 in the appendix shows the location of faults on the Simulink® model.

4.4.3 Faults modelled as arbitrary fault signals

As explained, the types of fault to be considered are sensor faults which can be modelled as arbitrary fault signals. Lots of theory has been formulated in the past on how to model faults as arbitrary signals.

Faults which are modelled as arbitrary signals consist of additive faults, which are added into the actual signal. The additive faults will cause a change in residual signal
There are a number of ways to separate the faulty signal from the non-faulty signal. Fixed or adaptive thresholds may be used to determine the occurrence of faults. The difference between the fixed threshold and adaptive threshold is that since disturbances and other uncontrolled effects vary with time, the thresholds should also vary with time instead of being fixed values. However, the choice of threshold depends on the problem in hand. If you want to detect faults direct from the faulty signal, adaptive threshold is the best choice as any change in the actual signal can be easily detected. If some statistical manipulation is required it is better to use a fixed threshold. But there is a price to pay for using fixed threshold as the number of missed detections increases.

However, the robustness of the threshold is related to its residuals. The sensitivity of these models lies with the designing of residuals. The residuals should be designed to be sensitive enough to both missed detection and false alarm. If a residual is above threshold, 1 is assigned to represent the occurrence of faults. If a residual is below threshold, 0 is assigned to represent no fault. Residuals that cause false alarms normally result in operators ignoring such alarms.
Ideally, the residuals should only be affected by the faults. However, the presence of disturbances, noise and modelling errors causes the residuals to become nonzero and thus interferes with the detection of faults. Thus it should be robust in the face of these unknown inputs. Much of the effort in designing residual generators goes into achieving robust residual performance.

One of the approaches is based on generating residuals which are insensitive to uncertainty, while at the same time sensitive to faults. It was considered that utilising statistical methods is a better solution to limit the oscillating of residuals as shown in Figure 4.3.2.2.3. Therefore the mean, standard deviation, variance, and median of the residuals signal were determined. Figure 4.4.3.1 shows the residuals. One can see that introduced faults can be easily detected in these signals. These residual properties still

depict the original behaviour of the residuals shown in Figure 4.3.2.2.3, but does not oscillate too much.

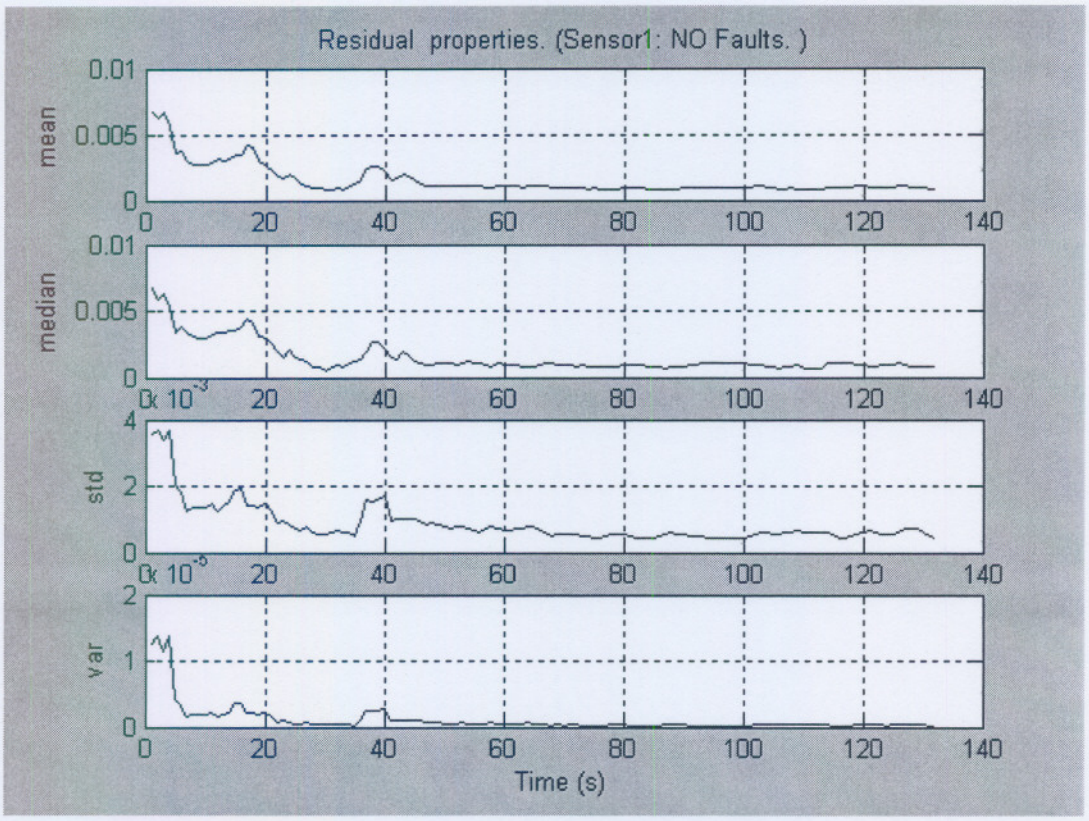

Figure 4.4.3.1: Residuals properties of Figure 4.3.2.2.3

4.4.4 Fault isolation

The interpretation of the 1's is the most important part of isolating faults. One more or less result in isolating occurred faults wrongly. This implies that it may often happen that some test quantities, that according to the residual structure should reach the thresholds, are below the threshold. The effect is serious since it can happen that the wrong fault is isolated.

To compensate for this, it is often required that the residual structure should be strongly isolating. This means that when a test quantity is not above the threshold, even though it should, there should be no other column that matches the thresholded test quantities.

If the residuals are designed to strongly isolate faults, the impact of missed detection on fixed threshold can be reduced as shown in Table 4.4.4.1. This table shows that residuals r_3 and r_4 are only sensitive to fault f_1. If f_1 is induced r_3, and r_4 will respond to the fault. This means that fault f_1 will cause a residual change in residual r_3 and r_4 only.

Considering Figure 4.4.4.1 below, one can see the impact of faults on the residuals. The fault signal shifts the residual to the positive y-axis direction. The red graph represents the non faulty signal, blue graph represents the faulty signal and green line represents the fixed threshold. This shows that the fault was correctly detected by the fixed threshold.

One can see that the induced fault affects the entire system (signal) as opposed to induced faults in chapter 3. Unlike in chapter 3, the system in chapter 4 is a closed loop system, which causes fault to propagate in the system. It is practical that if a valve fails because of blockage or leakage, water will rise or fall inside the drum. Therefore the simulated fault depicts a situation where the actuator fails because of blockage in the valve. This fault was simulated by increasing the valve gain by 5% of the original value.

We must bear in mind; however, that this is a case study, so its results should be interpreted with care. There is no guarantee that the performance ratio of tested diagnostic systems will be the same for other systems.

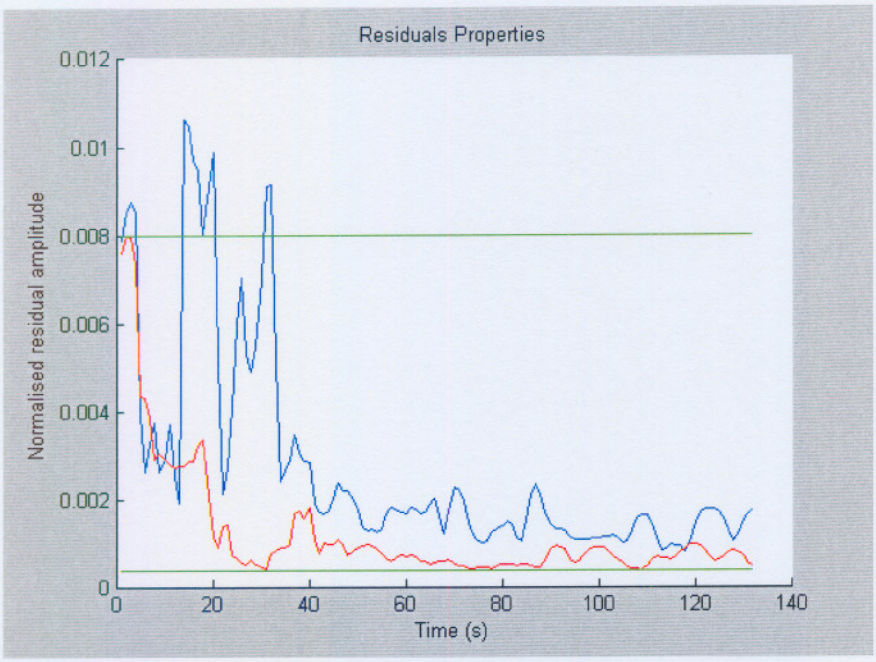

Figure 4.4.4.1: Detecting fault using fixed threshold

	nf	f_1	f_2	f_3
r_1	0	0	0	0
r_2	0	0	0	0
r_3	0	1	0	0
r_4	0	1	0	0
r_5	0	0	1	0
r_6	0	0	1	0
r_7	0	0	0	1
r_8	0	0	0	1

Table 4.4.4.1: Strongly isolability of faults

To keep the false alarm rate at a low level, the thresholds making the residuals to fire are set high (Gertler, 1991). It is therefore more likely that a residual that should fire don't, i.e. a 1 is replaced by a 0, rather than the other way around, i.e. a 0 is replaced by a 1. To avoid mis-isolation, the coding set should be constructed such that no two columns can

get identical when ones in a column are replaced by zeros. A coding set that fulfills this requirement is called a strongly isolating set. As mentioned before this method results in missed detection of faults.

The problem with a fixed threshold is that some part of the signal is ignored. Fixed thresholds are only concerned with the maximum and minimum peaks of the signal. However, the basic idea of adaptive thresholds is that since disturbances and other uncontrolled effects vary with time, the thresholds should vary with time instead of being fixed at constant values. The adaptive threshold adapts to the disturbances and therefore follows the test quantity as long as there are no faults. When the fault occurs, the residual crosses the threshold and the fault is detected. The next chapter will investigate robustness of adaptive thresholds as a means of detecting and isolating faults.

All three faults were correctly detected and isolated. The question is can this model work in a complex system like PBMM plant? The answer to this question is found in the next chapter.

4.5 Summary

The methodology for building a system model with Simulink®, and neural network was described, and then criteria for evaluating model effectiveness were given. The neural network was used to simulate the fault-free behaviour of the plant, in order to generate residuals.

Some methods on how to diagnose faults on a dynamic system were covered. Each diagnosis method has its own strength and weakness; however, one must bear in mind the consequences of isolating faults wrongly. Many researchers have different approaches to fault diagnosis. However, the bottom line is with the strongly isolability and detectability of such methods. Model-based fault diagnosis was adequate for the problem. Due to model uncertainties, the results might differ with a real plant system.

5. FLOWNEX MODEL OF THE PBMM

The goal of this chapter is to identify and diagnose faults by means of model-based fault detection and diagnosis. Simulated and experimental data will be used to identify and diagnose faults on the Pebble Bed Micro Model or PBMM plant. A neural network is used as a model to mimic the normal behaviour of the plant. This chapter integrates the methods investigated in the previous sections, to identify and diagnose faults on the overall PBMM system. Typical faults to be considered are instrumental and plant faults.

5.1 Background

Estimation of future electricity demand is an uncertain affair and Eskom's predictions of when demand will exceed supply have been revised over recent years. The PBMR is being currently developed in South Africa as a world wide international association between Eskom, the national utility, and other industrial partners to meet future electricity demands.

The PBMR, being purpose-designed for electricity generation, is inherently different from the Pressurised Water Reactor (PWR); the most common type in the world today of which the Koeberg nuclear power plant is an example. The PBMR uses helium (a gas) to cool the reactor core and drive the turbines. The fuel is based on a ceramic coating of very small enriched uranium dioxide fuel particles (silicon carbide coated particles of less than 1 mm diameter) embedded in a graphite matrix. The fuel is proof against damage up to 1 600 degrees C and will not melt below 3,500 degrees C.

The net result is a design which, if the unit is kept below a certain size, cannot exceed the temperature where fuel damage and radioactive release could occur, even with no external cooling. The plant is therefore considered inherently or "walk away" safe. This limits the size of the plant but avoids the need for highly reliable, diverse and redundant safety systems that are used to ensure adequate safety on current reactor designs (Yoshiaki, 2001).

Lots of researches have been conducted about the safety and operation of the PBMR plant by various organisations. To gain a better understanding of a Brayton power conversion cycle using three separate shafts, a prototype model called the *Pebble Bed Micro Model* (PBMM) was build. Note that another configuration will probably be built as a demonstration plant.

The project recently achieved a major engineering milestone with the successful starting up of a test rig of the proposed PBMR power conversion system. The test rig represents

the first closed-cycle, multi-shaft gas turbine in the world. The model was designed and built by the Faculty of Engineering at North West University, Potchefstroom Campus near Johannesburg, with technical input from the PBMR project team (Ferreira, 2003).

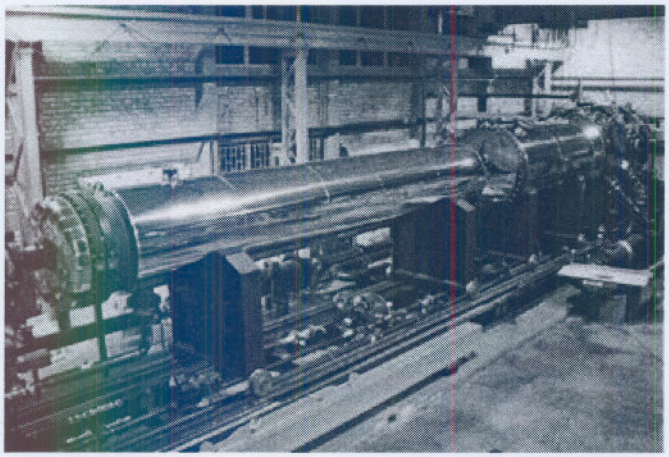

Figure 5.1.1 PBMM plant (Ferreira, 2003)

5.2 PBMR Power conversion cycle

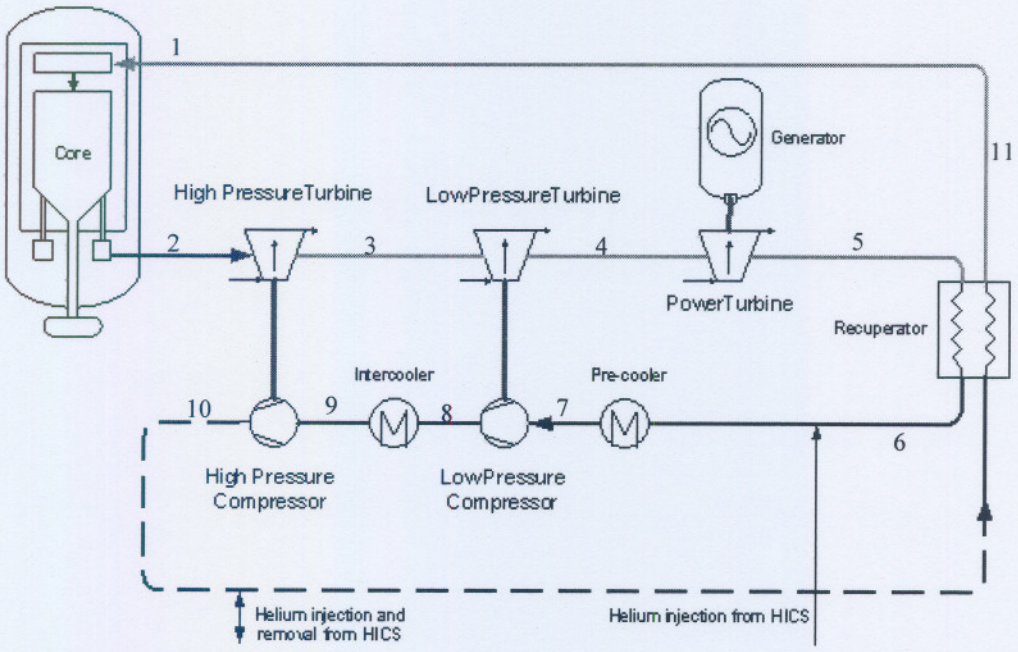

Figure 5.2.1: Proposed Pebble Bed Modular Reactor Schematic layout (Gee, 2002)

The schematic layout shown in Figure 5.2.1 depicts a Brayton power conversion cycle using three separate shafts and using helium as working fluid.

Helium enters the reactor at 500 degrees Celsius (at point1), and at a pressure of about 8.4 MPa. It leaves the reactor at about 900 degrees Celsius and drives the high pressure turbine. The high pressure turbine drives the high pressure compressor. After the high pressure turbine, the helium flows through the low pressure turbine that drives the low pressure compressor. While still hot, the helium leaves the low pressure turbine and drives the power turbine to produce the electricity through the generator.

The helium leaves the power turbine and is cooled in the recuperator. Return helium is then compressed back to a pressure of 8.5 MPa while it returns through the pre-cooler, low pressure compressor, inter-cooler and high pressure compressor. The coolers increase the efficiency of the compressors since they increase the density of the helium. The helium has also been cooled back down to 500 degrees Celsius and the cycle repeats itself as it travels back to the reactor (Gee, 2002).

The PBMR plant is still under development. In order to gain a better understanding of how the power conversion of the PBMR operates, a functional model called the PBMM was designed. The PBMM will have the same control topology as the PBMR.

Since the objectives of the PBMM is not to address issues related to the use of helium as the working fluid or to test the performance of individual components such as compressor, turbines, or heat exchangers, it was decided to use nitrogen instead of helium as working fluid (Greyvenstein and Rousseau, 2003).

The PBMM further differs from the PBMR in terms of the nuclear reactor as opposed to the electrical resistance heater used by PBMM. There are other differences between the PBMM and the PBMR, for more information refer to (Greyvenstein and Rousseau, 2003).

5.3 PBMM

The PBMM was build and tested by the Faculty of engineering at the North West University, Potchefstroom Campus. The working of the model was also simulated using Flownex software. The PBMM uses cheap off-the-shelf single stage centrifugal compressors and turbines rather than axial flow machines. The performance characteristics of centrifugal machines closely resemble that of axial flow machines and it will, therefore, suffice for purpose of this project (Greyvenstein and Rousseau, 2003).

Figure 5.3.1 below depicts a schematic layout of the PBMM recuperative Brayton cycle. Note that in the PBMM plant, the generator shown in Figure 5.2.1 is emulated by the load compressor connected to a power dissipation loop consisting of a flow control valve and a heat exchanger. The nitrogen gas is injected at point 1, just after the pre-cooler. The operation of the Brayton cycle in the PBMM plant is the same as the PBMR.

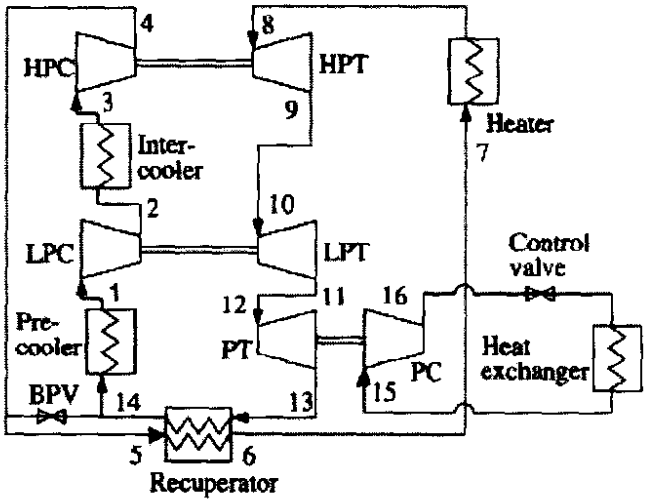

Figure 5.3.1: Schematic layout of the PBMM recuperative Brayton cycle (Greyvenstein and Rousseau, 2003)

The working of Flownex software was validated and verified using experimental data as well as other software's. In the past comparisons between experimental and simulated results of the PBMM were done. The Flownex model of the PBMM agreed with the experimental data.

Since the Flownex model have the same overall performance characteristics and control topology as that of the PBMM it was decided to use Flownex for simulation. Simulation software such as Flownex allows user to design a complex plant like the PBMM in a relatively short period of time at low cost.

Figure 5.3.2 below depicts a schematic layout of PBMM plant with the location of the test faults. The system will be modelled as single-input single-output sections. The data collected on the input and output of each system will be used to train a neural network.

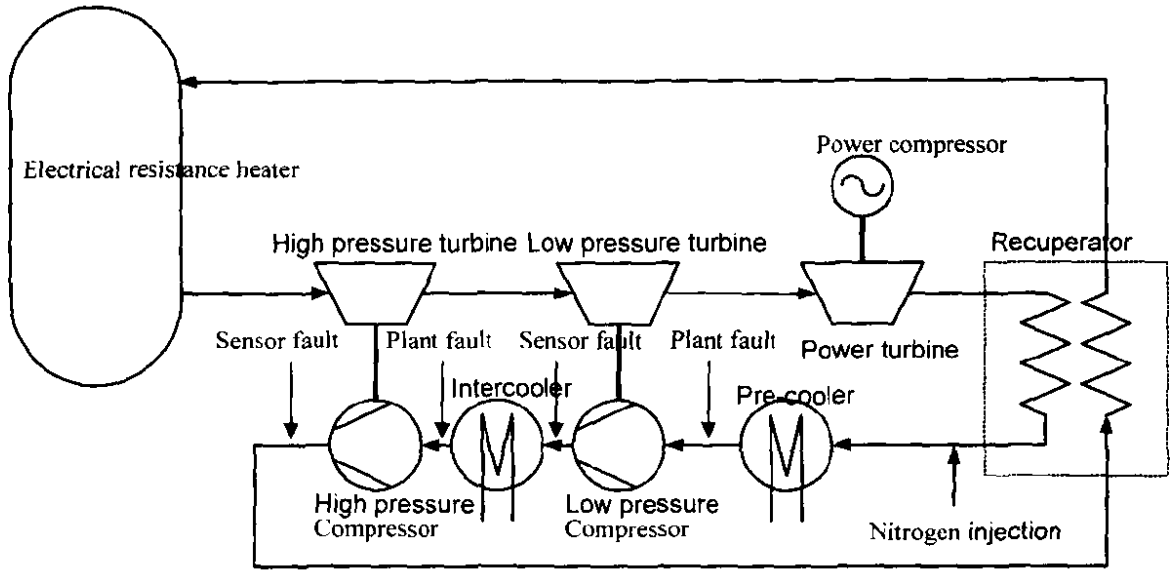

Figure 5.3.2: Schematic layout of PBMM with simulated fault locations

5.4 Modelling the PBMM using Flownex

Flownex is a thermal fluid simulation software package that has the ability to simulate the steady state and transient operation of the integrated system, making use of the performance characteristics of the individual components.

Model M400 of the PBMM is a model built in Flownex to simulate the behaviour of the PBMM. The model consists of all the power conversion cycle of the Brayton cycle.

The intention of this project is not to design or test the working of Flownex software, but to use Flownex to simulate the dynamic behaviour of plant, therefore description of Flownex software is considered to be out of the scope of this project. Flownex software has a Simulink-Flownet interface for external control. The operation of the PBMM model in Flownex can be triggered in Simulink using the link shown in Figure 5.4.1 below. The control valve in Figure 5.4.1 is used to control the amount of nitrogen to be injected.

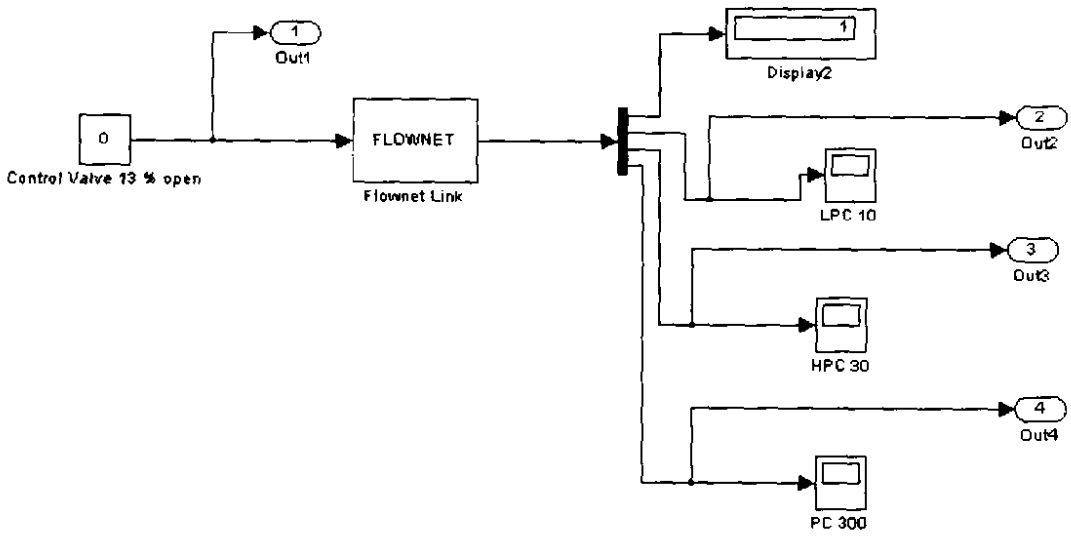

Figure 5.4.1: Flownex -Simulink interface

Power output of the system can be controlled by increasing or decreasing the mass inventory in the system. This can be done by injecting or extracting the nitrogen gas. The nitrogen gas is injected just before the pre-cooler to minimise the amount of energy for injection. The nitrogen gas is extracted at point of high pressure in order to ease the extraction. The inventory decrease (nitrogen extraction) is used to lower the power output of the generator.

Figure 5.4.2 below shows an example of the control of injection of nitrogen into the system. The injection from the nitrogen inventory control system tanks is controlled by the control valve which controls the flow of nitrogen gas into the system. If the injection of nitrogen gas is increased the pressure in the system also increases. The nitrogen gas must not be injected too fast in the system, as the system can shut down.

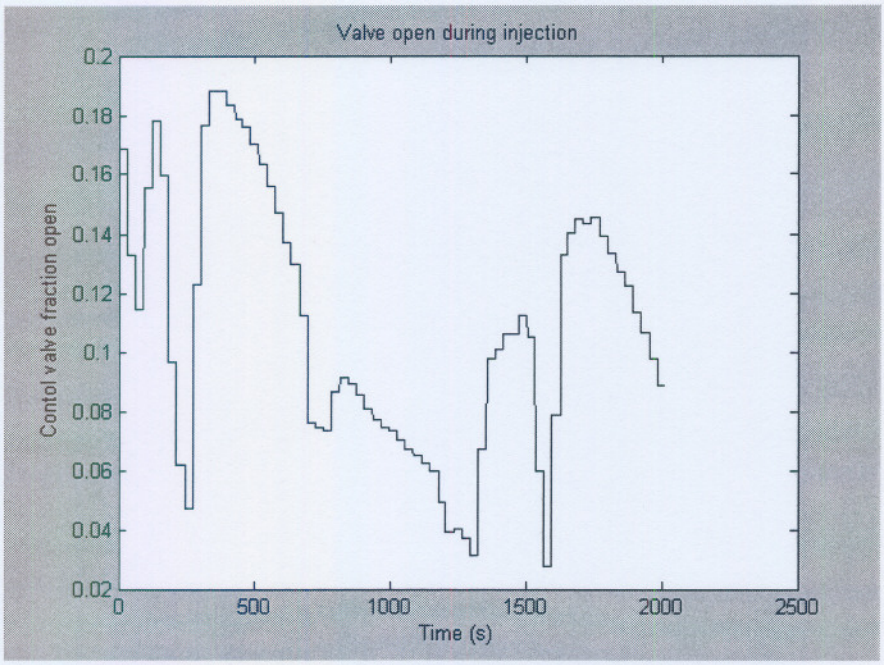

Figure 5.4.2: Valve opening during injection of nitrogen gas

Thus, the first part to concentrate on is the modelling of pressure variation across the low pressure compressor (LPC) and high pressure compressor (HPC). Later in the chapter modelling of the inter-cooler (IC) and pre-cooler (PC) will be considered. Note that for modelling of IC and PC only simulated data will be used.

5.5 Modelling the LPC and HPC

For this study, both simulated and experimental data will be used to train the neural network. The pressure in the inlet of both LPC, and HPC will be used as input to neural network. The pressure in the outlet of both LPC, and HPC will be used as target.

The LPC is represented by element 10, and the HPC is represented by element 30 in the Flownex model shown in Figure 5.5.1 below. The data to train the network will be collected at element 104 (input) and element 12 (output) for the LPC. For the HPC the data will be collected at element 22 and element 32 as shown in Figure 5.5.1 below.

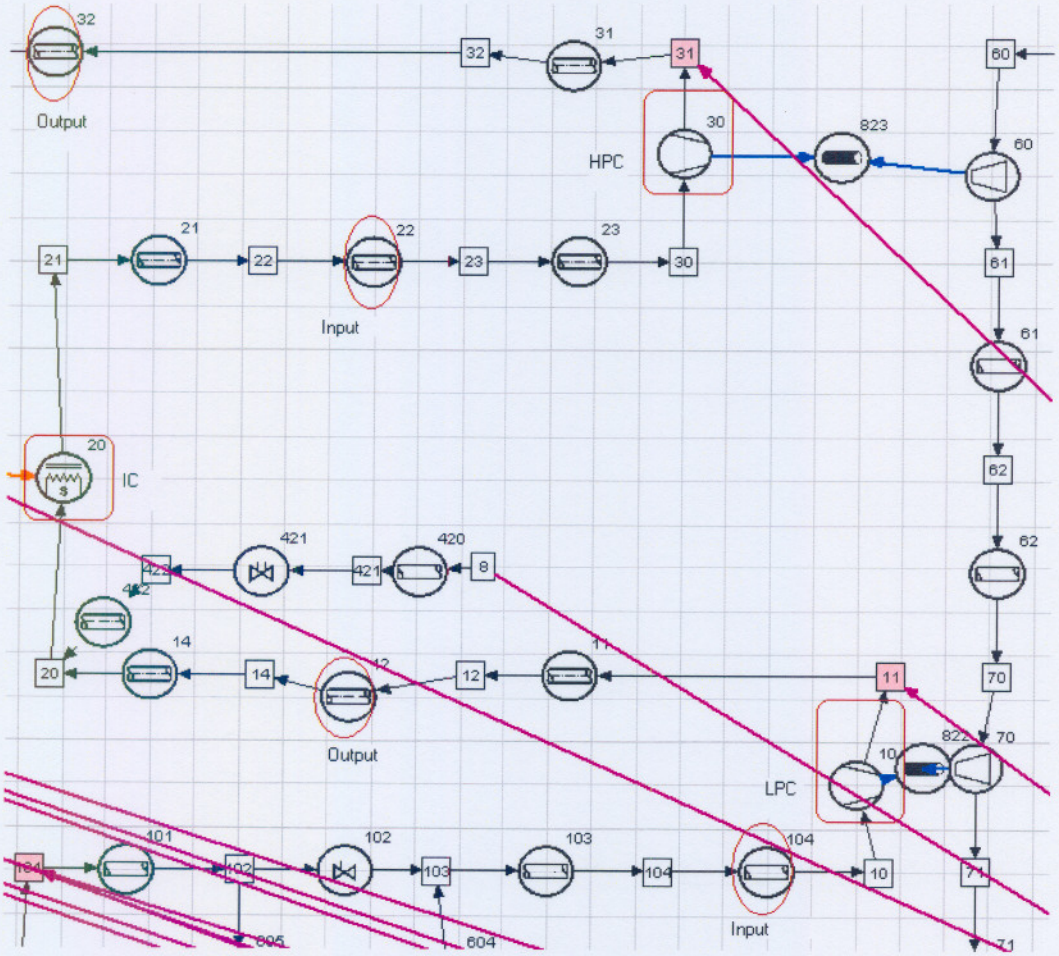

Figure 5.5.1: Part of Flownex model schematic layout

Figure 5.5.2 shows both input and output pressure generated by Flownex. Figure 5.5.3 shows both input and output pressure generated experimentally when running the PBMM plant. Note that the intention is not to compare the Flownex model with the experimental data. The aim is to show that the dynamic behaviour of PBMM can be modelled using a neural network as modelling tool for both simulated and experimental data.

For fault diagnosis on the LPC and HPC, only the experimental data will be used, because using simulated data will not contribute anything new.

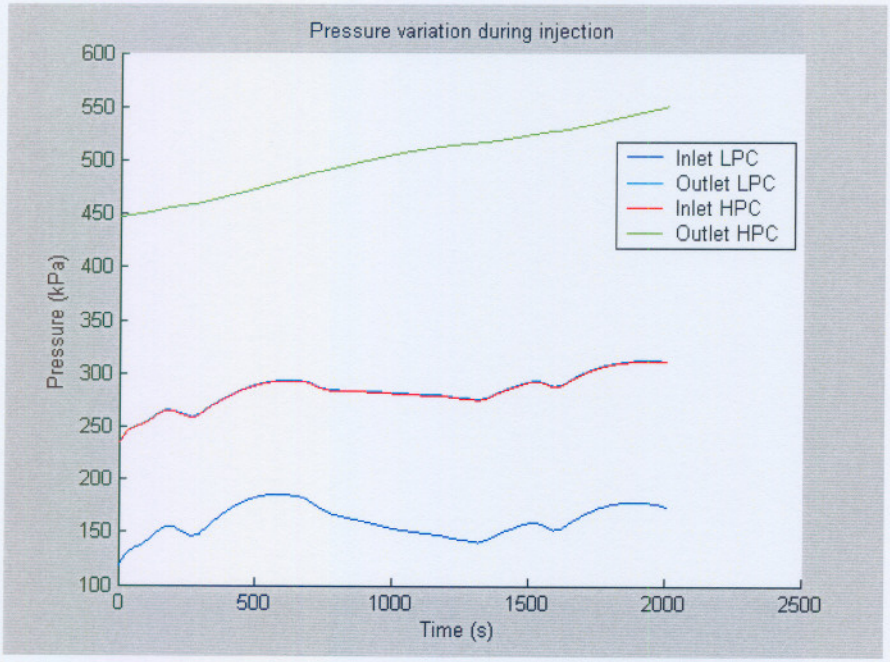

Figure 5.5.2: Flownex model of pressure variation during injection

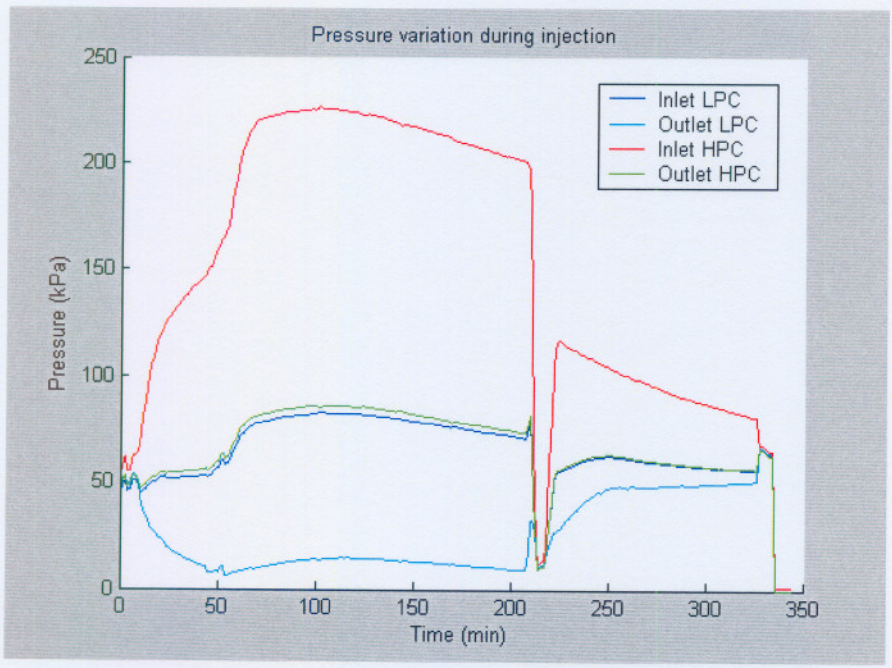

Figure 5.5.3: Experimental model of pressure variation during injection

5.6 Modelling the IC and PC

Figure 5.6.1 below shows pressure variation on the PC and IC during injection. The generated data was used to diagnose plant faults on the PC and IC. The typical plant faults that were considered are plugged process line and heat exchanger fouling. These types of faults are due to the decrease in available heat transfer area, and decreasing in the overall heat transfer coefficient respectively.

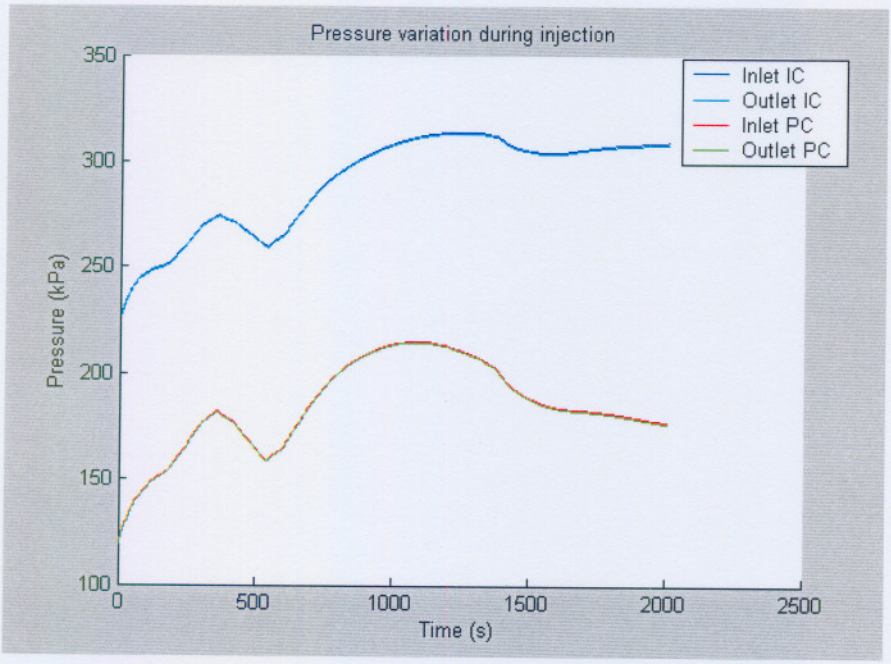

Figure 5.6.1: Flownex model of pressure variation during injection

5.7 Simulation and fault diagnosis

A model-based diagnosis system starts its reasoning from a model. The model represents in an explicit way the correct behaviour of the system to be diagnosed. If behaviour of the observed situation is different from the estimation carried out by the model of the same situation, the system concludes that there is a fault. Finding the causes of faults in the system is the most challenging part of diagnosing faults.

A model of the system to be diagnosed is used to make model-based diagnosis; the model can be well structured according to physical laws or can be made from human experience and data from the process, or combination of both. Model-based diagnosis is based on comparing observations of behaviour and the predictions from a model of the process. Model-based diagnosis depends on the model to make diagnosis.

The parameters that describe the behaviour of a large class of man-made systems such as PBMR plant are continuous and time-varying, typically modelled by a set of non-linear differential equations that relate outputs, inputs and system parameters. Analytic solution methods exist if equations are linear. For complex and non-linear equations, numerical techniques may be applied but the solution methods are computationally complex and there is no guarantee that they will converge.

One way of detecting faults in such complex plant is to have a set of sensors that will monitor the system behaviour. The deviation from normal operation of the sensors will indicate the occurrence of faults. The other problem in diagnosis is to isolate the occurred faults accurately. This can be done by studying the behaviour of the faults. Each occurred fault has its own characteristics. The method used to study the behaviour of faults should be reliable as the risk of isolating faults wrongly lies with the accuracy of the method.

In the previous chapters, case studies on how to identify and diagnose faults using model-based diagnosis were done. The intention of this chapter is to combine the ideas accumulated in the previous sections.

This study focused on analysing data generated by a Flownex simulation model, and experimental data generated when running the PBMM plant. The typical types of faults to be considered are instrumentation and plant faults. The question is how to simulate such faults? The faults can be simulated directly from the Flownex model, especially the plant faults. Plant faults will be simulated directly from the Flownex model (description covered later in the chapter). The sensor faults are simulated by adding a constant value

on the Flownex simulated data and experimental data. For location of faults see the attached schematic in the Appendix.

Faults were added to simulated data from the Flownex model and experimental data obtained from the PBMM plant. A constant value was added to normal data to observe changes between neural network and Flownex model. Since the neural network learn by example, if a fault is induced at a specific time the neural network will react to the fault. This will cause deviation between the two outputs.

As explained previously, the deviation between the two models is the key element to evaluate faults. The neural network model needs to be accurate in order to detect small faults.

5.7.1 Neural network model

Research has been done in the past on modelling PBMR plant using different simulation methods. A study conducted by (Strydom, 2004), find that neural networks can be used to model the dynamic behaviour of the PBMR.

A study conducted by (Strydom, 2004) compares different neural network topologies to model the HPC and LPC of PBMR. This minimised the searching of the best neural network topology, therefore the part to concentrate on is to design an accurate neural network model. Comparison of different neural network topologies will mean reiteration of the study conducted by (Strydom, 2004).

To gain a better understanding of the working of the PBMM plant, data was first simulated using Flownex, with the intention of finalising the test with experimental data.

57.2 Training the chosen neural network

One thing to consider when designing a neural network is the number of layers to be used. Normally the choice of number of layers depends on the problem in hand. MLP with one hidden layer was shown to be sufficient in the previous section. A feed-forward MLP with three layers (input, hidden, and output) will be considered again in this chapter.

When training a neural network (learning process) the examples must be selected carefully, otherwise useful time is wasted or even worse the network might not function properly. The problem is that it is very difficult to diagnose erroneous behaviour even for experienced analysts.

However, and despite the difficulties in understanding how they work, neural networks are widely used in pattern recognition because of their ability to generalise and to respond well to novel patterns.

The general concept is the following: During the training session neurons are taught to recognise specific patterns. Therefore the number of neurons plays a vital role in the performance of the network. The problem of selecting neurons is that there is no standard method to select the number of neurons to be used for a specific problem. Normally it is done by means of a trial and error method. In most situations, there is no way to determine the best number of hidden neurons without training several networks and estimate the generalisation error as shown on the table below.

If you have too few units, you will get high training error and high generalisation error due to under-fitting and high statistical bias. On the other hand, the training error can be made as small as desired by adding more neurons, but generally each additional unit will produce less and less benefit.

Starting with a single hidden neuron and increasing the number of neurons one at a time ensures that the smallest number of neurons is identified. However, the number of iterations can often be reduced by increasing the number of neurons by a small number

greater than one to get an approximate idea of how many neurons will be required, then refining this estimate.

Table 5.7.2.1 below shows the performance of neural network when increasing the number of hidden neurons. It was found that the increase in number of hidden neurons makes the network to perform poor as opposed to the increase in number of input neurons.

It is not only the neurons that affect the performance of the neural network, the time delay on input and hidden layers also has impact on network performance. Experiments have shown that the network perform well with only the input delay. The inclusion of a hidden delay makes the network to take too much time to train, or that it performs poorly.

Training algorithm	Neurons	Training Epoch	MSE
trainrp	30 1 1	500	0.0006
trainrp	**30 5 1**	**500**	**0.0003**
trainrp	30 10 1	500	0.0005
traingdx	30 1 1	500	0.01
traingdx	30 5 1	500	0.02
traingdx	30 10 1	500	0.02
trainoss	30 1 1	500	0.014
trainoss	30 5 1	500	0.003
trainoss	30 10 1	500	0.005

Table 5.7.2.1: Comparisons of neural network training functions

The trainrp trains the best in terms of the *Performance Measure* (MSE). The MSE between the actual output and the neural network is small as compared to the other two training algorithms.

The input layer delay was fixed at 200 delays. Figure 5.7.2.1 below shows the actual response compared to a neural network with both input and hidden layer delays. One can see that the neural network failed to learn the dynamic behaviour of the system. The

delay at the input was 200, and the hidden layer delay was 10 delays. The number of hidden delays was increased to be the same as the input delays. The network performed better, but it took too much time to converge. It shows that for the network to perform, the number of input delays must be equal to the number of hidden layer delays. Since the network performs better and fast with input delay only, the hidden layer delay was not further used in the experiment.

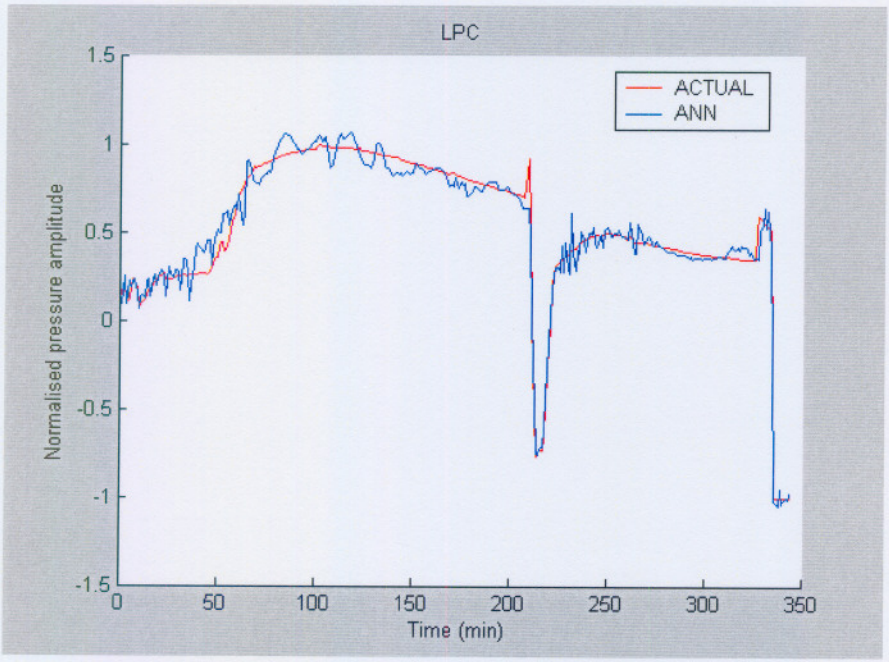

Figure 5.7.2.1: Modelling capabilities of NN with both input and hidden layer delay

The same experiment was repeated with input delays only, the network performed well, and the neural network learnt the system behaviour accurate as shown in Figure 5.7.2.2. The same experiment was repeated using trainbfg, but the network ran out of memory. This shows that trainbfg failed to train the data completely.

The same procedure (using a different training algorithm) was applied on the IC and PC simulated data. It was found that trainbfg with no input and hidden layer delay train the best in terms of the MSE. It shows that the training methods depend on the types of data presented to it.

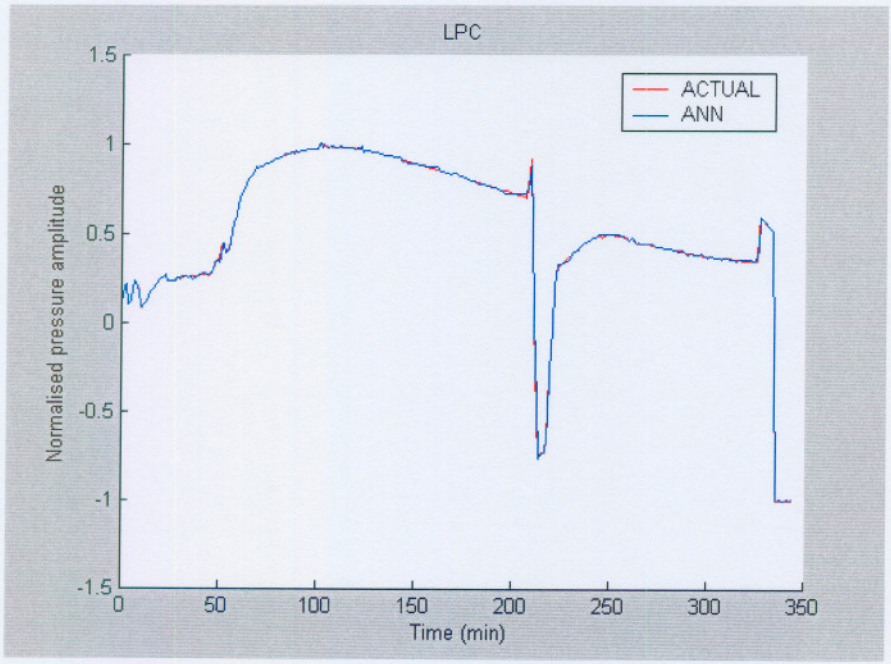

Figure 5.7.2.2: Modelling capabilities of NN with input delay

5.7.3 Fault identification and diagnosis

The neural network performed well on both simulated and experimental data. Most of the model-based fault detection and diagnosis methods rely on the concept of analytical redundancy. In contrast to physical redundancy, when measurements from parallel sensors are compared to each other, sensors measurements are compared to analytically computed values of the respective variable. Such computations use present and/or previous measurements of other variables, and the mathematical plant model describing their nominal relationship to the measured variable. The idea can be extended to the comparison of two analytically generated quantities, obtained from sets of variables. In either case, the resulting differences, called residuals, are indicative of the presence of faults in the system. The generation of residuals needs to be followed by residual evaluation, in order to arrive at detection and isolation decisions. Because of the presence of noise and model errors, the residuals are never zero, even if there is no fault. Therefore the detection decision requires testing the residuals against thresholds, obtained

empirically or by theoretical considerations. To facilitate fault isolation, the residual generators are usually designed for isolation enhanced residuals, exhibiting structural or directional properties (Tako, 2001).

Figure 5.7.3.1 below shows the impact of adaptive threshold. One can see that despite the normal operation of the plant the residual generated by the neural network and the actual plant is not zero.

A fault of 5% was introduced just after 5 seconds of normal operation. The fault was correctly detected by the threshold as the faulty signal deviated for 5 seconds from normal operation, and returned to normal operation as shown in Figure 5.7.3.2.

The robustness of a fault detection system means that it must be only sensitive to faults, even in the presence of model-reality differences. One of the approaches is based on generating residuals which are insensitive to uncertainty, while at the same time sensitive to faults (Tako, 2001).

The simulated data from the output of LPC and HPC was used to model sensor behaviour. The experiments show that in the absence of an actual model, simulation by a neural network can be used to detect faulty behaviour of plant and sensors.

This study was concluded with simulation of plant faults using output data from IC and PC. First a system response was recorded for a normal operation scenario. Then a physical fault was imposed on the IC and PC. A fault of 20% was introduced into system which resembles a decrease in the available heat transfer area. The other fault resembles a decrease of overall heat transfer coefficient which is typical of fouling in feed water.

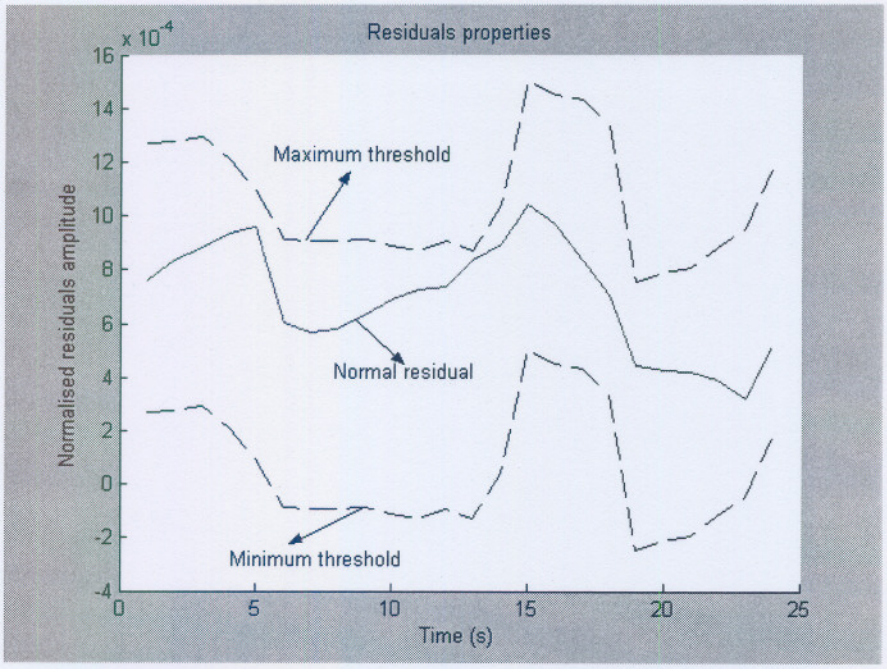

Figure 5.7.3.1: Impact of adaptive threshold

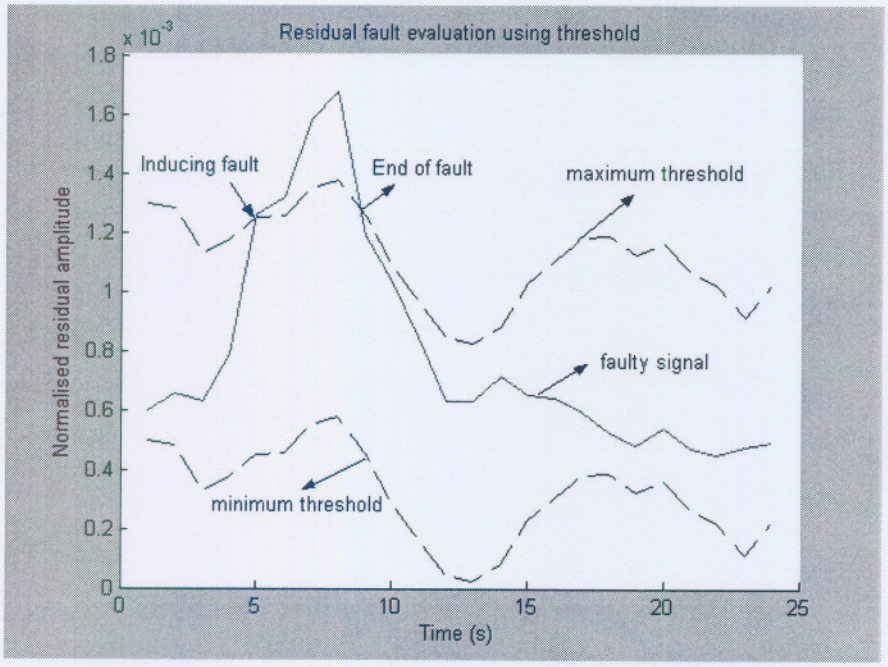

Figure 5.7.3.2: Detection of faults using adaptive threshold

Figure 5.7.3.3 below shows the setting of adaptive threshold to detected plant fault. One can see that the threshold adapt to signal changes. Figure 5.7.3.4 depicts a detection of plant fault in the PC (see Figure A5 in the appendix for location of the fault in the Flownex model). Unlike the sensor fault (Figure 5.7.3.2 above); the faulty signal does not return to normal operation. It is because the plant fault causes the entire system to deviate from normal operation.

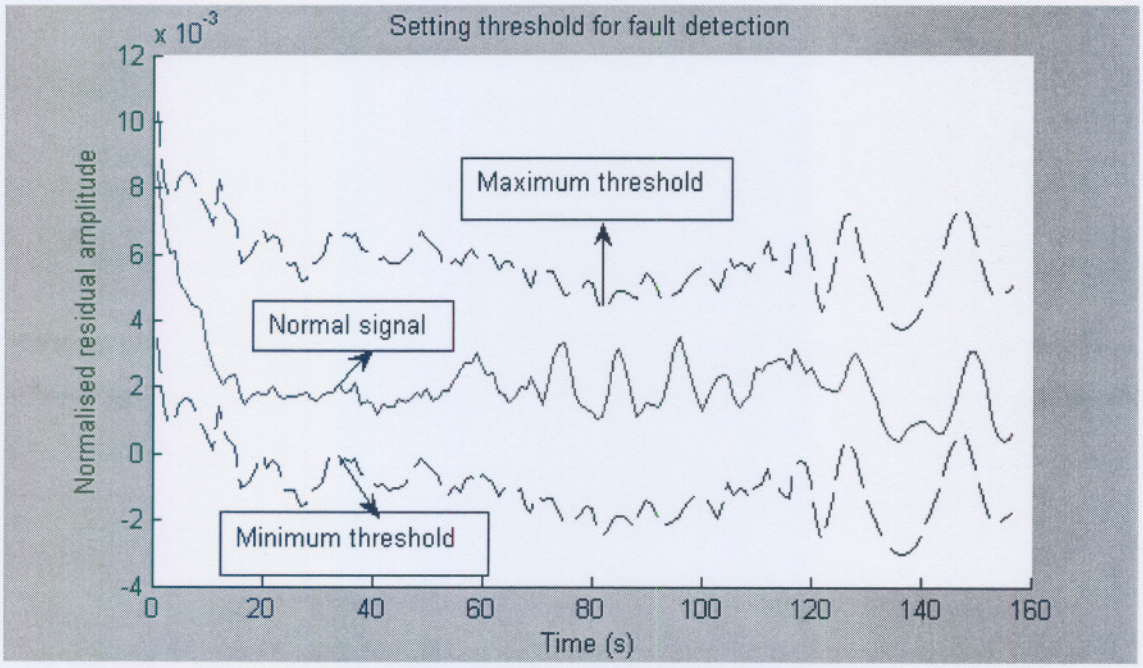

Figure 5.7.3.3: Setting threshold for detection of plant fault

All induced faults were correctly diagnosed. In conclusion it shows that model-based diagnosis can be used to detect and diagnose faults. One must bear in mind that to implement such ideas on a real system will need some fine tuning. This simulation results need to be interpreted with care. Finally, it shows that when accurate simulation software is available, model-based diagnosis can be used to generate fault scenarios without affecting plant operation or causing physical damage to the plant.

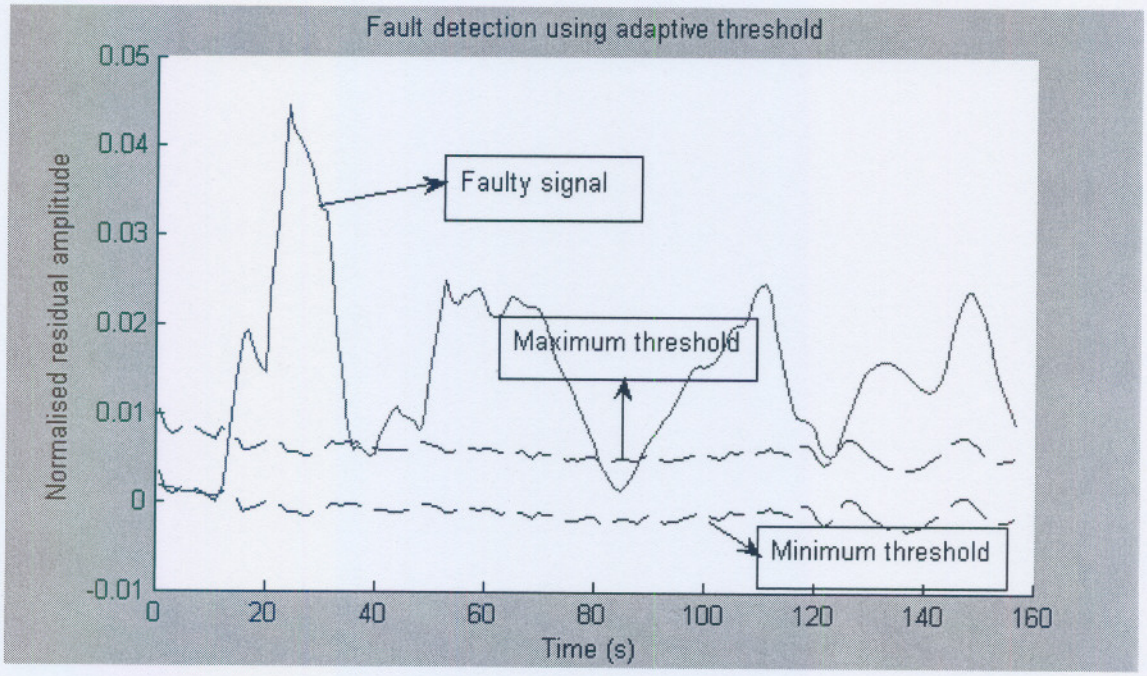

Figure 5.7.3.4: Detection of plant fault using adaptive threshold

5.8 Summary

The methodology for building a system model with Flownex, and neural networks, was described, and then criteria for evaluating model effectiveness were given. The neural network was mainly used to simulate the fault-free behaviour of the plant, in order to generate residuals. Experiments on how to simulate and diagnose typical sensors and plant faults have been covered in this section. It shows that model based diagnosis can be used to diagnose plant and sensor faults.

6. CONCLUSION

Synopses of the experimental results are given. The contributions of the research and areas for improvement in the study are stated. Some suggestions for future research are given.

6.1 Summary of experimental results

In this study, model-based diagnosis was investigated. Neural networks were used as models to mimic the normal behaviour of plant.

Experiments were conducted on various simulated plants. The knowledge accumulated on the benchmark model was used to diagnose faults on the PBMM plant. The model detected induced faults ranging from 5%-20% of the simulated data accurately.

Experimental results have shown that neural networks are capable of modelling the dynamic behaviour of the non-linear problems used in this study. Experiments have shown that different data patterns respond differently in terms of their accuracy measure (mean squared error). The mean squared error determines the discrepancies between the actual output and the predicted output after training the network. It is better to redesign the neural network each time you work with new pattern. Since most of the simulated faults investigated in this thesis are sensor faults, it was found that sensors failures can be diagnosed by using model-based diagnosis. Note it is not just sensors that become faulty; the plant as well may be faulty, therefore it is important to distinguish sensor faults from plant faults. The experiments were finalised by diagnosing sensor and plant faults successfully on the PBMM plant.

6.2 Contributions of the study

The multi-layer perceptron has shown to be able to model a wide variety of systems with both linear and non-linear characteristics.

The idea of model-based fault diagnosis was explained. This was done by addressing the following aspects; residual generation, residual evaluation, robustness concerning model uncertainty, and performance issues. It was illustrated how the measurement noise and possible disturbances affect the residuals. The impact of threshold on fault detection was illustrated using fixed and adaptive thresholds. In case of a system which is subjected to

model uncertainties adaptive threshold perform better compared to fixed threshold. The isolation of different faults type depends on the availability of fault signatures. It was shown that using decision logic can help to isolate each occurred fault uniquely.

6.3 Areas of improvements and future work

This project has demonstrated that the use of neural networks, combined with the appropriate use of preprocessing methods, is effective at modelling faults at any plant. However, additional research may be desired to implement this technique in practice.

Training neural networks using on-line data will make the results of this simulation practical. Future research should consider training of neural network on-line, which will be subjected to system disturbances due to noise and other uncertainties.

7. APPENDIX

In this appendix all schematic layouts which are too lengthy for the chapters are given.

7.1 Location of simulated faults in a drum control system

Figure A1 below shows the location of the faults in the drum control system. For the purpose of fault detection, three types of faults have been considered. These types of faults resemble typical sensor faults. The description of the faults has been covered in chapter 4 of this report. For diagnosis purposes, assumed that only one fault occurs at a time.

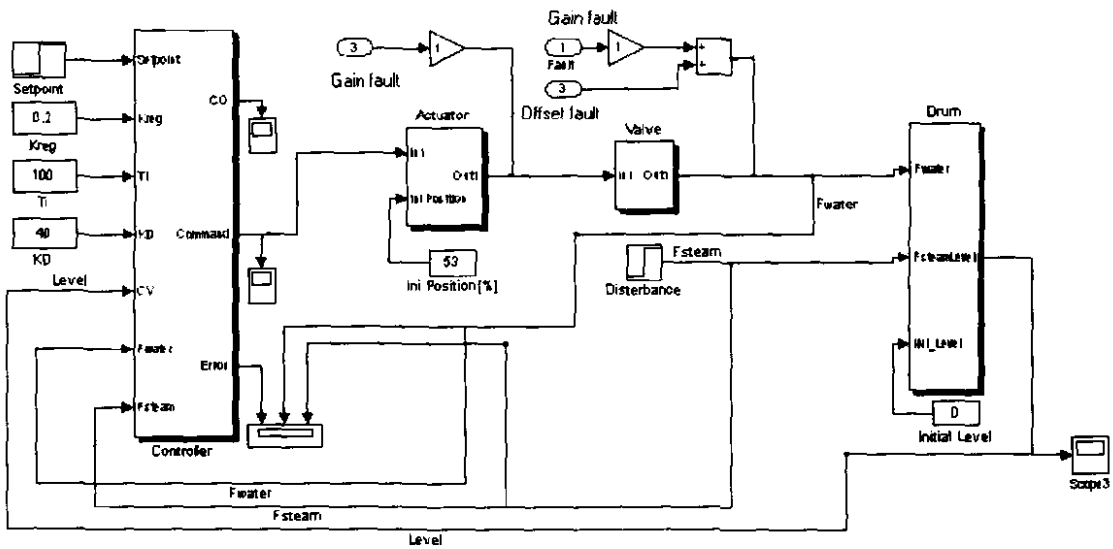

Figure A1: Schematic illustration of water level control system with the location of faults

7.2 Faults location in the PBMM plant

Figure A2 below shows a schematic layout of the PBMM plant. Note that for clarifying the diagram is zoomed. Sensor PT10 (circled red) from Figure A3 is used to capture the input data, and sensor PT11 is used collect the output data from the LPC. Those data are used to train the neural network as explained previously. Similarly, PT30 and PT31 are used to collect pressure response from the HPC (experimental data).

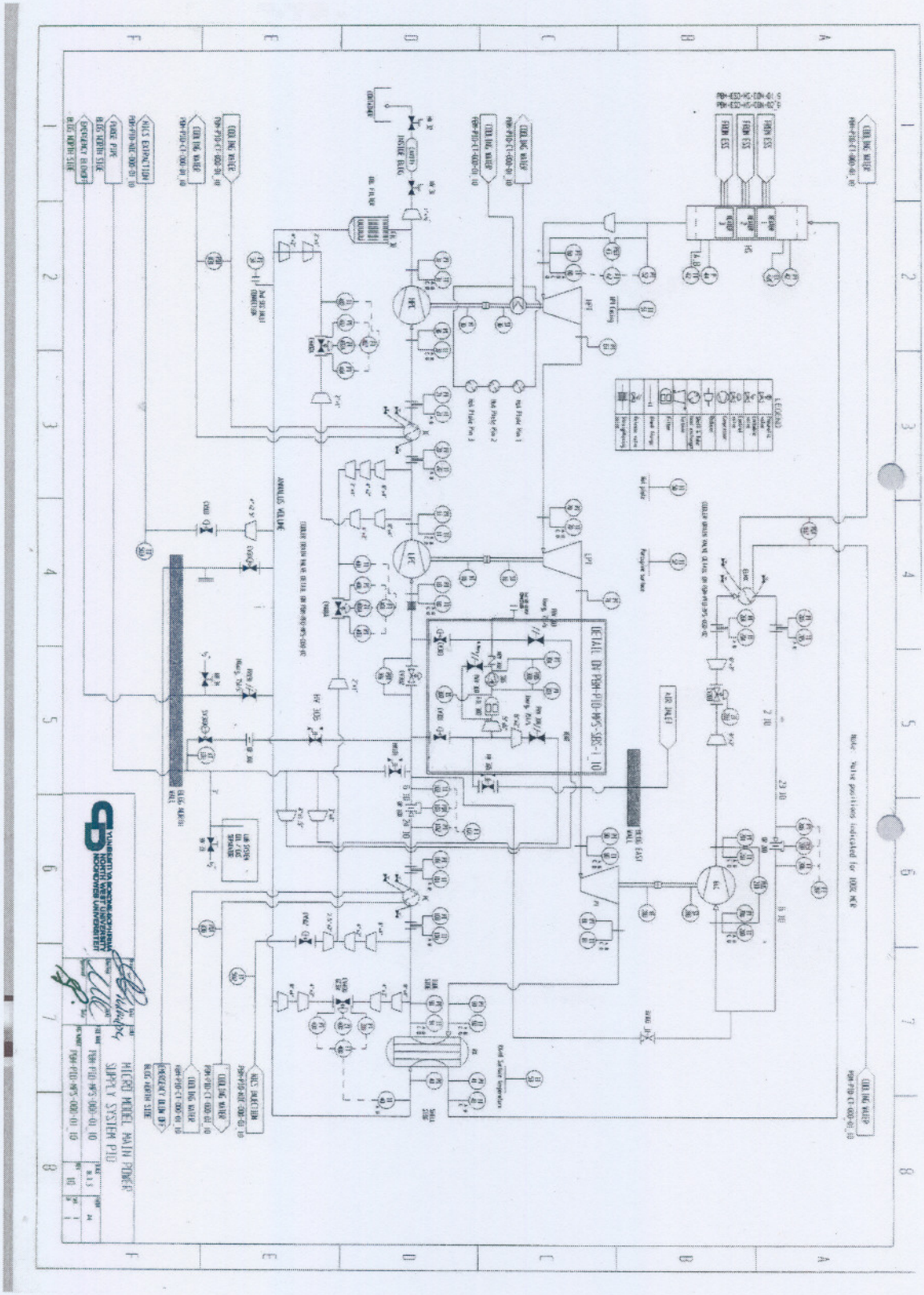

Figure A2: Schematic layout of PBMM plant with sensors for collection of data

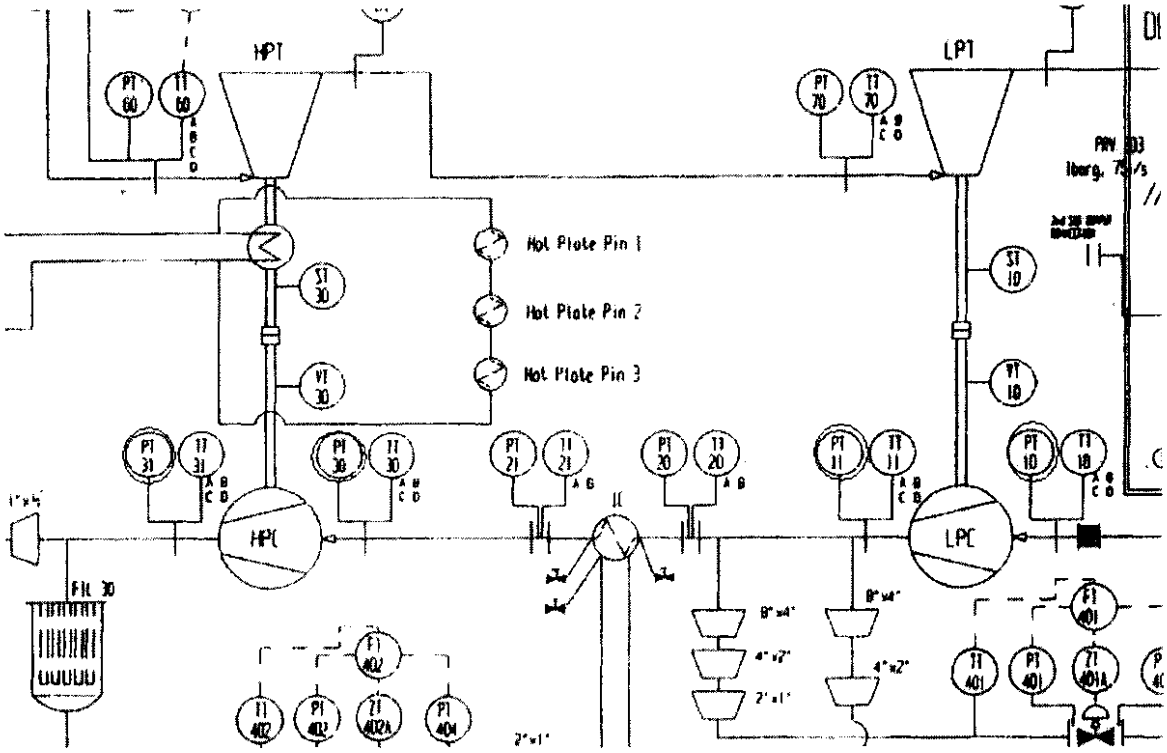

Figure A3: Zoomed schematic layout of PBMM plant

For the purpose of fault detection and diagnosis, the output from sensor PT11, and PT31 are added with a constant value 0.002. The faulty output is used to train the neural network. The same principle applied to Flownex simulated data.

In the case of plant fault the physical properties of the PC and IC were changed. First a system response was recorded for normal operation scenario. Then a physical fault was imposed on the IC and PC. A fault of 20% was introduced into system which resembles a decrease in the available heat transfer area. The other fault resembles a decrease of overall heat transfer coefficient which is typical of fouling in feed water. Elements 91 and 101 were used to collect data for PC as shown in the Figure A5. Elements 14 and 21 were used to collect input and output data from IC as shown in Figure 4 below.

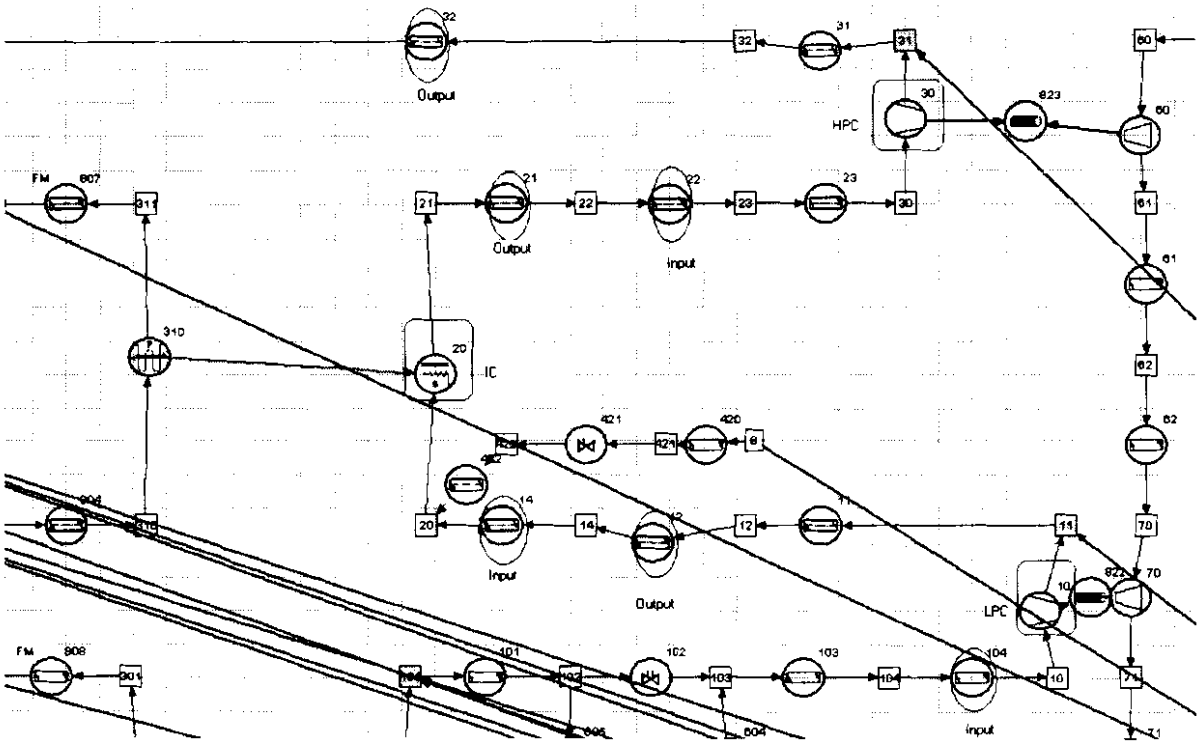

Figure A4: Part of Flownex Model showing the collection of input and output data

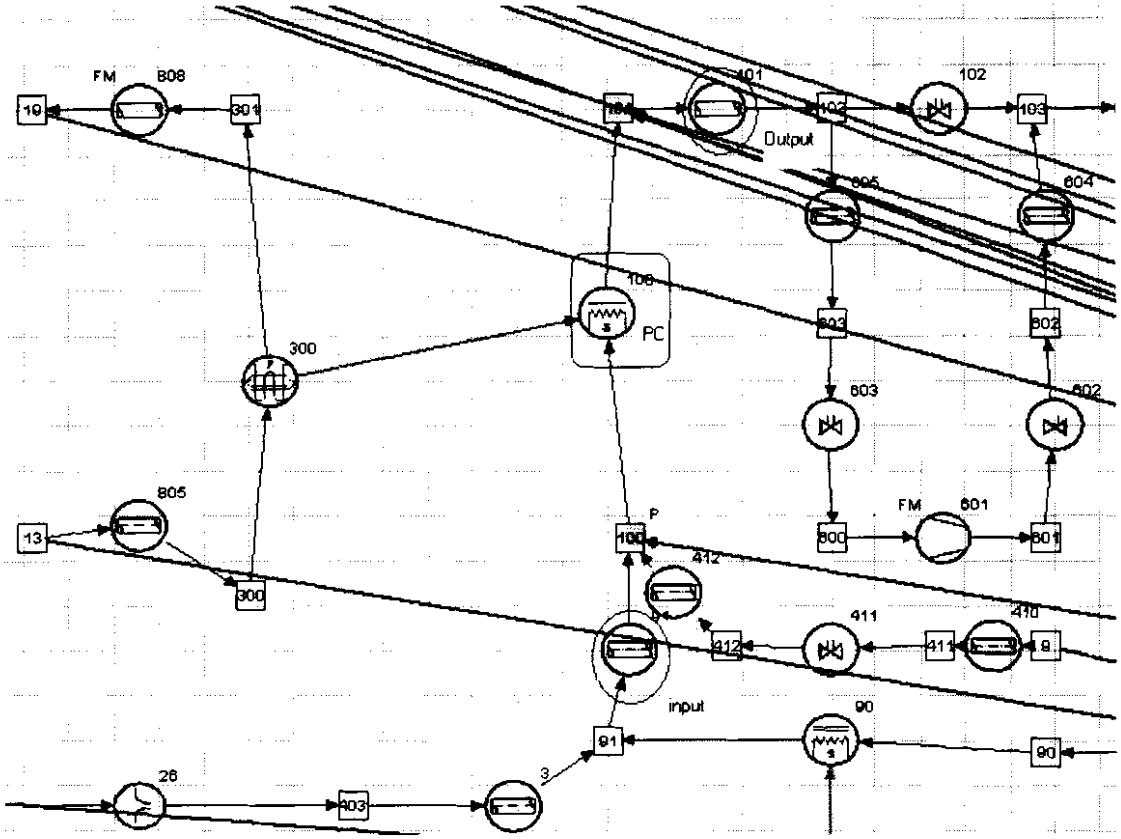

Figure A5: Part of Flownex Model showing the collection of input and output data

7.4 Source code included in the compact disc

7.4.1 Matlab Code

The Matlab code that was used for this Project is included in the attached Disc.

Chapter 3 Source code

- Single-input multi-output network: fid_4rd_simo.m
- Multi-input multi-output network: fid_4rd_mimo.m

Chapter 4 Source code

- Water_level.m

Chapter 5 Source code

- PBMM Flownex simulated model: plant_fault.m
- PBMM Experimental results: instrumental_fault.m

7.4.2 Data for training neural networks

7.4.2.1 Generated data from the Simulink model (Drum level control)

- Output.txt
- Input.txt
- Input_valvegain.txt
- Output_valvegain.txt
- Input_offsetvalve.txt
- Output_offsetvalve.txt
- Input_actuatorgain.txt
- Output_actuatorgain.txt

7.4.2.2 Generated data from Pre-cooler and Inter-cooler (Flownex model)

- PC_in.txt
- PC_ou.txt
- PC_oufl.txt
- IC_in.txt
- IC_ou.txt
- IC_oufl.txt
- Threshold_PCplant2.txt
- Threshold_ICplant2.txt
- hold_threshold.txt

7.4.2.3 Experimental data from LPC and HPC

- LPC_inj_in.txt
- LPC_inj_ou.txt
- HPC_inj_in.txt
- HPC_inj_ou.txt
- fault_inj.txt
- LPC_threshold.txt
- HPC_threshold.txt
- hold_2.txt

LIST OF REFERENCES

Clark, R., Patton, R. & Frank, P. 1989. Fault Diagnosis in Dynamic Systems, Theory and Application. UK: Prentice Hall Int.

Ferreira, T. 2003. South Africa's nuclear programme, Science in Africa. http://www.scienceinafrica.co.za/2003/june/pbmr.htm. Date of access: 15 Sep 2004

Frisk, E. 2001. Residual generation for Fault Diagnosis. Vehicular systems: Linköping Institute of Technology. (Thesis - PhD)

Gee, D. 2002. The Pebble Bed Modular Reactor, Course: EEE 460 http://www.eas.asu.edu/~holbert/eee460/dfg/. Date of access: 15 Sep 2004

Gee, D. 2002. Link to PBMR website, hosted by British Nuclear Fuel and Exelon http://www.pbmr.com/. Date of access 15 Sep 2004

Gertler, J. 1991. Analytical redundancy methods in fault detection and isolation-survey and synthesis, In IFAC Fault Detection, Supervision and Safety for Technical Processes. Germany: Baden-Baden.

Glad, T. & Ljung, L. 1991. Modellbygge och simulering, Studentlitteratur.

Greyvenstein, G. P. & Rousseau, P. G. 2003. Design of a physical model of the PBMR with the aid of Flownet. Nuclear Engineering and Design, 222(2-3): 203-213, Jun.

Hassoun, M.H. 1995. Fundamentals of Artificial Neural Networks. Cambridge: The MIT Press

Haykin, S. 1994. Neural Networks – A Comprehensive Foundation. New York: Macmillan College publishing Company

Howard, B. 1996. Neural network design. www.mathworks.com. Date of access: Jan 2005

Hunt, K.J., Sbarbaro, D. & Gawthrop, P.J. 1992. Neural Networks for Control Systems – A Survey. Automatica, 28(6): 1083-1112

Isermann, R. 2004. Model-based fault detection and diagnosis status and applications. Institute of Automatic Control: Darmstadt University of Technology

Isermann, R. & Ballé, P. 1997. Trends in the Application of Model-based Fault Detection and Diagnosis of Technical Processes. Control Engineering Practice, 5(5): 709-719.

Jeppesen, B.P. &. Cebon. D. 2001. Real-Time Fault Identification in an Active Roll Control System. Cambridge University Engineering Department, Trumpington Street, 20 Aug

Jia, Y. 2002. Model-based generic approaches for automated fault detection, diagnosis and evaluation. Drexel University. (Thesis - PhD)

Kaliberda, O. 1999. Modeling PID Regulator. www.geocities.com/oleg_kaliberda Date of acess: 03 Mar 2005

Karlsson, J. 2001. Diagnosis of the air distribution system of the JAS39 Gripen environmental control system. Vehicular systems: Linköping Institute of Technology. (Thesis- M.Sc). 28 Feb

Mann, H. Van Brussel, H. 1994. Metamodel and Design Methodologies for Mechatronics. Proceedings of the Tampere International Conference on Machine Automation, Mechatronics Spells Profitability, Finland: Tampere, Feb

McGhee, J. 1998. Sensor science – essentials for instrumentation and measurement technology. Industrial Control Centre: University of Strathclyde. 15 Oct

Nyberg, M. 1999. Model-based Fault Diagnosis Methods, Theory, and Automotive Engine Applications. Vehicular systems: Linköping Institute of Technology. (Thesis - PhD)

Olsson, R. 2002. Active model based diagnosis applied on the JAS39 Gripen fuel pressurization system. Vehicular systems: Linköping Institute of Technology. (Thesis- M.Sc), 14 Feb

Rakar, A. & Juricic, D. Matching the requirements in Model based diagnosis. Slovenia: Jozef Stefan institute

Sharkey, A.J.C. 1999. Diverse neural net solution to a fault diagnosis problem. University of Sheffield, (Thesis – M.Sc.), 30 Sep.

Simani, S., Fantuzzi, C. & Beghelli, S. 2000. Diagnosis Techniques for Sensor Faults of Industrial Processes. IEEE Transactions on control systems technology, 8(5), Sep.

Spaanenburg, L. & van Veelen, M. Model-based Containment of Process Fault Dissemination. Lund University: Dept. of Information Technology

Strydom, H.F. 2004. A comparison of different neural network topologies, Modelling of High and Low Pressure Compressors of the PBMR, Potchefstroom: North West University, (Thesis – M.Eng), Jun.

Surya N. & Kavuri, A. 2002. A review of process fault detection and diagnosis. West Lafayette: Purdue University, 22 Apr

Yoshiaki. I. 2001. A Perspective on Nuclear Power Generation in the Future Electric Power Industry - For Nonspecialists in the Electric Power Related Industries. Proceedings of the IEEE, 89(12), Dec.

Tako, L. 2001. Observed-based fault detection and diagnosis for non-linear system. Aalborg University, (Thesis – PhD)

Traichel, A., Schneider, V., Kästner, W. & Hampel, R. 2002. Advantages in Fault Diagnosis of Hydrostatic water level measurement in pressure vessels. Germany: University of Applied Sciences Zittau

Willis, M.J., Massimo, C.D., Montague, G.A., Tham, M.T. & A.J. Morris. A.J. 1991. Artificial Neural Networks in Process Engineering. IEE Proceedings-D, 138(3): 256-266.

Willsky, A.S. 1976. A survey of design methods for failure detection systems. Automatica, 12: 601-611.