Residual Generation and Fault Diagnosis of Rechargeable Lead-Acid Batteries

by

Sena Ergüllü

Submitted to the Graduate School of Sabancı University
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University

January, 2011

Residual Generation and Fault Diagnosis of Rechargeable Lead-Acid Batteries

APPROVED BY:

Assist. Prof. Dr. Ahmet Onat
(Thesis Advisor) .....

Assoc. Prof. Dr. Berrin Yanıkoğlu .....

Assoc. Prof. Dr. Serhat Yeşilyurt .....

Assist. Prof. Dr. Cemal Yılmaz .....

Assist. Prof. Dr. Hüsnü Yenigün .....

DATE OF APPROVAL: .....

To my most beloved family

This thesis was supported in part by The Ministry of Commerce of Turkey and Temsa Global, under the grand provided by SanTez project "'Ticari Araçlarda Uzaktan Arıza Tanıma ve Müdahale Sistemi"' ("'Remote Fault Diagnosis for Commercial Vehicles"'), code 00241.STZ.2008-1.

Residual Generation and Fault Diagnosis of Rechargeable Lead-Acid Batteries

Sena ERGÜLLÜ

ME, Master's Thesis, 2011

Thesis Supervisor: Assist. Prof. Dr. Ahmet Onat

Keywords: Fault Diagnosis, Residual Generation, Modeling of Nonlinear Dynamic Systems, Artificial Intelligence Methods

Abstract

In many process and manufacturing industries, early detection of faults has great practical importance. Since it saves time and cost involved in the repairing of the equipment.

Qualitative methods such as neural networks and fuzzy logic are popular tools in model based fault detection and classification of nonlinear dynamic systems. Since it is difficult to accurately model these kind of systems. In the first part of this work, neural network and adaptive neuro-fuzzy logic methods are used in the modeling of a water-tank system to produce residuals for fault classification. This study shows that neural networks have better performance but longer training time compared to the adaptive neuro-fuzzy logic. The second part of this research investigates the classification tree and Fisher Discriminant Analysis (FDA) approaches in fault classification of nonlinear dynamic systems. Comparing the performance of these approaches indicates that FDA method results in longer computational time but lower tree size for high dimensional training data. The contributions of this thesis are modeling and fault diagnosis of lead-acid battery system using qualitative techniques in combination with statistical methods such as classification tree.

Şarj Edilebilen Kurşun-Asit Bataryalarda Rezidu Oluşturma ve Hata Diyagnozu

Sena Ergüllü

ME, Master Tezi, 2011

Tez Danışmanı: Doç. Dr. Ahmet Onat

Anahtar Kelimeler: Hata Diyagnozu, Rezidu Üretme, Lineer Olmayan Dinamik Sistemin Modellenmesi, Yapay Zeka Metodları

Özet

Birçok üretim endüstrisinde hatanın erken tespiti önemli rol oynamaktadır. Bu zamandan ve maliyetten kazanç sağlayacaktır.

Yapay Sinir Ağları ve Bulanık Mantık gibi kalitatif metodlar, lineer olmayan dinamik sistemlerin modele dayalı hata tespit ve sınıflandırılmasında sıkça kullanılan yöntemlerdir. Bunun sebebi, bu sistemlerin doğru modellenmesi çok zordur. Bu çalışmanın ilk kısmında yapay sinir ağları ve adaptif sinir ağı-bulanık mantık metodları ile su tankı sistem modellemesi yapılmıştır. Böylece hata sınıflandırmada kullanılacak artıkların üretilmesi hedeflenmiştir. Bu çalışmadan görülmüştür ki yapay sinir ağları , adaptif sinir ağı-bulanık mantıktan daha iyi sonuç vermektedir, ama eğitim süresi uzamaktadır. Araştırmanın ikinci kısmında lineer olmayan dinamik sistemlerde hatanın sınıflandırması için sınıflandırma ağacı ve Fisher Diskriminant Analizi (FDA) yöntemleri kullanılmıştır. Bunların performansları karşılaştırıldığında FDA yöntemiyle büyük boyutlu eğitim verileri için daha uzun sürede ama daha az yapraklı ağaç oluşturulduğu görülmüştür. Bu tezin katkıları Şarj Edilebilir Kurşun-Asit Bataryaların modellenmesi ve hata diyagnozu alanında olmuştur. Bunun için gözlemsel metotlarla istatistik metotlar (sınıflandırma ağacı gibi) birleştirilmiştir.

Acknowledgements

It is a great pleasure to extend my gratitude to my thesis advisor Assist. Prof. Dr. Ahmet Onat for his precious guidance and support. I am greatly indebted to him for his supervision and excellent advises throughout my Masters study. I gratefully thank Assoc. Prof. Dr. Berrin Yanıkoğlu, Assoc. Prof. Dr. Serhat Yeşilyurt, Assist. Prof. Dr. Cemal Yılmaz and Assist. Prof. Dr. Hüsnü Yenigün for their feedback and their valuable time serving as my jury members.

I would like to acknowledge the financial support provided by The Ministry of Commerce of Turkey and TEMSA; through the project of SANTEZ "Remote Fault Diagnosis of Commercial Vehicles" under the grant 00241.STZ.2008-1.

I would sincerely like to thank to "Remote Fault Diagnosis of Commercial Vehicles" project member Yusuf Sipahi, for his pleasant team-work and providing me the necessary motivation during hard times.

Many thanks to Ahmetcan Erdoğan, Can Palaz, Kadir Haspalamutgil, Serhat Dikyar, Alper Mehmet Ergin, Ozan Tokatlı, Duruhan Özçelik, Aykut Cihan Satıcı and to all Mechatronics laboratory members, whom I wish I had the space to acknowledge in person for their great friendship throughout my Masters study.

Finally, I would like to thank my family for all their love and support throughout my life.

Contents

List of Figures

List of Tables

Chapter I

1 Introduction

Fault Diagnosis research deals with real world problems such as plant efficiency, maintainability and reliability. For safety critical systems, such as nuclear applications in plants and aircrafts, the detection of fault occurrence is highly important.

The consequences of the faults could be disastrous in these systems in terms of human mortality and environmental impact. Also in process and manufacturing industries, fault detection is crucial in order to improve the production efficiency, quality of the product and cost of the production.

There are two important approaches for fault diagnosis: *hardware redundancy* and *analytical redundancy*. Hardware redundancy uses multiplication of physical devices and a system to detect the occurrence of a fault and its location in the system. The main problem is the significant cost of the extra equipment. Analytical redundancy uses redundant functional relationships between the variables of the system. The main advantage of this approach compared to the hardware redundancy is that no extra equipment is necessary. However it requires more processing power.

1.1 Fault Detection and Diagnosis

In the early 1970s fault detection based on analytical methods has begun. Beard [1] designed an observer-based fault detection scheme and Johns [2]

continued his work. Their contribution is named as Beard-Johns Fault Detection Filter. Statistical approaches to fault diagnosis were first used in [3]. Luenberger observers were applied for the first time in [4]. Also Mironovsky [5] proposed a residual generation scheme based on consistency checking on the system input and output over a time window.

In 1980s and early 1990s major approaches on quantitative fault diagnosis were developed: observer-based approach, parity relation method, parameter estimation method etc [6]. It must be noticed that these methods are well-established theoretically. Therefore they are called classical or quantitative fault detection methods.

These methods have in common the use of a set of analytical redundancy relationships that represent the model of the system which follows the desired performance of the monitored system. The system is monitored for possible digressions that indicate the occurrence of the faults and may assist in isolating the faulty components.

In the last decade the research focused on fault diagnosis for nonlinear dynamic systems. Computational intelligence techniques such as neural networks, fuzzy systems and genetic algorithms have been successfully applied to the fault diagnosis.

1.1.1 Basic Terminology

These definitions are taken from International Federation of Automatic Control (IFAC) terminology.

Fault Diagnosis, Fault Tolerant Control

A *fault* represents an unexpected change of system function, although it may not represent a physical failure. *Failure* indicates a serious breakdown of a system component or function that leads to a significantly deviated

behavior of the whole system. The term fault rather indicates a malfunction that does not affect significantly normal behavior of the system.

An *incipient (soft)* fault represents a small and slowly increasing fault. At the beginning effects on the system are unnoticeable. A fault is called *hard (abrupt)* fault if its effects on the system are longer and bring the system very close to the limit of unacceptable behavior.

A fault is called intermittent if its effects on the system are hidden for discontinuous periods of time [6]. Although a fault is tolerable at the moment it occurs, it must be diagnosed as early as possible, otherwise it may lead to serious consequences in time.

A fault diagnosis system is a monitoring system that is used to detect faults and diagnose their location and significance. The system performs the following three functions:

Fault Detection: to indicate if a fault occurred or not in the system.

Fault Isolation: to determine the location of the fault

Fault Identification: to estimate the size and nature of the fault.

As another concept, a *fault tolerant control* system is a controlled system that continues to operate normally although there are faults in the system or in the controller. An important aspect of this system is automatic reconfiguration, once a malfunction is detected and isolated. Fault diagnosis decides how to perform the reconfiguration.

1.1.2 Statement of the Problem

Although technological developments have led to increasingly reliable mechanical, electrical and electronic vehicle systems, these systems still fail. The main goal of fault diagnosis system in a vehicle is to avoid damage to the vehicle and prevent dangerous situations for occupants.

In this research fault diagnosis of commercial vehicles, which requires diagnosis of all faults in the system, is performed. Fault diagnosis in vehicles is essential for low fuel consumption, high safety, efficient service and maintenance.

Objectives of This Research

- Investigate model based fault detection and diagnosis algorithms for nonlinear dynamic systems such as water-tank system, lead-acid battery system and mass-spring-damper system.
- Design a reasonable model of these systems and create fault scenarios.
- Validate the developed fault diagnosis algorithms on simulation and real time environment and compare their performances.

1.2 Fault Diagnosis Based on Analytical Models

Model based fault diagnosis is determination of the faults by comparing available system measurements with a priori information represented by the analytical model of the system through generation of residuals and their analysis. A *residual* is a fault indicator that reflects the faulty condition of the monitored system [7] similar to temperature or blood glucose level measurements of a patient which are used as symptoms to diagnose a disease.

Unfortunately an analytical model of the system is rarely accurate due to uncertainties, disturbances and noise. This results in differences between the analytical model output and the system output due to unmodelled dynamics and other uncertainties.

A fault diagnosis task contains two stages: *residual generation* and *residual evaluation* which are shown in Figure 1.1. Residual generation is a procedure to extract fault symptoms from the system using available input and

output information.

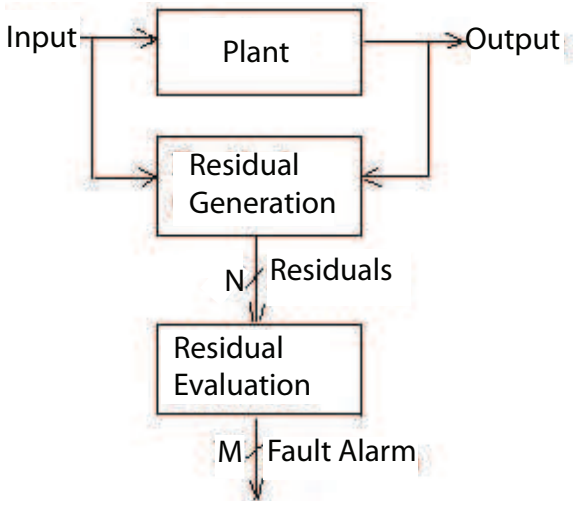


Figure 1.1: Two main stages in Fault Diagnosis

Residual generation represents an algorithm which is used to generate residuals. Residual evaluation represents examining residual signals in order to decide if a fault has occurred. It also isolates the fault. In most cases they must themselves be nonlinear dynamic systems. They may be implemented using statistical methods, e.g. likelihood ratio testing or classification tree [6].

Chapter II

2 System Modeling

In this section residual generation structure will be given and analytical conditions for fault detectability and isolability will be discussed. For simplicity it will be assumed that a linear model can reproduce system dynamics.

In the case of nonlinear dynamics, it is assumed that the model is linearized around a few operating points. The transition between different operating regions is performed using qualitative techniques such as fuzzy logic [8]. However nonlinear systems will be considered later in the thesis.

The information used for fault diagnosis and isolation is the measured input to the actuators and the output of the sensors. The measured output $y(t)$ is also used by feedback control and the controller generates the control signal $u(t)$ which is shown in Figure 2.1.

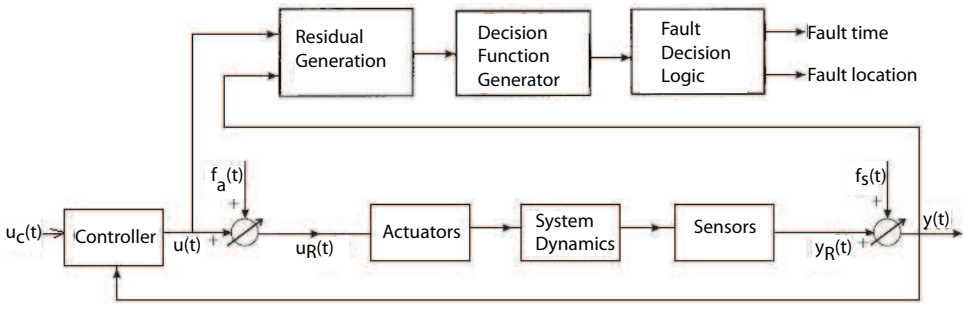


Figure 2.1: Fault Diagnosis System Scheme

If $u(t)$ is available Fault Diagnosis System (FDS) uses open loop model

of the system even if it is in a closed control loop. If the signal is not available, then FDS uses reference command $u_c(t)$ as an input. In this case the controller plays an important role because a robust controller can hide the effects of faults, therefore making fault diagnosis difficult [9].

2.1 General Structure of Faulty Systems

The state space model of the plant shown in Figure 2.1:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu_R(t) \\ y_R(t) &= Cx(t) \end{aligned} \tag{2.1}$$

where $x \in R^n$ is the state vector of the plant, $u_R \in R^r$ is the input vector to the actuator and $y_R \in R^m$ is the output vector of the plant. Under normal operating conditions,

$$\begin{aligned} u_R(t) &= u(t) \\ y(t) &= y_R(t) \end{aligned} \tag{2.2}$$

A, B, C are known matrices with known dimensions. Faults in the system could occur due to actuators, system components and sensors. The dynamics of the system can change as follows:

- actuator fault

$$u_R(t) = u(t) + f_a(t) \tag{2.3}$$

  where $f_a \in R^r$ is the actuator vector fault.

- system dynamics (component) fault

$$\dot{x}(t) = Ax(t) + Bu_R(t) + f_c(t) \tag{2.4}$$

  where $f_c \in R^n$ is the component vector fault.

- sensor fault

$$y(t) = y_R(t) + f_s(t) \tag{2.5}$$

where $f_s \in R^m$ is the sensor vector fault.

If the previous three fault categories are considered simultaneously, the time domain representation of the system model changes to:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + Bf_a(t) + f_c(t) \\ y(t) &= Cx(t) + f_s(t) \end{aligned} \tag{2.6}$$

In more general case with all possible faults in the state space model:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + R_1 f(t) \\ y(t) &= Cx(t) + R_2 f(t) \end{aligned} \tag{2.7}$$

where $f(t) \in R^g$ is a fault vector, $f_i(t)$(i= 1...g) are specific faults and R_1 and R_2 are fault entry matrices which represents the effect of faults on the system.

Input-output transfer matrix in frequency domain for the faulty system model is:

$$y(s) = G_u(s)u(s) + G_f(s)f(s) \tag{2.8}$$

where

$$\begin{aligned} G_u(s) &= C(sI - A)^{-1}B \\ G_f(s) &= C(sI - A)^{-1}R_1 + R_2 \end{aligned} \tag{2.9}$$

2.2 General Structure of Residual Generation

Input values of a residual generator are inputs and outputs of the monitored system as expressed by:

$$r(s) = H_u(s)u(s) + H_y(s)y(s) \tag{2.10}$$

where $H_u(s)$ and $H_y(s)$ are transfer matrices realizable using stable linear systems.

The residual ($r(t)$) must be designed (in ideal case) to be zero for fault free case and nonzero when a fault occurs which is shown in (2.11).

$$r(t) = 0 \text{ if and only if } f(t) = 0 \tag{2.11}$$

Therefore the matrices $H_u(s)$, $H_y(s)$ and $G_u(s)$ (defined in (2.9)) must satisfy the following constraint condition:

$$H_u(s) + H_y(s)G_u(s) = 0 \tag{2.12}$$

Equation (2.12) is a generalized representation of all residual generators [7]. For the aim of residual generation design, one must choose two matrices which satisfy (2.12). Based on the parametrization chosen for $H_u(s)$ and $H_y(s)$, a different way to generate residuals is obtained.

Assume that J is a function of residual signal $r(t)$. Fault detection is done by comparing the residual evaluation function $J(r(t))$ with a threshold function $T(t)$ using condition in (2.13). If the residual exceeds the threshold, a fault may be occurred.

$$\begin{array}{lll} J(r(t)) \leq \mathrm{T(t)} & \text{for} & f(t) = 0 \\ J(r(t)) > T(t) & \text{for} & f(t) \neq 0 \end{array} \tag{2.13}$$

2.3 Fault Detectability and Fault Isolability

For a faulty system the residual vector is defined as:

$$\begin{array}{rcl} r(s) & = & H_y(s)G_f(s)f(s) = G_{rf}(s)f(s) \\ & = & [G_{rf}(s)]_1 f_1(s) + [G_{rf}(s)]_2 f_2(s) ... + [G_{rf}(s)]_i f_i(s) \end{array} \tag{2.14}$$

where $H_y(s)$, $G_f(s)$ and $f(s)$ are given in (2.10), (2.9) and (2.8), respectively.

The residual-fault relationship is represented by $G_{rf}(s) = H_y(s)G_f(s)$, where $[G_{rf}]_i$ is the i^{th} column matrix of G_{rf} and f_i is the i^{th} component of $f(s)$.

Fault Detectability Condition:

If the i^{th} column of $G_{rf}(s)$ is nonzero, $G_{rf}(s)_i \neq 0$, the fault f_i is detectable in the residual $r(s)$. This is called the *fault detectability condition* of the residual $r(s)$ to the fault f_i [7].

Fault Isolability Condition:

A fault is isolable using a residual vector set, if it is distinguishable from other faults using this set. Usually each residual from the considered set is sensitive to a subset of faults and insensitive to the others [10].

2.4 Quantitative Diagnosis Methods

The main point in model based fault diagnosis is residual generation method each of which has its specific technique of computing the residual vector.

Three important methods will be represented in this section. These methods focus on discrete-time dynamic linear models.

2.4.1 Residual Generation via Parameter Estimation

When the process parameters are not known exactly, they can be determined with parameter estimation methods by measuring the input and output signals, if the basic structure of the model is known [6] .

It is assumed that the faults are reflected in the physical system parameters and these parameters are estimated online using well-known parameter estimation methods. The results are then compared with the parameters of the reference model obtained under fault-free assumptions. Any discrepancy

would indicate that a fault may have occurred.

For the n^{th} order discrete-time estimated model:

$$\Theta = [a_1...a_n, b_1...b_n]^T \tag{2.15}$$

is the parameter vector where a_i and b_i (i=1,..,n) represent the coefficients in $A(z)$ and $B(z)$ transfer matrices.

Output error of the system (or the loss function) is calculated as:

$$e(t) = y(t) - \hat{y}(\Theta, t) \tag{2.16}$$

where

$$\hat{y}(\Theta, z) = \frac{\hat{B}(z)}{\hat{A}(z)} u(z) \tag{2.17}$$

is the model output in which $\hat{A}(z)$ and $\hat{B}(z)$ correspond to the estimates of $A(z)$ and $B(z)$ as depicted in Figure 2.2.

Since $e(t)$ is a nonlinear parameter, direct calculation of Θ is generally not possible. Numerical optimization methods can be used to minimize the loss function (2.16) as (2.17).

If a fault in the process changes one or several parameters by $\Delta\Theta$ the output changes for small deviations according to

$$\Delta y(t) = \psi(t)^T \Delta\Theta(t) + \Delta\psi(t)^T \Theta(t) + \Delta\psi(t)^T \Delta\Theta(t) \tag{2.18}$$

and the parameter estimator indicates a change $\Delta\Theta$.

Generally the process parameters Θ depend on physical process coefficients p(like stiffness,damping factor, resistance...). If L is a function depending on p,

$$\Theta = L(p) \tag{2.19}$$

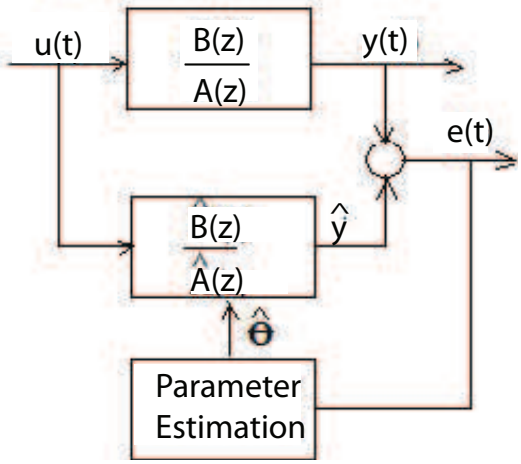


Figure 2.2: Parameter Estimation Method Output

via nonlinear algebraic equations. If the inversion of the relationship exists [11]:

$$p = L^{-1}(\Theta) \tag{2.20}$$

where changes (Δp) of the process coefficients, from which the fault alarm is obtained, can be calculated.

2.4.2 Observer Based Approaches

The main idea of the observer based technique is to estimate the outputs of the system from the measurements by using either Luenberger observers in a deterministic setting or Kalman filters in a noisy environment. The output estimation error is used as residual. The advantage of using observer is the flexibility in the selection of its gains which leads to a rich variety of FDS schemes [12].

In order to obtain the structure of an observer the discrete-time, time-invariant and linear dynamic model for the plant in a state space form is considered.

$$\begin{aligned} x(t+h) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{2.21}$$

where h is the sampling time interval, $u(t) \in R^r$, $x(t) \in R^n$ and $y(t) \in R^m$.

Assuming that all matrices A,B and C are perfectly known, an observer is used to reconstruct the system variables based on the measured inputs and outputs $u(t)$ and $y(t)$.

$$\begin{aligned} \hat{x}(t+h) &= A\hat{x}(t) + Bu(t) + He(t) \\ e(t) &= y(t) - C\hat{x}(t) \end{aligned} \tag{2.22}$$

The observer scheme described by (2.22) is drawn in Figure 2.3. For the state estimation error e_x(t), it follows from the equations (2.22) as:

$$\begin{aligned} e_x(t) &= x(t) - \hat{x}(t) \\ e_x(t+h) &= (A - HC)e_x(t) \end{aligned} \tag{2.23}$$

If the observer is stable, the state error e_x(t) vanishes asymptotically.

$$\lim_{t \to \infty} e_x(t) = 0 \tag{2.24}$$

This can be achieved by the proper design of the observer feedback matrix H [7]. Thus, the design of the observer feedback matrix H is important in residual generation. However if the signals are affected by noise, Kalman filter must be used instead of classical observers [13].

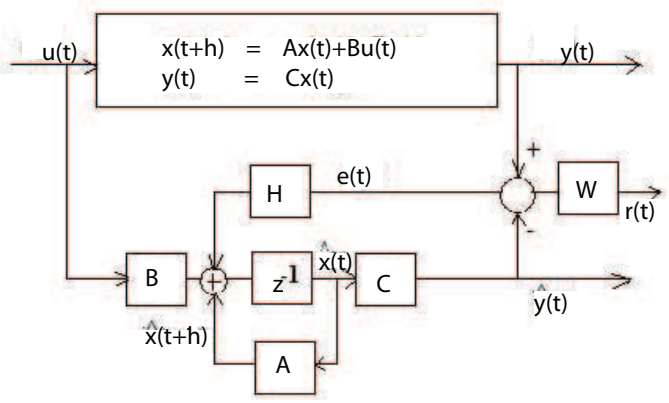


Figure 2.3: Process and State Observer

2.4.3 Parity Vector (relation) Methods

Parity relations approach provides a proper check of the parity (consistency) of measurements acquired from the monitored system. In the early development of fault diagnosis, the parity vector (relation) approach was applied to static or parallel redundancy schemes [14] which may be obtained directly from measurements (hardware redundancy) or from analytical relations (analytical redundancy).

In the first case, two methods can be used to obtain redundant relations which requires several sensors with similar functions to measure the same variable. The second approach consists of dissimilar sensors to measure different variables but their outputs being relative to each other.

In case of analytical model based fault detection, the model can be written in the form of $G_m(z) = \hat{A}(z)/\hat{B}(z)$ and to run it in a parallel to the process described by the transfer function $G_p(z)$:

$$G_p(z) = \frac{A(z)}{B(z)} \tag{2.25}$$

Thereby forming an error vector $e_P(z)$:

$$e_P(z) = (\frac{A(z)}{B(z)} - \frac{\hat{A}(z)}{\hat{B}(z)})u(z) \tag{2.26}$$

The methodology described here is shown in Figure 2.4.

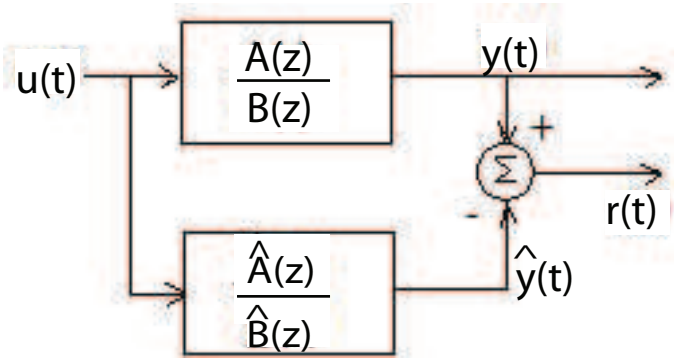


Figure 2.4: Output Error Method

Assume that

$$\begin{array}{rcl} G_m(z) & = & G_p(z) \text{ i. e.} \\ \frac{\hat{A}(z)}{\hat{B}(z)} & = & \frac{A(z)}{B(z)} \end{array} \tag{2.27}$$

then the residual becomes

$$e_P(z) = \frac{A(z)}{B(z)} f_u(z) + f_y(z) \tag{2.28}$$

where $f_u(z)$ and $f_y(z)$ are additive input and output faults which are shown in Figure 2.5. Moreover the error vector r(z) computed by (2.28) corresponds to the output error of the parameter estimation method which is computed by (2.16).

The residuals generated are called parity equations [10] under the assumptions of fault occurrence and of exact agreement between the process and the model.

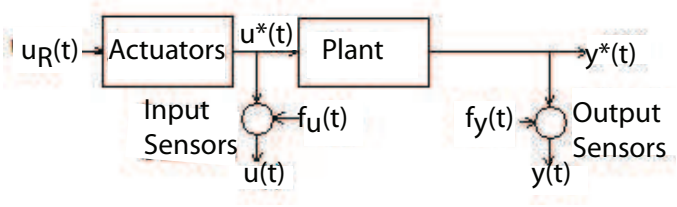


Figure 2.5: Fault Topology of the Monitored System

Therefore (2.28) can be used to implement and design the residual generation system in order to meet fault detection and isolation specifications as well [15] .

2.5 Qualitative Diagnosis Methods

2.5.1 Fuzzy Model Based Residual Generation

During the last forty years, modeling and control of dynamic systems with fuzzy set techniques have received considerable attention. Many systems are not suitable to conventional modeling techniques due to lack of precise, formal knowledge about the system and due to time varying characteristics [16] .

Fuzzy modeling along with neural networks are powerful tools to facilitate effective development models. One of the reasons for this case is that fuzzy systems are capable of integrating information from different sources such as physical laws, measurements and heuristics.

Fuzzy models can be seen as logical models which use "IF-THEN" rules to establish qualitative relationships among variables in the model. Fuzzy sets serve as smooth interfaces between the qualitative variables involved in the numerical data at the inputs and outputs of the model. The rule-based nature of fuzzy models uses the information expressed in the form of natural language statements.

More specifically, Mamdani [17] and Takagi Sugeno (TS) [8] are two successful fuzzy system types, since they are able to approximate any continuous function with a desired level of accuracy [18] .

Mamdani Fuzzy System

Fuzzy information is expressed as fuzzy sets and linguistic variables which are called membership functions. Also fuzzy rule base is required for the representation of fuzzy information generally in the form :

$$Rule^l : \text{IF } (x_1 \text{ is } A_1^l) \ldots (x_n \text{ is } A_n^l) \text{ THEN } (y \text{ is } B^l) \tag{2.29}$$

The membership functions of fuzzy sets A_i^l and B^l are denoted as $\mu_{A_i^l}$ and μ_{B^l}, respectively, where

$$\begin{aligned} \mu_{A_i^l} &: X \rightarrow [0,1], (i = 1, 2, .., n) \\ \mu_{B^l} &: Y \rightarrow [0,1], (l = 1, 2, ..., M) \end{aligned} \tag{2.30}$$

In (2.30); n is the number of inputs of the fuzzy system and M is the number of IF-THEN rules. In the Mamdani fuzzy model minimum fuzzy inference system is used. For a given input $x^* = (x_1^*, x_2^*, .., x_n^*) \in X$, the output of the fuzzy inference system $\mu_{B^l}(y)$ is defined as:

$$\mu_{B^l}(y) = max_{(1 \leq l \leq M, y \in Y)}[min_{(y \in Y)} \mu_{A_1^l}(x_1^*), \mu_{A_2^l}(x_2^*), .., \mu_{A_n^l}(x_n^*), \mu_{B^l}(y)] \tag{2.31}$$

where the min operator selects the minimum value among the values of membership functions in the IF proposition of a given input x^* and the membership function of the THEN proposition of the output universe Y. For the final output of the fuzzy system, a defuzzifier is needed which represents the fuzzy set at the output of the system.

TS Fuzzy System

Generally in nonlinear dynamic processes TS fuzzy models are preferred.

Unlike Mamdani fuzzy rules, TS rules use piecewise linear functions of the input variables.

Each rule comprises IF-THEN condition and has the following form:

$$Rule^l : \text{IF } (x_1 \text{is } A_1^l) \ldots (x_n \text{is } A_n^l) \text{ THEN } y_l = \sum_{i=1}^{n} k_i^l x_i + c_i^l) \tag{2.32}$$

where l refers to the l^{th} rule, n is the number of inputs, A_i^l is the fuzzy set in the input (antecedent) and y_l is a crisp first-order polynomial function in the output (consequent) [19]. Finally k_i and c_i represent a factor and a constant of the polynomial defined in the first order TS model, respectively.

Output of the fuzzy system with M rules is aggregated as:

$$y = \frac{\sum_{l=1}^{M} \mu_l y_l}{\sum_{l}^{M} \mu_l} \tag{2.33}$$

where μ_l is the degree of activation of the rule l:

$$\mu_l = \prod_{i=1}^{l} \mu_{A_i^l} \tag{2.34}$$

where $\mu_{A_i^l}$ is defined in (2.30).

Generally with the similar system requirements such as number of rules and membership function TS is more accurate than Mamdani model. Due to the incompleteness of knowledge, the rules and its predicates need to be updated to optimize the system. Also the main relations representing the Mamdani model is not continuous due to the presence of MAX or MIN operator. Therefore the optimization techniques that use derivatives, e.g. gradient descent method can not be applied. This makes the Mamdani model less adaptable to fault diagnosis application [20].

2.5.2 Neural Network Model Based Residual Generation

The potential of neural networks for fault detection and isolation has been better understood recently. Artificial neural network based approach is

especially suitable for processes for which accurate mathematical models are too difficult or too expensive to obtain.

Neural networks try to mimic the computational structures of the mammal brains by nonlinear mapping between the input and output that consists of interconnected nodes arranged in layers. The layers are connected such that the signal on the input side can propagate through the network and reach output side. Neural network behaviors are determined by the transfer functions of the units, network topology and connection pattern of layers.

Among all the forms of Neural Networks, the two layer feed-forward neural network has been the most popular. This class of networks consists of two layers of nodes, namely the hidden layer and the output layer. Also there exist two layers of weights serving as connection between the input and the hidden layer, as well as between the hidden layer and the output layer. No connection is allowed with its own layer and the information flows in one direction. Sigmoid functions are usually selected as the transfer function for hidden layer nodes and linear functions for the nodes of the output layer. Equations of the transfer functions are:

$$f_1(x) = x \tag{2.35}$$

$$f_2(x) = \frac{1}{1+e^{-x}} \tag{2.36}$$

$$f_3(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \tag{2.37}$$

where the functions are pure linear, log-sigmoid and tan-sigmoid functions respectively.

This class of neural networks can approximate any functional continuous mapping from one finite dimensional space to the other arbitrarily well, provided that the number of hidden neurons is sufficiently large. Therefore this class of neural networks and two layers of weights can approximate any

decision boundary to within arbitrary accuracy. This is the reason why two layer functions are applied to process modeling and fault diagnosis by pattern recognition.

During the training time, neural network uses the error in the output values to update the weights connecting layers, until the accuracy is within the tolerance level. The training time for the feed forward neural network using one of the variations of backpropagation is important. For large scale applications, memory and computation time required for training a neural network can exceed the hardware limits. Therefore the performance of neural networks is determined by the available data. It is possible that neural networks will generate unpredictable outputs when presented with an input out of the range of training data. So retraining of neural network may be required.

Neural networks can be applied to fault detection and diagnosis as a process model or a pattern classifier. For this purpose neural networks can be summarized in three categories which is shown in Figure 2.6.

In the first one, neural network is used to differentiate various faulty output patterns from normal operating conditions. According to the different measured process output data. Training of the neural network can be performed offline or online. In the second figure, neural networks are used as classifiers to isolate faults represented by process model-generated residuals. The process model is the mathematical model of the process based on fault diagnosis structure which uses the mechanism provided by the model. When the mathematical models are not available, a neural network process model can be employed to generate residuals; another network is then used to isolate faults.

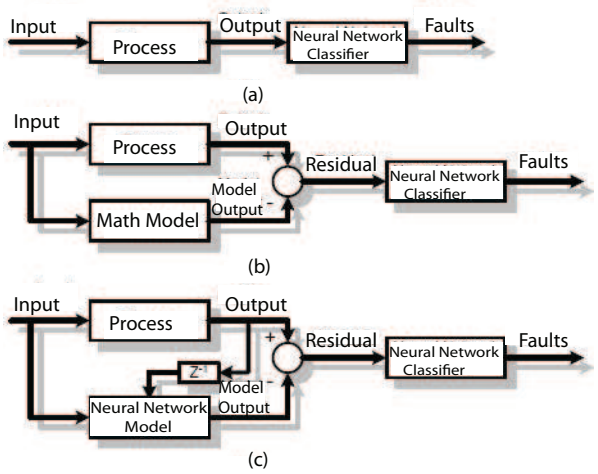


Figure 2.6: Neural Network Applications in Fault Diagnosis

2.5.3 Neuro-Fuzzy Model Based Residual Generation

The main drawback of neural networks as stated in the previous section is their "black box" nature, while the disadvantage of fuzzy systems is represented by difficult and time consuming process of knowledge acquisition. The advantage of neural network over fuzzy systems is learning and adaptation capabilities, while the advantage of fuzzy system is the human understandable form of knowledge representation. Neural networks use an implicit way of knowledge representation while neuro-fuzzy systems represent knowledge in an explicit form, such as rules.

The combination of neural networks and fuzzy systems can be done in two ways:

Neural networks implemented using fuzzy logic

These hybrid systems are mainly neural networks equipped with abilities of processing fuzzy information. These systems are usually termed as Fuzzy Neural Networks and they are networks where the inputs, outputs and weights are fuzzy sets, and they consist of a special type of neurons called fuzzy neurons.

Fuzzy logic implemented using neural networks

These systems can be viewed as fuzzy systems augmented with neural network facilities, such as learning, adaptation and parallelism. These systems are usually called Neuro-Fuzzy Systems. Neuro-Fuzzy Systems can be always interpreted as a set of fuzzy rules and can be represented as a feed-forward network architecture [21].

In addition to these two approaches, there is another way of hybridization of neural networks and fuzzy systems, where each method maintains its own identity and the hybrid neuro-fuzzy system consists of modules cooperating in solving the problem. These kind of neuro-fuzzy systems are combinations of hybrid systems. Detailed explanations and related equations are given in section 3.2.2.

In some approaches a neural network (such as self organizing map) can preprocess the input data for fuzzy system. However in fault diagnosis applications, fuzzy system is used as a pre-processor for a neural network.

A Neuro-fuzzy (NF) system is a neural network which is topologically equivalent to the structure of a fuzzy system. Network inputs/outputs and weights are real numbers but the network operations are specific to fuzzy systems: fuzzification, fuzzy operations (conjunction , disjunction), defuzzification. Therefore NF systems can be used to identify fuzzy models directly from input-output relationships, but they can be used to optimize an initial

fuzzy model acquired from human expert, using additional data.

2.6 Conclusion

In this section the importance of fault diagnosis for industrial systems is explained, a literature overview is done and basic terminology about fault detection is given.

The ways of designing residuals are discussed. The most commonly used residual generation techniques are introduced and applicability of analytical model based fault diagnosis are discussed.

Other Fault Diagnosis methods such as fuzzy logic, neural networks and qualitative modeling have been discussed. In the next chapter this method will be implemented on a water-tank system and the residual will be obtained using the analytical model. Then the residual will be classified using the same methodology and faults will be identified clearly.

Chapter III

3 Fault Diagnosis Based on Qualitative Methods

3.1 Introduction

The main objective of fault diagnosis is early warning for the operators to take appropriate measures and prevent the system from breaking down after the occurrence of faults. This will improve the reliability and safety of the system [22]. Since the systems are becoming more complex, automated fault monitoring schemes are developed in case of human operators.

For fault diagnosis of nonlinear plants, computer intelligence based methods such as neural networks, fuzzy logic and genetic algorithms are often used [23] [24]. Among these techniques, neural networks are important for their ability to approximate nonlinear functions and their online learning ability. Also they can be used as a model to generate residuals for classifying and isolating the faults [25]. However their disadvantage is the difficulty in isolating the faults due to their black box nature. Another approach is fuzzy reasoning which allows symbolic generalization of numerical data by fuzzy rules and expert knowledge integrated into the fault diagnosis procedures to achieve better diagnosis [26]. On the other hand adaptive neuro-fuzzy systems have the ability of neural networks which can approximate nonlinear functions with arbitrary accuracy and they are able to incorporate fuzzy

rules which allows expert knowledge in linguistic form to be included.

In this section, a brief overview is given about Adaptive Neuro Fuzzy System (ANFIS) structure and neural network structure. The proposed scheme is to be illustrated by a simulation example of a water-tank system. Performances of these two approaches are compared. Finally a conclusion is drawn.

3.2 ANFIS Structure

A Fuzzy Logic System (FLS) is a nonlinear mapping from an input space to an output space. The mapping is based on the conversion of the inputs from crisp numerical domain to fuzzy domain using fuzzy sets and fuzzifiers, and then applying fuzzy rules and fuzzy inference engine to perform the necessary operations in the fuzzy domain. In the end, the result is converted back to the crisp numerical domain using defuzzifiers. Hence, a FLS contains five main components: fuzzy sets, fuzzifiers, fuzzy rules, an inference engine and defuzzifiers [27]. Adaptive neuro-fuzzy networks are enhanced FLSs with learning, generalization and adaptation capabilities. These networks encode the fuzzy if-then rules into a neural network-like structure and then use appropriate learning algorithms to minimize the output error based on training/validation datasets.

ANFIS is a Fuzzy-Sugeno model of integration where the final fuzzy inference system is optimized via neural network training [8]. It maps the inputs through the input membership functions and parameters, then through the output membership to the outputs. It will be explained next.

For simplicity a first order Sugeno model is considered to represent the

fuzzy inference system which is expressed with four rules as follows:

$$\begin{aligned}
&\text{Rule 1 : If } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } f_1 = p_1x + q_1y + r_1 \\
&\text{Rule 2 : If } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } f_2 = p_2x + q_2y + r_2 \\
&\text{Rule 3 : If } x \text{ is } A_1 \text{ and } y \text{ is } B_2 \text{ then } f_3 = p_3x + q_3y + r_3 \\
&\text{Rule 4 : If } x \text{ is } A_2 \text{ and } y \text{ is } B_1 \text{ then } f_4 = p_4x + q_4y + r_4
\end{aligned} \tag{3.1}$$

where x and y are the inputs, A_i and B_i are the fuzzy sets and f_i (i=1,2,3,4) are the membership functions (fuzzy region specified by the fuzzy rules) and p_i,q_i and r_i are the design parameters. "If x is A_1 and y is B_1" part is called the *premise part of a rule*, and "then $f_1 = p_1x + q_1y + r_1$" is called the *consequent part of a rule*. Using f_i (i=1,2,3,4) the output function for this model is expressed as,

$$\begin{aligned}
f &= \frac{w_1f_1+w_2f_2+w_3f_3+w_4f_4}{w_1+w_2+w_3+w_4} \\
&= \bar{w}_1f_1 + \bar{w}_2f_2 + \bar{w}_3f_3 + \bar{w}_4f_4
\end{aligned} \tag{3.2}$$

where w_i (i=1,2,3,4) is explained in (3.6). ANFIS architecture for these four rules is illustrated in Figure 3.1.

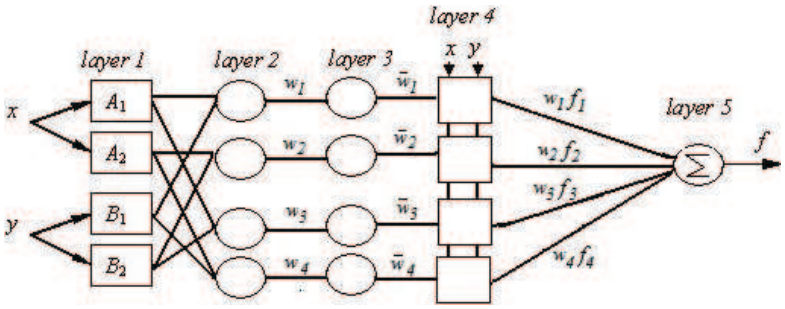


Figure 3.1: ANFIS Model of Sugeno's fuzzy inference method

This structure has five layers and the node functions in each of these layers are explained below:

Layer 1: Membership Value of Input

The fuzzification process is taken here where the membership functions transform the input x to the output O_i^1. The output of every node i in the first layer is defined with a node function O_i^1 where the superscript denotes the layer number:

$$O_i^1 = \mu_{A_i}(x), (\text{for } i\text{=}1,2) \tag{3.3}$$

$$O_i^1 = \mu_{B_{i-2}}(x), (\text{for } i\text{=}3,4) \tag{3.4}$$

where x is the input to node i, $\mu_{A_i}(x)$ is the membership function defining linguistic label (small, large, etc.) A_i (or B_i). Generally the $\mu_{A_i}(x)$ is chosen as bell-shaped, which is between 0 and 1, with the following formula:

$$\mu_{A_i}(x) = \frac{1}{1 + \left|\frac{x-c_i}{a_i}\right|^{2b_i}} \tag{3.5}$$

where a_i, b_i, c_i are referred to the *premise* parameter set of this layer. Other continuous and piecewise differentiable functions such as trapezoidal or triangular-shaped membership functions can also represent the node functions in this layer.

Layer 2: Firing Strength of Rule

Each node output represents a firing strength of a rule. The T-norm (product, fuzzy-AND..) operators perform the node function in this layer. These nodes multiply the incoming signals and send the product out. For instance,

$$O_i^2 = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), (i = 1,2) \tag{3.6}$$

where x is the T-norm implemented as product.

Layer 3: Normalized Firing Strengths

The normalization process is performed in this layer. The i^{th} node calculates the ratio of the i^{th} rule's firing strength to the sum of all rules' firing strengths:

$$O_i^3 = \bar{w_i} = \frac{w_i}{w_1 + w_2 + w_3 + w_4}, (i = 1,2,3,4) \tag{3.7}$$

Layer 4: Consequent Parameters

Every node in this layer is shown with a node function:

$$O_i^4 = \bar{w_i} f_i = \bar{w_i}(p_i x + q_i y + r_i) \tag{3.8}$$

where $\bar{w_i}$ is the node output of layer 3 and p_i, q_i, r_i is the parameter set which is called as *consequent* parameters. The output of this layer forms Takagi-Sugeno outputs.

Layer 5: Overall Output

The single node in this layer sums all the incoming signals and computes the overall output. Output is linear in terms of the consequent parameters as seen in the last equation of (3.9):

$$\begin{aligned} O_1^5 &= \sum_i \bar{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \\ &= \frac{w_1}{w_1 + w_2 + w_3 + w_4} f_1 + \frac{w_2}{w_1 + w_2 + w_3 + w_4} f_2 + ... \\ &= \bar{w_1}(p_1 x + q_1 y + r_1) + \bar{w_2}(p_2 x + q_2 y + r_2) + ... \\ &= (\bar{w_1} x)p_1 + (\bar{w_1} y)q_1 + (\bar{w_1})r_1 + (\bar{w_2} x)p_2 + (\bar{w_2} y)q_2 + (\bar{w_2})r_2 + ... \end{aligned} \tag{3.9}$$

Input space partitioning of the two-input ANFIS structure is shown in Figure 3.2. Grey area shows the undetermined region whereas dark area is the membership function combinations for these inputs. Two membership functions are associated with each input. The premise part of a rule (defined in 3.1) linearizes the fuzzy subspace and the consequent part (defined in 3.1) specifies the output within this fuzzy subspace. Therefore ANFIS uses two set of parameters which are called as $S1$ and $S2$ where $S1$ is the set of premise parameters (a_i,b_i,c_i for i=1,2,3,4)and $S2$ is the set of consequent parameters (p_i,q_i,r_i for i=1,2,3,4).

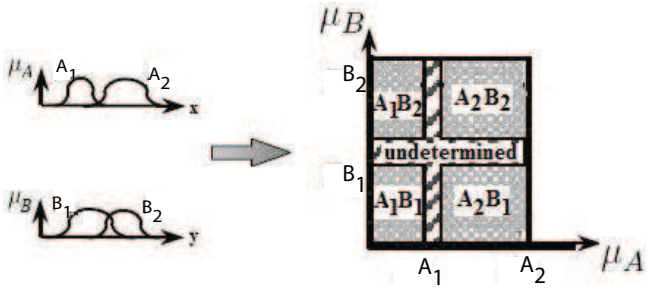


Figure 3.2: Input space partitioning of ANFIS structure

ANFIS learning algorithm is a two-pass hybrid learning algorithm consisting of forward pass and backward pass. In the forward pass, functional signals go forward up to the layer 4 and $S2$ parameters are computed using least squared error (LSE) algorithm on layer 4. In the backward pass, the error rates are propagated backward and $S1$ parameters are computed using a gradient descent algorithm (usually backpropagation) to be explained next. In Table 1 the signals and parameters for each pass is represented.

Table 1: Two passes in hybrid learning algorithm for ANFIS

	Forward Pass	Backward Pass
Premise Parameters	fixed	gradient descent
Consequent Parameters	least-squares	fixed
Signals	Node Outputs	Error Signals

3.2.1 ANFIS LSE Algorithm

In the forward pass of hybrid learning algorithm, consequent parameters are identified by least squares estimate. Assume that S is the total parameter set which is the combination of $S1$ and $S2$ sets.

$$S = S1 \cup S2 \text{ and } S1 \cap S2 = \phi \tag{3.10}$$

Then the output becomes:

$$Output = F(\bar{I}, S) \tag{3.11}$$

where $\bar{I}$ is the input vector and

$$H(Output) = HoF(\bar{I}, S) \tag{3.12}$$

where H is a function of output and H o F is linear in terms of S_2.

For the given values of S_1, using P training data 3.12 can be transformed into the equation:

$$B = AX \tag{3.13}$$

where X is the unknown vector containing the elements of S_2.

Generally no exact solution is found for this equation. Therefore LSE minimizes the error $\|AX - B\|^2$ by approximating X with X^* (least squares estimate of X). The estimate of X, X^* can be defined as:

$$X^* = (A^T A)^{-1} A^T B \tag{3.14}$$

where A^T is the transpose of A, and $(A^T A)^{-1} A^T$ is the pseudo-inverse of A if $A^T A$ is nonsingular.

It is difficult to compute the LSE of X^* because P is large. Therefore X is often solved iteratively using the formulas [28]:

$$S_{i+1} = S_i - \frac{S_i a(i+1) a(i+1)^T S_i}{1 + a(i+1)^T S_i a(i+1)}, \tag{3.15}$$

$$X_{i+1} = X_i + S(i+1)a(i+1)(b(i+1)^T - a(i+1)^T X_i) \tag{3.16}$$

for $i = 0, 1, ..., P-1$ where $X_0 = 0, S_0 = \gamma I$ (γ is a large number and I is identity matrix), a_i^T is the i^{th} row of matrix A, b_i^T is i^{th} element of vector B, X^* is X_P (X_{i+1} value for $i = P-1$).

3.2.2 ANFIS Backpropagation Algorithm

For a training data set with P entries, the error measure or energy function can be defined as:

$$E_p = \sum_{m=1}^{N(L)} (T_{m,p} - O_{m,p}^L)^2 \tag{3.17}$$

where $(1 \leq p \leq P)$; $N(L)$ is the number of nodes in layer L; $T_{m,p}$ is the m^{th} component of p^{th} target output vector and $O_{m,p}^L$ is the m^{th} component of actual output vector. The overall error measure is:

$$E = \sum_{p=1}^{P} E_p \tag{3.18}$$

Next, the error rate is calculated for the gradient descent in E over the parameter space. The error rate for the output node at (L, i) can be calculated as:

$$\frac{\partial E_p}{\partial O_{i,p}^L} = -2(T_{i,p} - O_{i,p}^L) \tag{3.19}$$

For the internal node at (k, i) the error rate is defined by the chain rule:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{N(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^k} \tag{3.20}$$

where $(1 \leq k \leq L-1)$.

Generalization of this equation for the α parameter is:

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}, \tag{3.21}$$

where S is the set of nodes whose outputs depend on α. The derivative of the overall measure E with respect to α is:

$$\frac{\partial E_p}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \tag{3.22}$$

For the generic parameter α the updated formula is:

$$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha} \tag{3.23}$$

where $\eta = \frac{k}{\sqrt{\sum_\alpha (\frac{\partial E}{\partial \alpha})^2}}$ is the learning rate, k is the step size and $\frac{\partial E}{\partial \alpha}$ is the ordered derivative.

Two-pass training is much faster than the gradient descent algorithm since it decomposes the parameter set as $S1$ and $S2$. It is possible if the membership function of each rule is replaced by a piecewise linear approximation with two consequent parameters. As seen in Figure 3.3, the consequent parameters constitute set $S2$ and the hybrid learning rule can be applied directly.

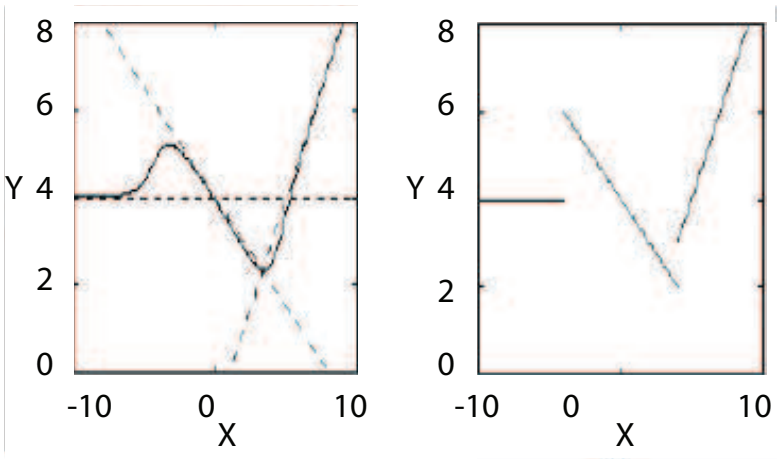


Figure 3.3: Piecewise Linear Approximation of ANFIS Output

3.3 Neural Network Structure

Feed-forward neural network is a nonlinear mapping between input and output that consists of interconnected nodes arranged in layers. The layers are connected such that signal on the input side propagates through the network and reaches the output side without feedback loops.

Consider a multilayer feed-forward neural network with one hidden layer shown in Figure 3.4. The input signals to the n_i input layer nodes are denoted by x_1,x_2...,x_{n_i}; the output signals of the n_O output layer nodes are denoted by y_1,y_2...,y_{n_O}; and the output signals of the n_h hidden layer nodes are denoted by h_1,h_2...,h_{n_h}. The nonshaded nodes are bias nodes with inputs set equal to unity. Connection between nodes of different layers of network are weights and biases which correspond to the dotted line connections in Figure 3.4. No connection is allowed back to its own layer and the information flow is only one directional.

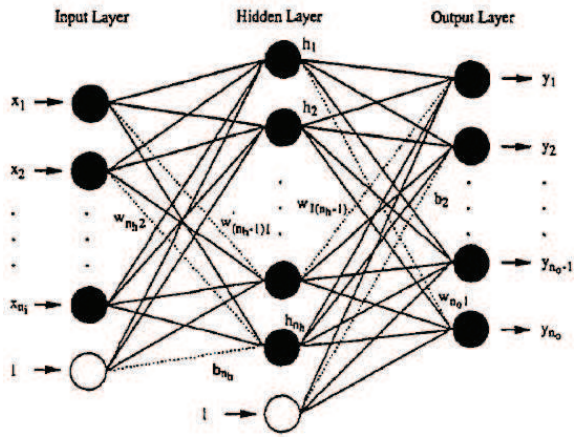


Figure 3.4: A two layer feed-forward neural network

Consider an initial forward feed of the neural network structure. For a specific input pattern (set of input values) output of j^{th} hidden layer is given

by,

$$h_j = f(\sum_{i=1}^{n_i} w'_{ji} x_i + b'_j) \tag{3.24}$$

where f is the activation function, w'_{ji} is the strength of connection from the i^{th} input to the j^{th} hidden layer node and b'_j is the bias value for the j^{th} hidden layer node.

The output of the k^{th} output node is given by,

$$y_k = f(\sum_{j=1}^{n_h} w_{kj} h_j + b_k) \tag{3.25}$$

where b_k is the bias for the k^{th} output node and w_{kj} is the strength of the connection from the j^{th} hidden layer node to the k^{th} output node.

Backpropagation, one of the popular training algorithms, uses gradient descent algorithm to update the weights and therefore the activation functions must be differentiable. Some of the activation functions for the neural networks are given in section 2.5.2. The result of the feed-forward process is the output pattern $y_1, y_2, ...y_{n_O}$.

In the training stage, the neural network uses input/output training sets to learn the functional mapping of the inputs to the outputs. Output training data is referred to the target output of the neural network. The goal is to train the network until the output of the neural network is close to the target output [29].

Training process goes until the output pattern is suitably close to the target pattern which is achieved by minimizing the sum-of-squares error (SSE) with respect to weight vector w, given of the form;

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{c} (y_k(x^n; w) - t_k^n)^2 \tag{3.26}$$

where

N= the number of training patterns

c= the number of outputs

x^n=input vector

t_k^n=target value for output node k when the input vector is x^n

Neural Network is trained by updating the weights using a backpropagation learning rule. Change in weight (w'_{ji}) is in the m^{th} iteration is given by:

$$w_{ji}^{(m)} = w_{ji}^{(m-1)} + \Delta w_{ji}^{(m)} \tag{3.27}$$

where

$w_{ji}^{(m)}$: the weight between the j^{th} node of the output layer and the i^{th} node of the hidden layer in the m^{th} training iteration.

$w_{ji}^{(m-1)}$: the weight between the j^{th} node of the output layer and the i^{th} node of the hidden layer in the $(m-1)^{th}$ training iteration.

$\Delta w_{ji}^{(m)}$: the weight adjustment

Weight adjustment is given by

$$\Delta w_{ji}^{(m)} = \eta \delta_j^{(m)} o_i^{(m)} + \alpha \Delta w_{ji}^{(m-1)} \tag{3.28}$$

η: learning rate,

$\delta_j^{(m)}$: error signal of the j^{th} node in the m^{th} training iteration

$o_i^{(m)}$: output value of the i^{th} node of the hidden layer in the m^{th} iteration

α: momentum term, $0 < \alpha < 1$

If j is an output layer node, $\delta_j^{(m)}$ is:

$$\delta_j^{(m)} = (t_j^{(m)} - y_j^{(m)}) \tilde{g}' (\sum_i w_{ji}^{(m)} o_i^{(m)} + w_{jo}^{(m)}) \tag{3.29}$$

where $t_j^{(m)}$: target value for output layer node j

$y_j^{(m)}$: network output value of node j

$\tilde{g}'$: derivative of output layer transfer function

$w_{jo}^{(m)}$: weight between the o^{th} node of the hidden layer and the j^{th} node of the output layer in the m^{th} training iteration.

If j is a hidden layer node,then we have:

$$\delta_j^{(m)} = g'(\sum_i w_{ji}^{(m)} o_i^{(m)} + w_{jo}^{(m)}) \sum_k \delta_k^{(m)} w_{kj}^{(m)} \tag{3.30}$$

where g′ is the derivative of the hidden layer transfer function.

Using error backpropagation algorithm error for each node is calculated and the weights of all nodes are recursively updated starting from the output layer to the hidden layer.

Usually, the nodes in the hidden layer and in the output layer employ the same transfer function, for instance log-sigmoid function for the hidden layer nodes and linear function for the output layer nodes.

3.4 Case Study: Water-Tank System

3.4.1 Purpose and Method of the Study

Generally, in closed-loop controlled systems, it is difficult or not possible to observe fault effect on the system, since the controller tolerates the faulty situation and attempts to bring the system to the desired operating point.

The main purpose of this study is to detect and identify the predefined faulty situations in the system. For this purpose the tank system is modeled using the qualitative techniques. Then the residuals obtained from the normal mode and the faulty modes are classified using neural network classifiers.

3.4.2 Process Description

The process under investigation is a water-tank system obtained from MATLAB/SIMULINK environment. The aim of this process is to model the

water-tank system and simulate it in closed-loop such that water level is at the desired position indicated by a reference. For this reason a PID controlled valve is used which allows the entrance of water into the tank (Figure 3.7). Also there is another valve at the bottom of the tank to deplete the water in the tank.

Consider an open water tank with cross-sectional area A (see Figure 3.5). Water is pumped into the tank through the valve at the top, at rate of flow of q_{in} cubic meters per second. Water is flowing out of the tank through a hole in the bottom of the tank of area a. The rate of flow of water through the hole is according to the Bernoulli equation given by,

$$q_{out} = a\sqrt{2gh} \tag{3.31}$$

where h is level of tank and g is the acceleration of gravity. Conservation of mass yields,

$$A\frac{dh}{dt} = q_{in} - q_{out} = q_{in} - a\sqrt{2gh} \tag{3.32}$$

This relation shows the nonlinear behavior with dynamic characteristics of the system depending on the operating point. It depends on the direction of the valve position changes (opening or closing). Valve is driven by the PID controller which controls the difference between the reference flow rate and the actual flow rate. The system has two outputs which are water level and instantaneous flow rate. The system is shown in Figure 3.6. Also the dimensions of the tank are given in Table 2. Initial water level in the tank is 0.5 meter and the reference point is changed randomly with a sample time of 50 seconds.

The problem is to detect and diagnose the faults in closed-loop operation of the system. The investigated cases are :

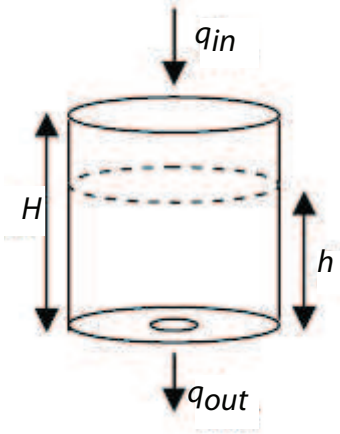


Figure 3.5: Water-Tank System Parameters

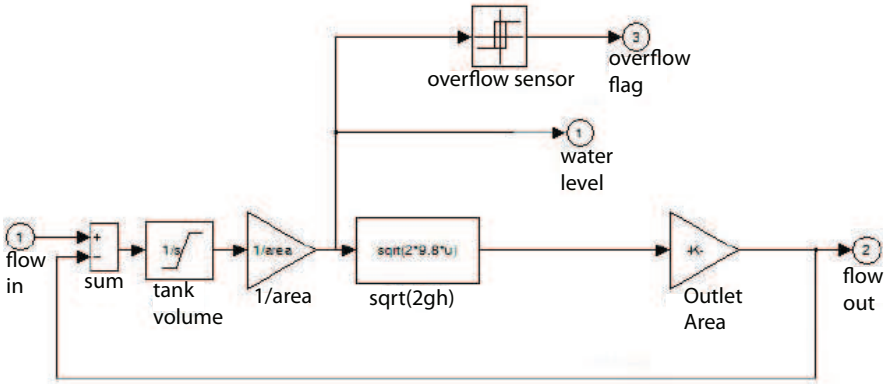


Figure 3.6: Water-Tank System Simulation

Table 2: Water-Tank Parameters

Height	2 m
Bottom Area	1 m^2
Outpipe Cross Section	0.01 m^2
Initial Level Height	0.5 m

F0: Normal operating condition

F1: Restriction at the output valve of the tank

F2: Leakage on the wall of the tank at a specific height from the bottom.

The faults affect the system in different ways. Generally the remarkable difference between the outputs of normal operating condition and neural network model which are called residuals indicate the fault alarm in the system.

3.4.3 Neural Network Process Model

As it is said in the previous section, residuals are necessary for the fault diagnosis process. Therefore, it is important to determine the current value of water level for normal operating conditions. A neural network can be used for this purpose. Analytically modeling is not preferred since the aim of this thesis is modeling of the automotive systems. Although these kind of systems are not complicated, their structure is commercially reserved.

The simulink model of the tank and controller is shown in Figure 3.7.

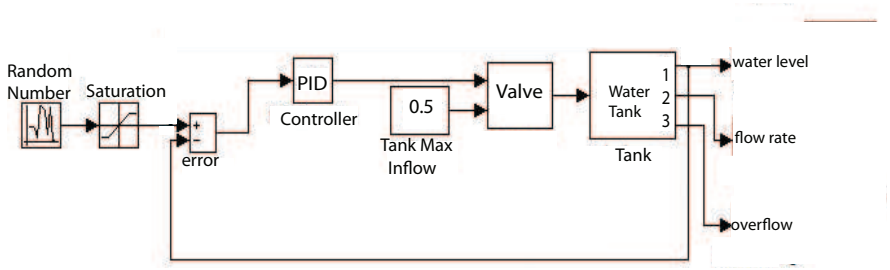


Figure 3.7: Water-Tank System in Closed-Loop

Modeling is done by using the Neural Network Toolbox of MATLAB. Since the actual system has two outputs, model of the actual system will also have two outputs which are water level and water flow rate from the

output valve. Therefore two neural network models are designed for this system both of which are three-input, one-output including three layers: input layer, hidden layer and output layer. In the input layer three neurons and in the output layer one neuron is used. In the hidden layer 25 neurons are used for both of the systems. Knowledge of the actual system, extensive training of different combinations of input variables and network topologies are utilized to identify the input to the neural network. The inputs represent a trade off between performance of the neural network under normal and faulty conditions. Input and output variables for training are;

Inputs = $u(i)$, $y(i-1)$, $\frac{dy}{dt}|_{(i-1)}$;

Output=$y(i)$;

where i is the current discrete time value, (i-1) refers to the previous value; u is instantaneous entering water flow rate, $y(i$-1) and $dy/dt|_{i-1}$ are one sample time delayed water level and output flow rate of the water. The sampling interval is taken as 0.1 second.

One of the neural networks will model the water level whereas the second one will model the flow rate of the water. The scheme of the neural network model is given in Figure 3.8.

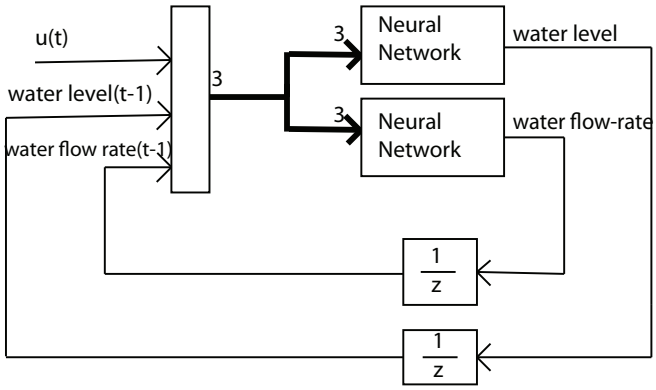


Figure 3.8: Neural Network Model of the Water-Tank System

The neural network is trained in a batch mode (offline) using experimental data obtained as the system operates in normal mode. The system is simulated for 800 seconds. Therefore, the training data set with 8000 input/output data is used until the average error for each training pattern is approximately 10^{-4}.

A tan-sigmoid activation function (2.37) is used for the hidden layer and a pure linear activation function (2.36) is used for the output layer. Matlab Neural Network Toolbox is utilized in the training process. The actual output vs neural network output of the system are plotted as a function of time in Figure 3.9 and Figure 3.10 .

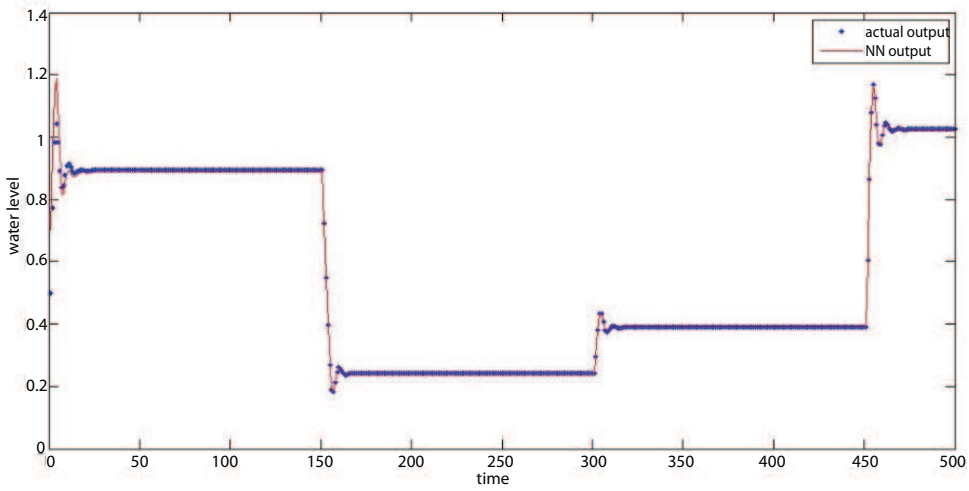


Figure 3.9: Neural Network Model Water Level vs Actual Water Level

3.4.4 Residual Generation Techniques

As it is mentioned, neural network model for the residual generation can be used for fault diagnosis purposes. Residuals are based on the comparison of features from the process with the nominal ones realized from the model.

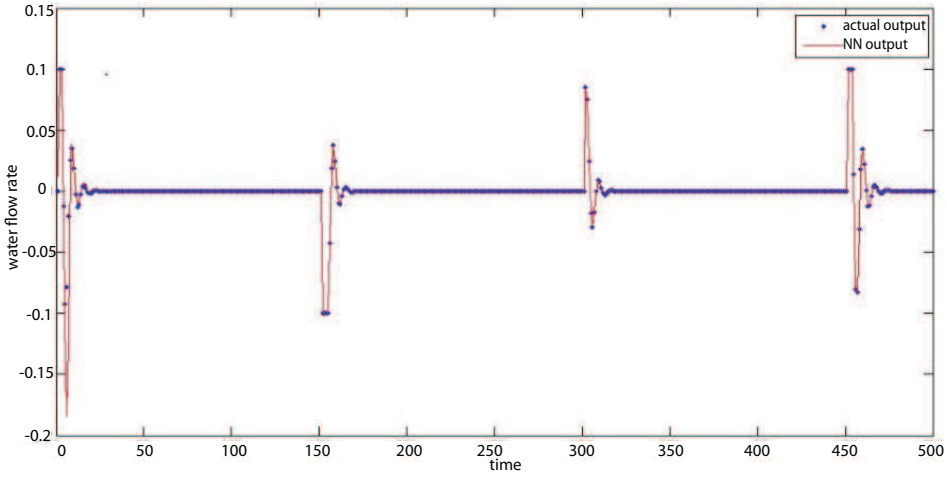


Figure 3.10: Neural Network Model Flow Rate vs Actual Flow Rate

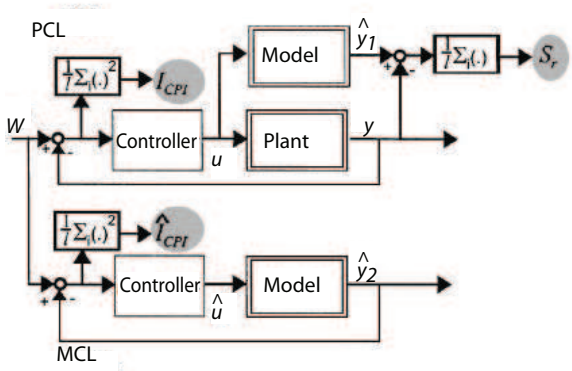


Figure 3.11: Residual Generation Using Model of the System

Simulation is performed with model in closed loop (MCL) control which runs in paralel to the process in closed-loop (PCL) using the same reference signal (Figure 3.11). For this case, two types of residuals are generated:

1. Output Based Residual

These types of residuals can be derived in both closed-loop and open-loop operation, and they do not require process excitation. It is the output error

between the process and the model within a time window of appropriate length l:

$$S_r = \frac{1}{l}\sum_{i=1}^{l} |\hat{y_1}(k-i) - y(k-i)| \tag{3.33}$$

For processes with multiple outputs, the residuals are decoupled from outputs [22].

2. Signal Based Residual

The performance of the controller in constant operation regions and during the set point changes affects the system behavior. Therefore the symptoms can be derived by defining different performance indices (CPI). In this case the difference between the control reference signal $W(k)$ and the controlled variable $y(k)$:

$$S_{CPI} = I_{CPI} - \hat{I}_{CPI} \tag{3.34}$$

$$= \frac{1}{l}\sum_{i=1}^{l}(W(k-i) - y(k-i))^2 - \frac{1}{l}\sum_{i=1}^{l}(W(k-i) - \hat{y}_2(k-i))^2 \tag{3.35}$$

In the constant operating regions S_{CPI} is also affected from noise and disturbances. Therefore, for comparable residuals these effects must be similar.

The next step of fault diagnosis is defining the fault-residual relationships. This can be solved by prior knowledge or from experiments.

3.4.5 Relationship Between Residuals and Faults

In the system fuzzy logic and neural network methods are used to relate the faults to the residuals.

Application of Adaptive Neuro-Fuzzy Logic as Residual Evaluators

For the purpose of relating the residuals with faults fuzzy logic method is used. In this method, the membership functions for the residual are built and for every faulty condition a chain of rules is chosen. By the way a database

relating the residuals to the faults is created. The conditions are normal condition, restriction at the output valve of the tank (fault 1) and a leak on the wall of the tank (fault 2) conditions.

For building the membership functions, residual data is classified as one dimensional. When there is a coincidence condition a new cluster is added. In this case, a data cluster which is changing from fault to fault is composed. The membership functions for the residual are shown in Figure 3.12. The values of the membership functions change between 0 and 1. Figure 3.12a corresponds to the membership function of the residual 1 (S_R) and Figure 3.12b corresponds to the membership function of the residual 2 (S_{CPI}). Those membership functions are defined with linguistic labels (small, big, normal) for the regions of residual data.

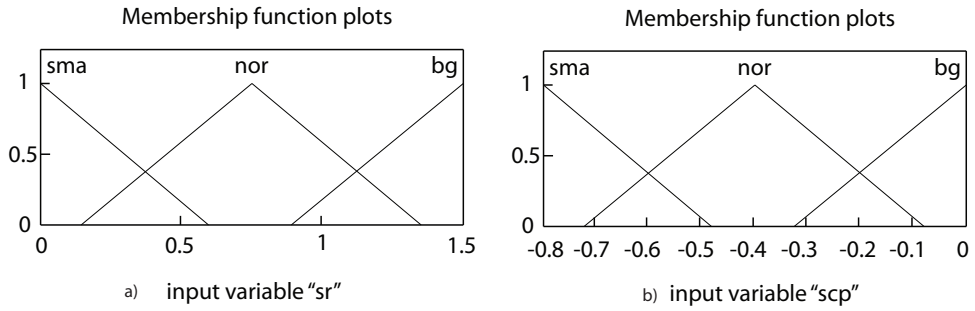


Figure 3.12: Membership Functions for residuals

In the selection of rules it is important to minimize the training dataset. When a rule is found to separate the fault from the training dataset, the data which is in relation with this fault is eliminated. This process is continued until a fuzzy logic rule is defined for every fault [30]. The rules obtained are defined in (3.36). These rules are tested in the simulation and verified.

1. IF (residual 1 "small" residual 2 "large") THEN no fault
2. IF (residual 1 "large" residual 2 "normal") THEN fault 1 (3.36)
3. IF (residual 1 "normal" residual 2 "small") THEN fault 2

Table 3: Fuzzy Logic Training Set

FAULT	RESIDUAL 1	RESIDUAL 2
0	Small	Large
1	Large	Normal
2	Normal	Small

From these rules a fuzzy logic table is generated for fault classification purposes which is shown in Table 3. Based on this table, a Multiple Input-Single Output (MISO) system having two inputs and one output is generated. Each input variable is represented by two membership functions which make four rules. The membership functions for this Takagi-Sugeno fuzzy model are chosen as triangular functions. The inputs are output based residual and signal based residual, and the output of the fuzzy block is assumed 0 for no fault condition and 0.5 for fault 1 condition, 1 for fault 2 condition (Figure 3.13). As seen in this figure, the outputs represent the fault with good accuracy. However, the coincidence of residuals gives wrong results similar to fault 2 condition in which some of the residual indicate no fault condition.

Application of Neural Networks as Residual Evaluators

It is also possible to apply neural networks for evaluating the residuals. In order to define the fault condition, the system is exercised under healthy condition and two faulty conditions which are mentioned in the previous

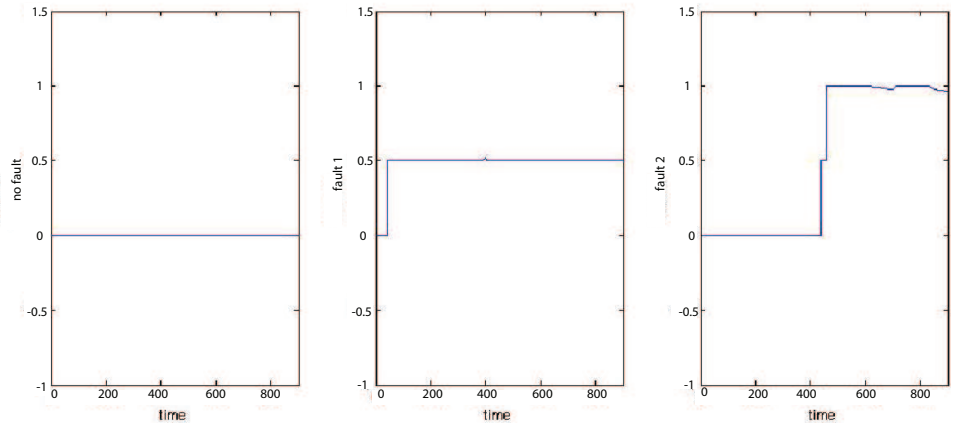


Figure 3.13: ANFIS Output for Fault Classification

section. To utilize the network in fault diagnosis process, the neural network is trained using these three conditions. The inputs of the neural network are signal-based and output based residuals and the output of the network constitutes a pattern that represents the normal mode or one of the two fault modes of operation. Therefore three input data set, which are available from the normal condition and fault conditions of the system, are used to train the network whose outputs are zero or one, mapped from the three cases shown in Figure 3.15. Since the output of the neural network is between zero and one, the rounding block of MATLAB library is used (Figure 3.14).

The neural network architecture is composed of 2 input nodes, 5 hidden layer nodes and 3 output layer nodes. A *tan-sigmoid* activation function is used for hidden layer and *purelin* activation function is used for the output layer. The network was trained until number of epochs reaches 5000.

The method described shows how neural networks can be used for fault diagnosis purposes. Model based approach compares the neural network

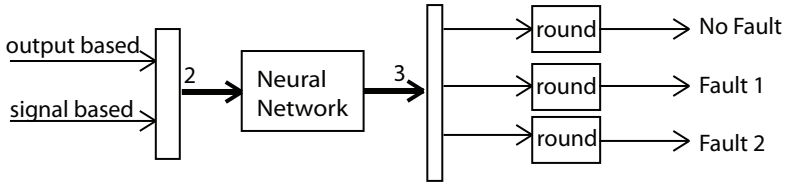


Figure 3.14: Neural Network Model for Fault Classification

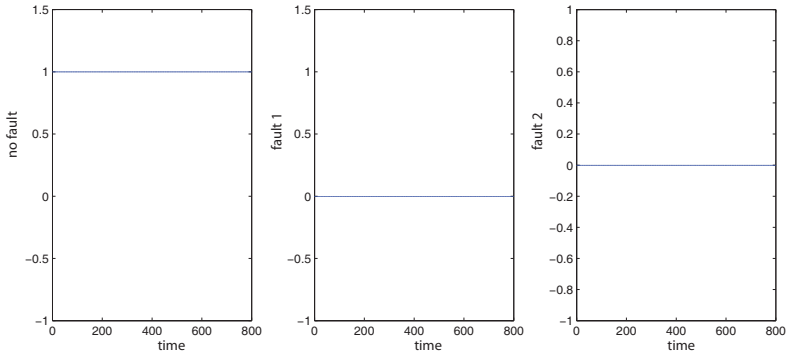


Figure 3.15: Neural Network Output in Fault Classification

model of normal plant operation to actual system operation to determine residuals under faulty conditions. Then the residuals are classified using ANFIS or neural networks.

3.5 Results

In the model based fault diagnosis of nonlinear dynamic systems, residuals are necessary which compares the actual system with the analytical model. Since it is difficult to mathematically model the water-tank system, neural network modeling technique, which mimics the actual system, is used.

Actual system has two outputs which are water level and flow rate of water. Therefore two neural network models with three layers and 25 nodes

are generated which work in coordination.

In the training stage, gradient descent backpropagation algorithm is utilized to update the weights in each layer. Chosen activation functions are tan-sigmoid and pure linear functions for the hidden layer and the output layer, respectively.

In residual evaluation stage of the nonlinear system, neural network and fuzzy logic are used. Misclassification rates and computational times of both approaches are given in Table 4.

Table 4: Comparison of NN with FL

	NN	FL
Elapsed Time(s)	1029.51	69.97
Misclassification Rate for F1	1.637e-3	4.8e-2
Misclassification Rate for F2	1.273e-4	2.375e-4

Comparing the performances of the neural network and fuzzy logic in residual evaluation process, it is observed that the neural network gives better results. However the computation time of the neural network takes longer than the fuzzy logic.

3.6 Conclusions

In this section some system identification and fault diagnosis methods have been introduced and a fault diagnosis approach for the nonlinear systems is applied to the water-tank system. The aim of this study is to illustrate the fault diagnosis methods on the simulation of the system. Therefore two different fault scenarios are generated. One of them is opening a hole on the surface of the wall and the other is the restriction at the output valve.

Initially the system is exercised under normal conditions to generate a neural network model of the actual plant which run in parallel. The training data set of this network is the input and output data of the actual system.

Based on the nominal process model, two different residuals are generated in closed-loop operation. Based on the difference between the model and the system outputs, two types of residuals are defined which are the output based and signal based residuals. Based on these residuals a neural network model and fuzzy logic model are generated for the residual evaluation process. Their performances and misclassification rates are compared. The faults are detected and identified correctly.

Chapter IV

4 Fault Diagnosis Based on Classification Tree and Fisher Discriminant Analysis

4.1 Introduction

Classification trees (decision trees) were first introduced by social scientists in the early 1960s [31]. In the 1980s Breiman et al [32] proposed classification and regression tree methodology for the analysis of large data sets with binary recursive partitioning procedure. Classification tree analysis is a clear and fast computing method that makes no assumption about the distribution of the predictor variables.

In this section process fault diagnosis scheme will be applied to rechargeable lead acid batteries based on classification trees. The purpose of utilizing the classification tree is reducing the amount of data to achieve good learning, classification accuracy, compact and easily understood knowledge-base, and a reduction in computational time [33].

This new approach is an integrated method which combines the classification and regression trees (CART) with neural networks for fault diagnosis of lead acid batteries. The proposed approach has three main steps. First the neural network modeling of the real-time system is performed. Then residuals are obtained from the difference between the system and the model output. Finally the classification tree is performed on the generated resid-

uals to diagnose the faults of lead-acid batteries which is also integrated with Fisher Discriminant Analysis (FDA). Generally, classification tree with FDA has longer training time and higher tree size compared to the normal classification tree but it has lower error rate.

Evaluation is done experimentally. Actual data of a healthy lead acid battery is acquired. Then faults are introduced into the battery and new data is recorded. The analysis is performed on the gathered data.

In the next subsection the classification tree principles are reviewed and the new analysis procedure is applied to residuals. Rechargeable Lead Acid Batteries are presented as a first case study and performances of classification trees with and without FDA are compared. The same procedure is applied on mass-spring-damper systems as a second case study. Conclusions are given in the final section.

4.2 Classification Tree Principles

Classification tree is a form of binary partitioning algorithm [32] which is similar to those used in decision tree induction such as ID3 and C4.5 [34]. However, classification tree splits the training samples into smaller and smaller subsets recursively. The trees produced by CART consist of internal nodes (each of them with two children) and terminal nodes or leaf nodes (without children). Each internal node uses a decision function to indicate which node to visit next, whilst each terminal node shows the output of a given input vector which leads the visit to this node [35]. The decision tree shown in Figure 4.1 shows the classification regions resulting from analysis of a set of process data. The training data contains three classes: normal (f1), fault 1 (f2), fault 2 (f3). At first, all samples are assigned to one node. The samples in the first node are divided into two groups according to the property of the

first residual (R1). The tree looks for the most suitable threshold to separate each group and tries to find the purest node possible. For the second node separating data based on property of the second residual (R2), gives purer child nodes. Finally the partition stops when this data set has been split into three pure sets where *pure* means that no sets contain points belonging to another fault.

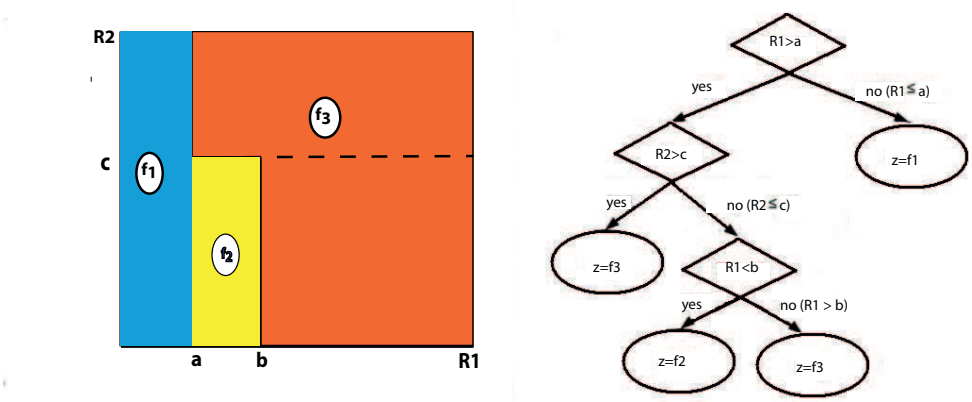


Figure 4.1: Simple Classification Tree for Illustration

The fundamental principle of classification tree is to reach the most informative property that makes the data reaching intermediate child nodes as pure as possible based on a minimum cost-complexity principle. The first phase is called *tree building* and second phase is called *tree pruning*.

4.2.1 Tree Building and Tree Cost

The initial state of a decision tree is the root node (the first internal node) which assigns all examples of the training set. If all examples are in the same class, no further decision is required and solution is completed. If the examples at this node belong to two or more classes, a test is made

to split the training set into two sub-spaces based on a threshold value of a single variable. The process is recursively repeated for each of the new terminal nodes until a completely discriminating tree is obtained.

The best known impurity measure of impurity is entropy impurity index. Entropy impurity index denotes the tree cost which has the formula as shown in (4.1):

$$E(t) = -\sum_{j} p(j|t) log_2 p(j|t) \tag{4.1}$$

where $E(t)$ shows the impurity of a node t and p(j|t) is the portion of observations at node t belonging to class j. If all the observations are in the same class the impurity index is 0; otherwise it is positive and the maximum value occurs when the different classes are equally possible.

Another impurity measure is the Gini diversity index [32] that describes the expected error at node t if the node label is selected randomly from the class distribution present at that node. The node cost is formulated as

$$i(t) = \sum_{i \neq j} p(i|t) p(j|t) \tag{4.2}$$

where $i(t)$ is the Gini diversity index of the node t and p(i|t) is the portion of observations in node t belonging to class i. p(j|t) is the portion of observations in node t belonging to class j.

Although the optimization is performed at a single node, the recursive splitting process can go until each leaf node becomes perfectly pure. The impurity measure of the tree which is defined below becomes zero if each node corresponds to a single training sample.

$$I(T) = 1 - max_j[p(j|t)] \tag{4.3}$$

where T is the classification and regression tree.

4.2.2 Optimal Tree Size Decision using Cross-Validation and Threshold Value

When the classification tree is fully-grown and each leaf node achieves zero impurity, the tree overfits the training data meaning that tree represents the data explicitly and generalization or noise immunity performance is not good; the tree is also large. However if the partitioning stops too early, the error on the training data is not low enough and the performance on the new data may not be sufficiently good.

Cross validation is a general approach to decide on the optimal tree size. The training dataset is randomly split into N subsets and one of these subsets is reserved as an independent test dataset and the other N-1 subsets are combined as the training dataset. Trees with different sizes are tested and at each size, N trees are generated, with a different subset of the data reserved as the test dataset each time. Therefore N different trees are generated each of which is tested against its corresponding test dataset. The average performance of N trees is an excellent estimate of the performance of the original tree which is entire training set. The average performance of these trees is compared and the one with the lowest prediction error is selected as optimal tree size.

Setting a threshold to the node impurity index is also used to reduce the tree size. If the impurity reduction is less than this threshold value the splitting stops at that node. This approach uses all the training data set and generates classification trees with balanced leaf node impurity. The major disadvantage is that it is often difficult to determine the threshold value, because the relationship between the threshold value and the tree performance is rarely simple.

4.2.3 Tree pruning

Another approach for the optimum classification tree is tree pruning method. The tree obtained from building phase utilizes the training dataset [36] and may have a large number of branches which increases the tree complexity. Therefore it is necessary to prune the tree to improve the accuracy of the classifier and overcome the overfitting problem.

Pruning algorithm is based on the misclassification rate. Assume that T_{max} is the fully-grown tree obtained in the building phase. Entropy impurity measure (or cost-complexity measure), $E_\alpha(T)$ of subtree $T \subset \mathrm{T}_{max}$ is defined as:

$$E_\alpha(T) = E(T) - \alpha|\tilde{T}| \tag{4.4}$$

where $\tilde{T}$ is the set of terminal nodes in T, $|\tilde{T}|$ is the number of terminal nodes in T and α is a complexity parameter which is denoted by α_t in (4.5) as:

$$\alpha_t = \frac{E(t) - E(T_t)}{|\tilde{T}_t| - 1} \tag{4.5}$$

The aim in the pruning process is to find the minimal α_t parameter which makes T-T_t as the next minimizing tree for each internal node t. This parameter is recursively updated until the optimum tree size is achieved by using an independent test data set or performing cross-validation which will determine the best degree of pruning coming with a huge computational overhead.

Pruning of tree reduces the possibility of missing classification functions close to the leaves for a fully-grown tree. Pruning used with cross validation technique determines the best degree of pruning and decrease the computational cost of the tree significantly.

4.2.4 Leaf Node Label Decision

Assigning class label to leaf nodes is simple: if the classification tree is pruned to its optimal depth, it is most likely that each leaf node has zero or a small positive impurity index. If the leaf node has only observations from one class it will be labeled as that class otherwise it will be labeled by the class that has most observations represented. Therefore every leaf node with a large amount of same class observations will be generated.

The final decision tree is a representation of input/output mapping. For instance, the decision tree shown in Figure 4.1 is equivalent to a set of crisp rules,

$$\begin{array}{lll} \text{If } R1 < a & \text{then} & z = f_1 \\ \text{If } R1 > a \text{ and } R1 \leq b \text{ and } R2 \leq c & \text{then} & z = f_2 \\ \text{If } R1 > b \text{ and } R2 > c & \text{then} & z = f_3 \end{array} \tag{4.6}$$

4.3 Implementation of Classification Tree

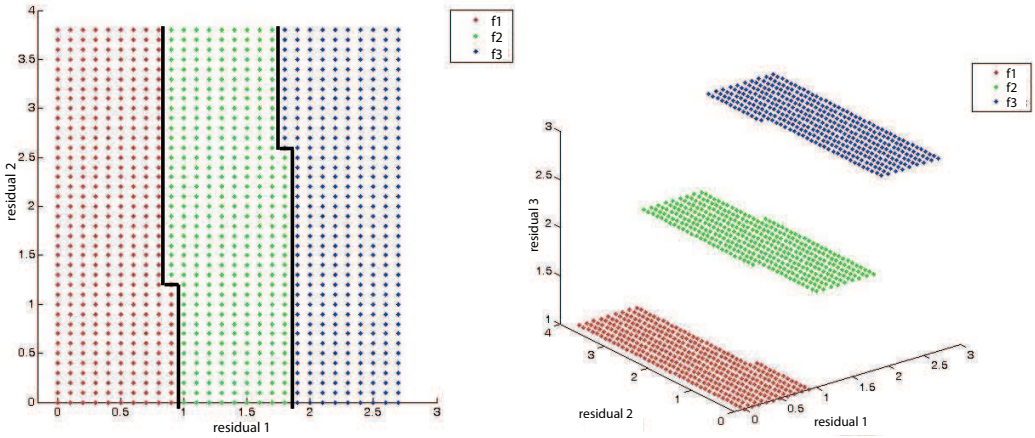
The significant benefit of classification tree is that it is easy to render the information on the tree as logical expressions so that the relationship between the prior knowledge and the result of the tree can be obtained easily.

The performance of the classification tree highly depends on the quality and quantity of the original training dataset. The faults on the tree must match the faults on the training set from which the tree is built.

4.3.1 Decision Boundary Generation

The classification tree creates decision boundaries (based on the information of the nodes) with portions perpendicular to the property axes as shown in Figure 4.2b. If the tree is sufficiently large, any decision boundary can be

approximated well, provided that enough training data is present.



(a) Decision Regions for 2-D Residual Space (b) Decision Regions for 3-D Residual Space

Figure 4.2: Decision Regions Created By Classification Tree

It is possible to obtain unnecessarily complicated decision boundaries which do not align with the property axes. An obvious example is shown in Figure 4.3. The simple decision boundary has to be approximated with segments of lines. However using a linear combination of the variables it is possible to result in a much simpler tree. For this reason Fisher Discriminant Analysis (FDA) finds the optimal combination.

4.3.2 Computational Efficiency

Building a classification tree is computationally expensive. Assume that there are n training patterns and the dimension of the patterns is d. The computational complexity of a fully-grown tree is represented as follows.

At the root node, the training pattern must be sorted on each of the d dimensions. It takes $O(dn\ log(n))$. Calculating impurity index takes $O(dn)$, therefore, the computational cost of the root node is $O(dn\ log(n))$. Since there are two nodes at the next level, and each node takes $O(\frac{1}{2}dn\ log(\frac{n}{2}))$, the

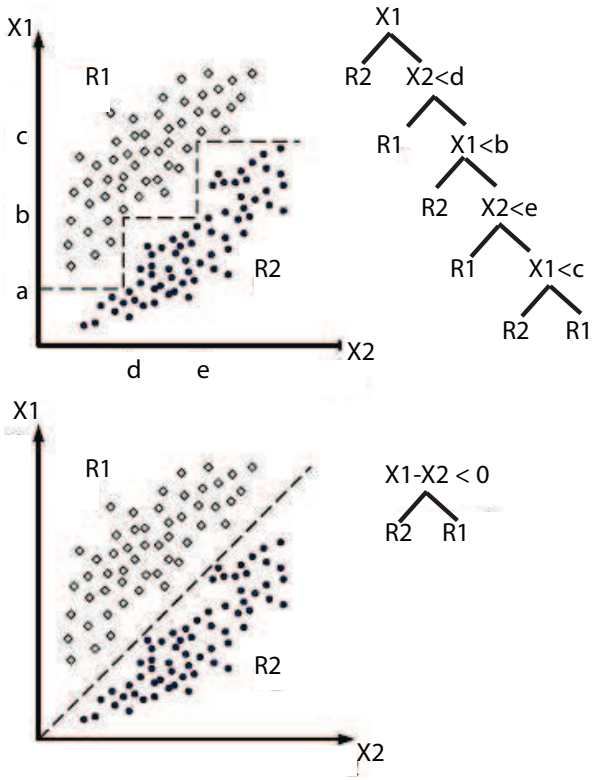


Figure 4.3: Simple Decision Boundary for Optimal Tree

computational cost of level 2 is $O(dn\ log(\frac{n}{2}))$. Moreover the computational cost for level 3 is $O(dn\ log(\frac{n}{4}))$ and for level 4 $O(dn\ log(\frac{n}{8}))$, and so on. Since the depth of the tree is $logn$, by summing up the cost of each level, the total cost of the tree is $O(dn\ (logn)^2)$.

In the process of pruning a classification tree, cross validation method is generally applied to determine the optimal tree depth. For instance, one wants to examine the classification performance of D different depths of tree and divides n training patterns into M groups. For each depth, M classification trees are built and their performance evaluated, which takes $O(Md(\frac{M-1}{M})n\ (log(\frac{M-1}{M})n)^2)$, or $O(d(M-1)n\ (log(\frac{M-1}{M})n)^2)$. A total of

D different depths of trees are examined which results in a computational cost of $O(dD(M-1)n\ (log(\frac{M-1}{M})n)^2)$. Assuming that M>>1, the step of determining the tree depth is $O(DMdn\ (\text{logn})^2)$.

4.4 Generation of Classification Tree with Fisher Discriminant Analysis

In pattern classification literature, dimensionality reduction is an important factor when the dimension of the observation space is large while the number of observations is relatively small [37]. Also the computational requirements are greatly reduced for some applications like neural networks when the training data is proportional to dimensions of the process data.

One of the most important dimensionality reduction technique is Fisher Discriminant Analysis which takes into account the information between the classes and provides an optimal lower dimensional representation in terms of discriminating among classes of data [38]. FDA determines a set of projection vectors which maximizes the scatter between the classes while minimizes the scatter within each class.

Let's define n as the number of observations, m as the number of measurement variables, p as the number of classes and n_j as the number of observations in the j^{th} class, $x_i \in R^m$ represents the vector of measurement variables for the i^{th} observation. If the training data for all classes have been stacked into the matrix $X \epsilon R_{nxm}$, then the transpose of the i^{th} row of X is the column vector x_i. To perform FDA, it is necessary to calculate the total-scatter, the within-class scatter and the between-class scatter. The total scatter matrix is

$$S_t = \sum_{i=1}^{n} (x_i - \bar{X})(x_i - \bar{X})' \tag{4.7}$$

where $\bar{X}$ is the total mean vector

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.8}$$

Let X_j be the set of vectors x_i that belong to class j, the within-class scatter matrix of class j is

$$S_j = \sum_{x_i \epsilon X_j} (x_i - \bar{x_j})(x_i - \bar{x_j})' \tag{4.9}$$

where $\bar{x_j}$ is the mean vector of class j

$$\bar{x_j} = \frac{1}{n_j}\sum_{x_i \epsilon X_j} x_i \tag{4.10}$$

The within class scatter matrix is

$$S_w = \sum_{j=1}^{p} S_j \tag{4.11}$$

and the between class scatter matrix is

$$S_b = \sum_{j=1}^{p} n_j(\bar{x_j} - \bar{X})(\bar{x_j} - \bar{X})' \tag{4.12}$$

The total scatter matrix is equal to the sum of the between-class scatter matrix and the within-class scatter matrix

$$S_t = S_b + S_w \tag{4.13}$$

FDA is given by a vector $v \in R^m$ which maximizes the scatter between the classes whereas minimizes the scatter within classes.

$$J(v) = max\frac{v'S_b v}{v'S_w v}, v \neq 0 \tag{4.14}$$

The second FDA vector is computed so to maximize the scatter between the classes while minimizing the scatter within classes among all axes perpendicular to the first FDA vector and so on for the remaining FDA vectors. FDA vectors are equal to the eigenvectors v_k of the generalized eigenvalue problem.

$$S_b v_k = \lambda_k S_w v_k \tag{4.15}$$

where the eigenvalues λ_k indicate the degree of overall separability among the classes by projecting data onto v_k. Let's define matrix $\mathrm{W}_p \epsilon R^{mxp}$ with the p FDA vectors as columns. Then the projection of the data from m-dimensional space to p-dimensional space is described by

$$z_i = W_p' x_i \tag{4.16}$$

While generation of classification tree with FDA method, FDA extracts the most significant scores in the original process data and achieves optimal discrimination among different faults. The classification tree uses FDA results to separate the observations into different classes with lower dimensional representation.

4.5 Case Study 1: Rechargeable Lead Acid Battery

4.5.1 Lead Acid Battery Principles

The Lead Acid Battery is an integral part of an automotive electrical system for many decades and fundamentals of the lead acid battery technology has not changed.

Essentially it consists of two electrodes immersed in sulfuric acid electrolyte. Some modern techniques include adding valves to the battery and

immobilizing the electrolyte (by gelling using silicon dioxide [39]), allowing it to be used in any orientation, also allowing it to be "sealed" and maintenance-free. Generally these type of batteries are called sealed valve-regulated lead-acid (VRLA) batteries and used in automotive starting, lighting and ignition (SLI) applications. The main purpose of the battery is starting the engine in a vehicle and second one is to provide power during high transient loads and maintain charge be fulfilled.

The basic structure of a lead-acid battery consists of two electrodes; negative electrode attached to a spongy active mass and positive electrode attached to a porous grid containing granules of metallic lead dioxide. These two materials are arranged in a matrix and immersed in concentrated sulfuric acid electrolyte to provide the mobile positive and negative charges. The matrix comprises a cell, several of which are placed in series to form the battery [40]. Modern SLI batteries contain six cells, each with a nominal voltage of 2.1 V.

The redox reactions at the electrodes during discharge are given in (4.17) [41].
Positive Electrode:

$$\begin{aligned} PbO_2 + 4H^+ + 2e^- &\rightarrow Pb^{2+} + 2H_2O \\ Pb^{2+} + SO_4^{2-} &\rightarrow PbSO_4 \end{aligned} \tag{4.17}$$

Negative Electrode:

$$\begin{aligned} Pb &\rightarrow Pb^{2+} + 2e^- \\ Pb^{2+} + SO_4^{2-} &\rightarrow PbSO_4 \end{aligned} \tag{4.18}$$

In the discharge case, the positive electrode (anode) accepts electrons and becomes oxidized whereas the negative electrode (cathode) gives up electrons i.e., is reduced. In the charge case this process in reversed. Therefore it can be said that the movement of charges uses the dissolving and precipitating of charge-carrying ions through the electrolyte.

In the next section some basic terminology will be given about lead-acid batteries.

4.5.2 Lead Acid Battery Characteristics

In this section the characteristics and parameters of the lead-acid battery will be given as they affect the modeling process.

Rechargeable vs. non-rechargeable

Rechargeable battery (secondary battery) is a kind of battery which is designed to allow the charging chemical reaction efficiently as it may be recharged by applying the current. All automotive batteries are these kind of batteries. Primary battery is a type of battery which is not rechargeable.

Capacity

Capacity is the amount of charge that can be drawn for some length of time before the battery is considered as discharged.

Theoretically for a battery rated 10 Amperes-hour(Ah), a discharge current of 1 Amper (A) should deplete the battery in 10 hours or 10 A in 1 hour. However due to the nonlinear behavior of available capacity as a function of discharge current, these expectations are not accurate.

Automotive manufacturers specify a term called *reserve capacity* in terms of hours during which a fully-charged battery can be discharged at constant current without terminal voltage dropping below 10.5 V [39]. At this voltage the battery is assumed to be completely discharged.

State of Charge

State of Charge(SoC) is a measure of how much current the battery can deliver after partial discharge or charge. It is difficult to determine reliably and precisely due to many nonlinear effects in batteries.

Integrating the amount of current that entering or leaving the battery

should allow good estimation of SoC, based on the equation,

$$SOC = 100(1 - \frac{\int_0^t I_B(t)dt}{Q}) \tag{4.19}$$

where Q is nominal capacity in Ah, t is independent time variable in h, I_B is the battery current in A with negative defined as discharge.

Surface Charge

This is a major cause of nonlinearity in lead-acid batteries that does not exist in modern battery technologies. It refers to the phenomenon resulting from plates comprising the positive and negative electrodes being of finite thickness. It causes a battery recently discharged partially, incorrectly appear to be exhausted(indicated by terminal voltage) or a battery recently charged to a small SoC to be fully charged.

The electrochemical reaction that produces electricity takes place only at the interface of the electrode material and the electrolyte. Therefore when the battery is being charged, the charge accumulated on the surface must diffuse into the plates of electrodes. Since this happens in a solid, it is slower than the charge carrying ions diffusing through the electrolyte.

Conventionally, to check the battery's SoC, it must be released for 4 to 12 hours, and then it must be discharged at approximately 33% of its rated capacity for 5 minutes to remove any surface charge. After waiting for 10 minutes, terminal voltage of the battery must be read and referred to a table. This process linearly correlates the battery's SoC to its open-circuit voltage.

Surface charge is important when the charging behavior of the lead-acid batteries is modeled whereas in other battery types due to their different or more advanced design it is not a parameter.

4.5.3 Process Description and Data acquisition

The main objective of the study is to find whether the battery is in good condition or in faulty condition preferably during actual use. If it is in faulty condition then the aim is to discriminate the faults. This section is focused on the classification tree for fault diagnosis of lead-acid battery. Referring to Figure 4.4 sensor measurement and data acquisition will be discussed under experimental equipment and procedure.

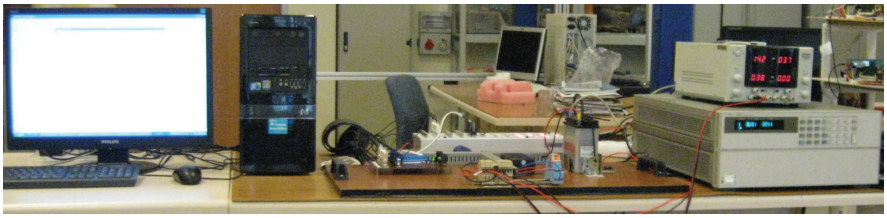


Figure 4.4: Experimental Setup of Lead-Acid Battery

Experimental Equipment

The experimental setup is composed of the following:

- 12V 5Ah lead-acid battery with 6 cells.
- Agilent N3300 1.8 kW programmable electronic load, capable of drawing 120 A at 240V.
- Topward 6303D Digital Display manually adjustable current limited Laboratory DC Power Supply(source), capable of delivering 5 A at 5V up to 30 V.
- GT Power A-6 charger with maximum output power of 600 W and maximum current of 10A which can charge/discharge up to 18 cells in series.

- DSPACE 1104 fast control loop prototyping system that is used for data acquisition, setup control and for electronic communication commands to the load using RS232 protocol.
- MATLAB R2009a and SIMULINK which is interconnected with DSPACE and desktop PC.
- Voltage divider circuit for the DSPACE and relay circuit.
- 5A DC industrial Relay contactor which will isolate the battery when not being charged or discharged.
- Three Hall effect current sensors that measure current through the cables connected to the positive terminal of the battery, source and electronic load.
- Desktop PC for control of the experimental setup.

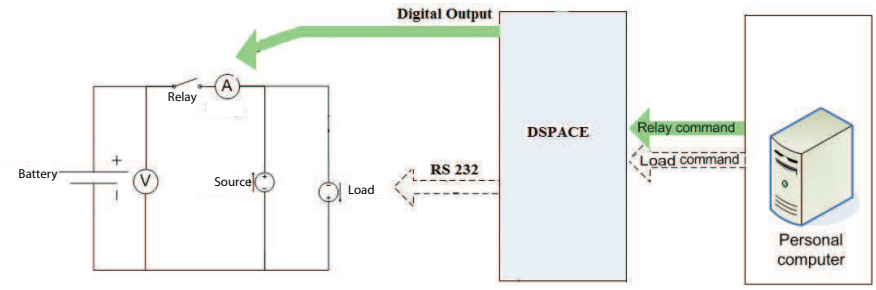


Figure 4.5: Schematic Diagram of the Experimental Setup

Process Description and Data Acquisition

The process involves SoC control of rechargeable lead-acid battery circuit. As shown in Figure 4.5, the battery is connected to a power source and an electric load. The load is controlled from the RS232 signals transmitted by the DSPACE environment.

The main purpose of the lead-acid battery system is to characterize the battery, especially terminal voltage under various SoC and load (current) conditions. Therefore a special discharge method is devised where the real-time system is integrated with Matlab-Simulink toolboxes and DSPACE libraries. Assumptions used in this process are as follow:

- Internal resistance is supposed to be constant during the charge and discharge cycles and does not vary with the amplitude of the current.
- The capacity of the battery does not change with the amplitude of the current.
- The temperature does not affect the battery's behavior.

Under these assumptions closed-loop control system is built to control the SoC of the battery. Such systems are particularly useful in hybrid vehicle applications where battery SoC is controlled according to anticipated load (incline, traffic jam etc.).

This system is exercised for nearly seven hours with a sampling time of one second. A large amount of data, consisting of current, voltage and SoC measurements of the battery, is obtained.

As it is seen in Figure 4.6, at t=0 the SoC value of the battery is 100% which means that the battery is fully charged. Suddenly the SoC value is pushed to be 65%. Therefore the battery is discharged until the SoC value catches the reference point(at t=4900 s). When the reference point becomes 80% battery starts to be charged and SoC value increases etc. The operating point of the process is changed after every 6000 second interval.

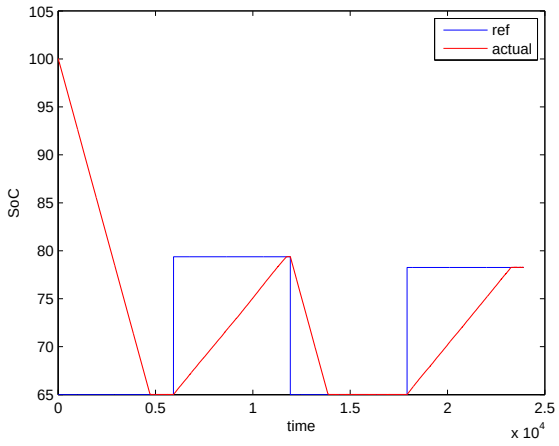


Figure 4.6: Reference SoC vs Actual SoC

4.5.4 Neural Network Process Model

To build the neural network model of the lead-acid battery, data obtained during the real-time process is used. Two neural network blocks are generated one of which is for the modeling of the output voltage and the other one is for the modeling of SoC value. In residual generation and feature extraction process, which will be explained in the next subsection, neural network model of the output voltage value will be used(Figure 4.8).

In the training stage of the neural network, all training data(obtained under normal operating conditions) are scaled to the range of [0,1] prior to the training process. The networks are trained using a Levenberg-Marquardt optimization algorithm [42]. The input layer has tan-sigmoid function where as output layer has pure linear function as shown in (1.25) and (1.27). 500 epochs are used for training stage and in the hidden layer 30 neurons are used.

In the modeling of the output voltage, a multi layer feed forward neural

network with one hidden layer is used to develop the nonlinear representation of the lead-acid battery simulation. A three input one output feed forward neural network is implemented offline with the Matlab Neural Network Toolbox. Input and output values for training are;

Inputs= $I_B(i)$; V_B(i-1); SoC(i-1)

Output= $V_B(i)$

where i is the current discrete time value, (i-1) refers to the previous value; $I_B(i)$ is the battery input current; $V_B(i\text{-}1)$ and SoC(i-1) are one sample time delayed battery voltage and battery state of charge values. Neural network voltage output is shown in Figure 4.7. The sampling interval is taken as 0.1 second.

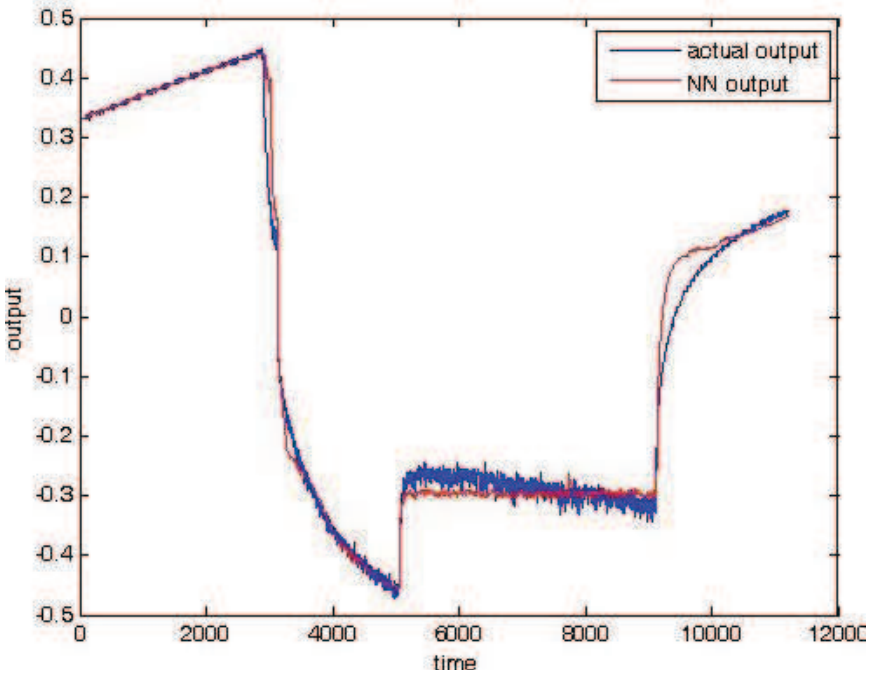


Figure 4.7: Neural Network Output vs Actual Battery Voltage

The Matlab model of the system running in parallel with neural network

model is shown in Figure 4.8.

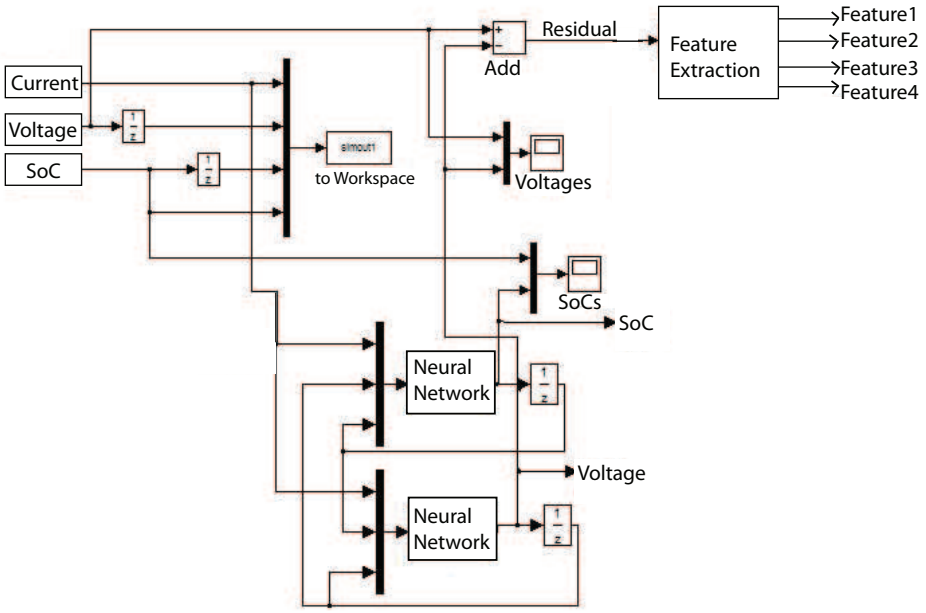


Figure 4.8: Lead-Acid Battery Model from the Actual Data

4.5.5 Residual Generation and Feature Extraction

After training the neural network for modeling the output, residuals are generated and features are extracted for fault detection and diagnosis purposes.

$$x(t) = y_{actual} - y_{model} \tag{4.20}$$

Assume that x is a residual based on (4.20). Statistical methods have been widely used in fault diagnosis which can provide the physical characteristics of the nonlinear time based residual. Statistical analysis yields different statistical parameters which are selected as basis for this study. These are;

RMS Value:

RMS value is a measure of the magnitude of the varying quantity.

$$X_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{4.21}$$

where x_i $(i = 1, .., N)$ is the amplitude at sampling point i and N is the number of sampling points.

Kurtosis Value:

This value indicates the flatness or spikiness of the signal. It is low for normal condition and high for faulty condition due to spiky nature of the signal.

$$X_{K_V} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^4}{X_{RMS}^4} \tag{4.22}$$

where

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.23}$$

Power:

This is a measure of effective energy or power content of the signal associated with average value of the signal.

$$P(x) = \bar{x} + 0.5\sigma_x \tag{4.24}$$

where

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{4.25}$$

The lead-acid battery system is simulated in real-time for three different cases. These are system under normal operation, fault one and fault two cases. Fault one case is reduction of electrolyte in the battery and fault two case is irreversible damage caused by leaving battery at deep discharge for 6 days. As it is denoted in Figure 4.8, the system works in parallel

with neural network model and residual data is obtained for those three cases. Applying (4.21), (4.22), (4.24) on the difference between the actual output voltage and neural network model voltage, three different residuals are extracted. By this way, three dimensional representation of the residual space is obtained(Figure 4.9). There are some overlaps for fault 1 and fault 2 cases.

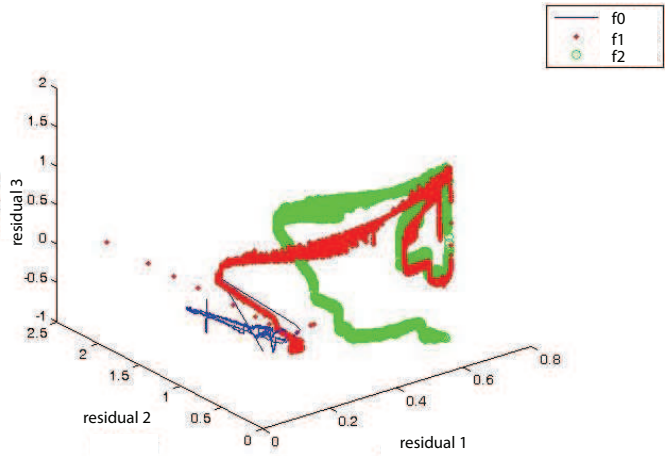


Figure 4.9: Three Dimensional Residual Space

4.5.6 Residual Evaluation with Classification Tree

As a final step in the fault diagnosis procedure, residuals generated in the previous section are evaluated and classified to the fault classes by using classification tree approach.

The maximum tree (every leaf node containing only one class) is created with the training dataset based on their respective fault classes (see Figure 4.10). In this tree, training data is classified into different classes for fault detection. It is obvious that the maximum tree fits training data well but have problem with the test dataset since the lower branches may be

affected by the process noise of training data as explained in section 4.2.2.

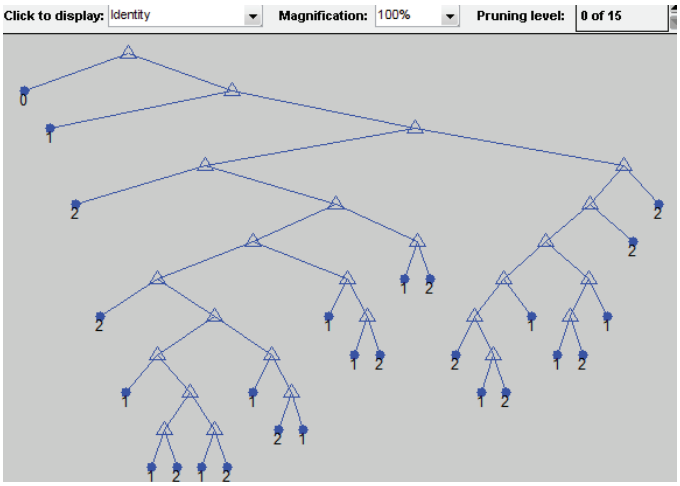


Figure 4.10: Classification Tree Before Pruning

Cross validation technique (defined in 4.2.2) is applied to estimate the optimal tree size. The procedure for this technique is as follows:

1. Divide data into 10 mutually exclusive subsets of approximately equal size(S_0, S_1,..,S_9).

2. Drop out each subset in turn, build a tree using data from the remaining subsets, use it to predict the responses for the omitted subset.

3. Calculate the estimated error for each subset (e.g. for sum of least squares regression tree, the error is the sum of squared differences of the observations and predictions) and sum overall subsets.

4. Repeat steps 2 and 3 for each size of tree.

5. Select the estimated tree with the smallest error rate.

Although chosen randomly, dividing into groups based on the value of the response variable gives smaller and more accurate estimates of the true error rate [32].

First "resubstitution error" is computed which means the proportion of original observations that are misclassified by various subsets of the original tree. Then "cross validation error" of various tree sizes is calculated and plotted in Figure 4.11. As the tree size increases, the error rate decreases.

From the figure, *best tree* is taken as the smallest tree such that its estimated error rate is within one standard error of the minimum. The standard error of the estimate is calculated for each tree size.

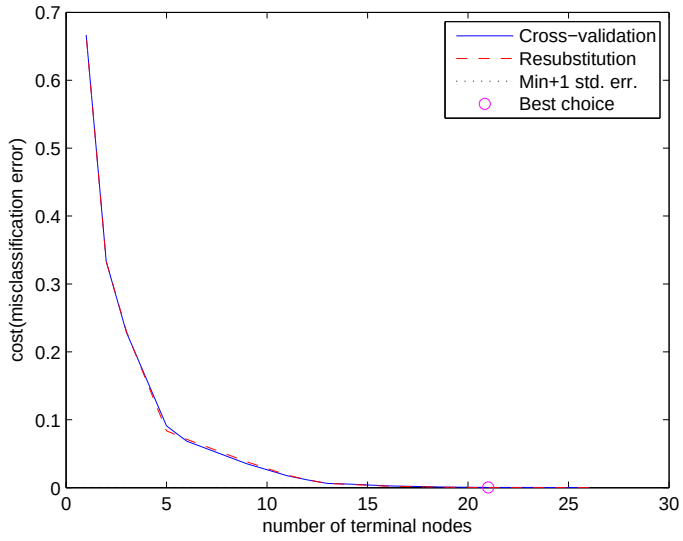


Figure 4.11: Cross Validation Error of Maximum Tree

This is shown on the graph by computing a cutoff value that is equal to the minimum cost plus one standard error. (best level=0 corresponds to the unpruned tree, so 1 is added to use it as an index into the vector outputs from the tested tree.)

According to the best tree size, the maximum tree is obtained which is shown in Figure 4.12. After the pruning stage, the tree size is decreased while decreasing the error rate.

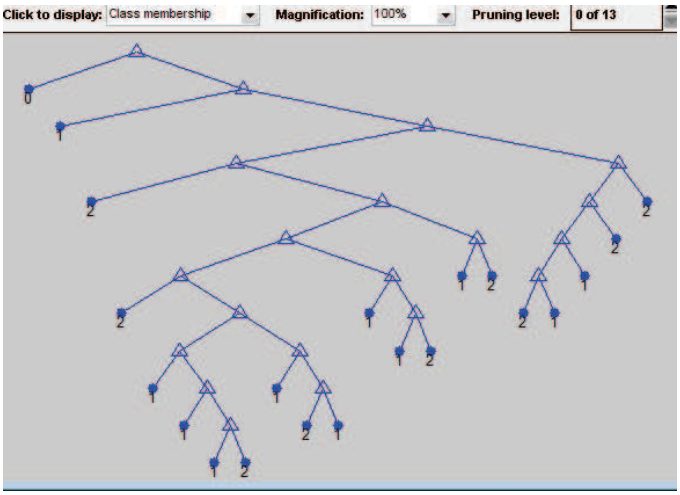


Figure 4.12: Classification Tree After Pruning and Cross Validation

4.5.7 Residual Evaluation with Classification Tree and Fisher Discriminant Analysis

Fisher Discriminant Analysis is implemented based on the algorithm described in Section 4.4. The original training data are transformed into FDA data which are used to build the classification tree. Testing data is also transformed in the same way to get FDA data for the classification tree to perform fault detection and diagnosis.

In this study, third order FDA, in which the order shows the dimension of residual space, is performed on the classification trees generated. Cross validation is applied to produce the optimal classification tree and the cost of the tree is plotted in Figure 4.13.

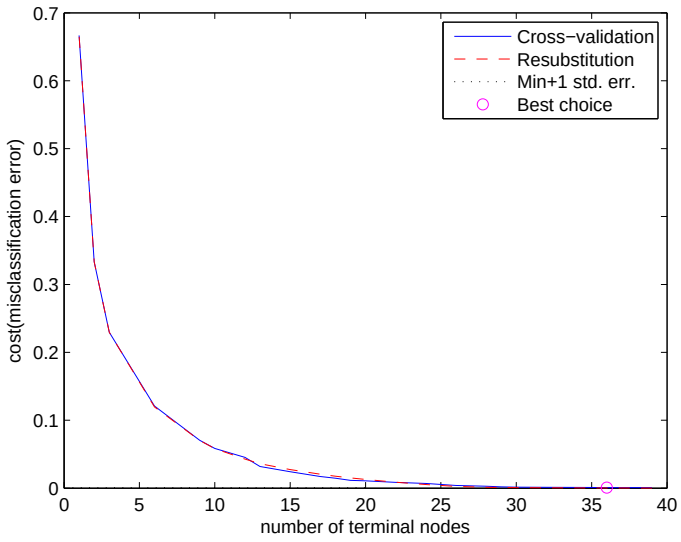


Figure 4.13: Cost of the Tree with FDA using Cross Validation

4.5.8 Results and Discussion

Comparing the performance of various approaches, both methods are trained with training data and tested against the testing data. Procedure is as follows:

1. The system is exercised for three cases (one healthy condition and two faulty conditions) which are denoted by f_i $(i = 1, 2, 3)$.

2. Three different residuals are obtained using the neural network model of the actual system which are denoted as $R_1^{f_i}$, $R_2^{f_i}$, $R_3^{f_i}$ $(i = 1, 2, 3)$.

3. First, classification procedure is applied on this residual set which is shown in Figure 4.14. Each fault is denoted by a number in the classification tree as "0" for healthy case, "1" for fault 1 case and "2" for fault 2 case. As it is seen in the figure, for the fault 1 case and fault 2 cases the tree shows a small

amount of data in wrong class but a great amount is classified correctly.

4. Second, the classification with FDA is applied on this residual set which is shown in Figure 4.15. Same notation in the previous step is used for the fault cases. In this tree overall misclassification error is less compared to the classification tree. Although there is still some misclassification between fault 1 and fault 2 cases.

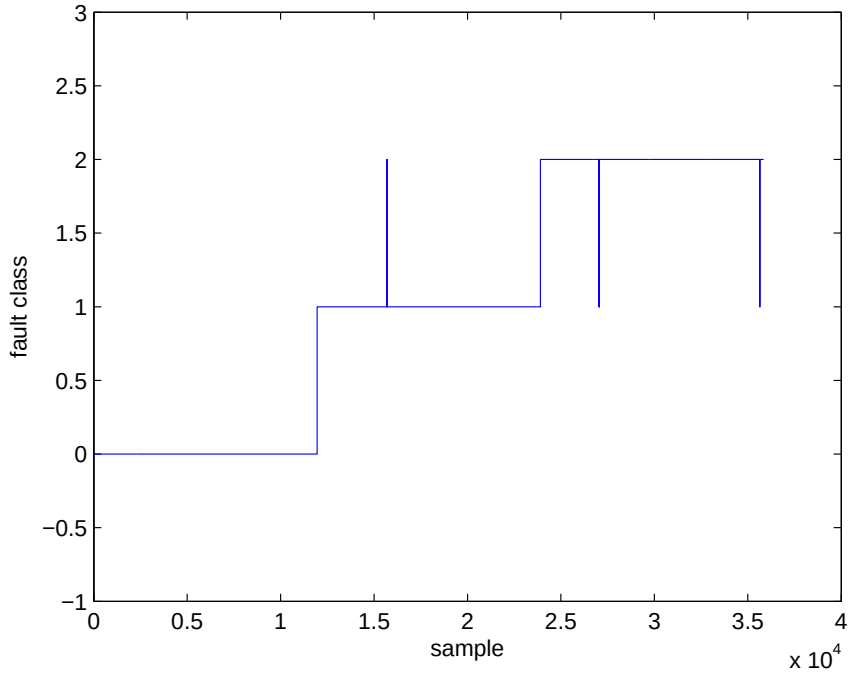


Figure 4.14: Classification Tree in Fault Classification

The misclassification rates on the testing data of classification tree and classification tree with FDA are compared in Table 5.

As it is seen from Table 5, the misclassification rate for the FDA is very close to that of the classification tree. However the elapsed time is increased for classification tree with FDA because of the relatively small size of the

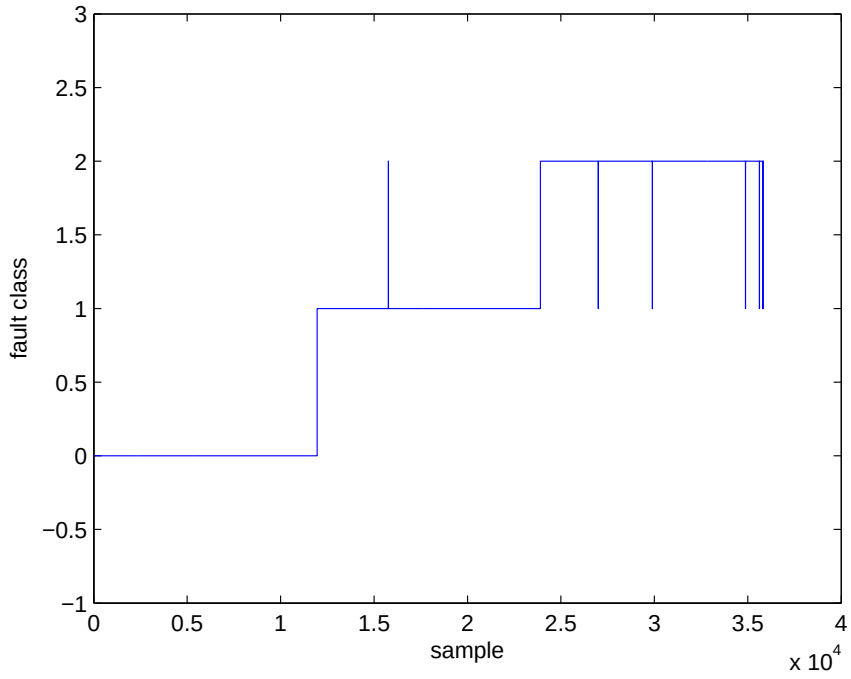


Figure 4.15: Classification Tree with FDA in Fault Classification

Table 5: Comparison of CT and CT with FDA

	CT	FDA with CT
Elapsed Time(seconds)	22.442	32.756
Misclassification Rate for f1	3.333e-4	1.667e-4
Misclassification Rate for f2	5.833e-4	6.667e-4

training data set. Another disadvantage is that classification tree-based methods are not effective when a big portion of the faulty states falls closely to normal states.

4.6 Case Study 2: Nonlinear Mass-Spring-Damper System with Coulomb Friction

An application example based on a nonlinear mass-spring-damper system in a closed loop control is given in this section as a second example to illustrate the fault diagnosis procedure using classification tree and classification tree with FDA approaches.

Modeling of the system is as follows:

1. Figure 4.16 represents the diagram of mass-spring-damper (MDS) system with mass m (in kg), spring constant k (in N/m) and viscous damper of damping coefficient c (in kg/s).

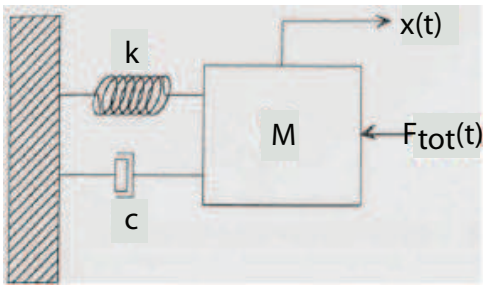


Figure 4.16: Schematic Diagram of Mass-Spring-Damper System

2. This system is subject to an oscillatory force:

$$F_s = -kx \tag{4.26}$$

and a damping force:

$$F_d = -cv = -c\frac{dx}{dt} = -c\dot{x} \tag{4.27}$$

where x represents the position vector.

3. In order to design a nonlinear effect in the system, Coulomb friction force is added on the damping force which is represented as an effect at zero velocity. In this study, Coulomb force is taken as an offset:

$$F_c = 0.5 \tag{4.28}$$

4. Applying Newton's Second Law on the free body mass, the total force F_{tot} becomes:

$$\begin{aligned} F_{tot} &= F_s + F_d + F_c \\ m\ddot{x} &= -kx - c\dot{x} - 0.5 \end{aligned} \tag{4.29}$$

5. Then the differential equation is:

$$\ddot{x} + \frac{k}{m}x + \frac{c}{m}\dot{x} + \frac{0.5}{m} = 0 \tag{4.30}$$

6. For a random step reference input force, PID controlled MSD system is given in Figure 4.17.

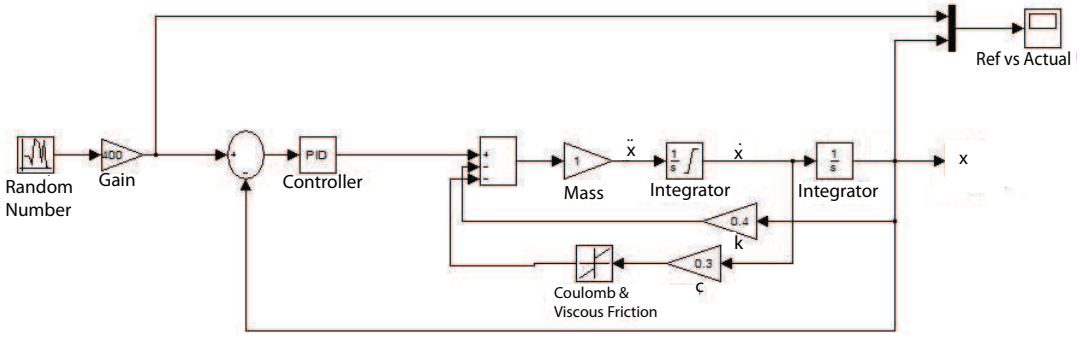


Figure 4.17: Matlab Model of Mass-Spring-Damper System

7. The system is modeled in Matlab/SIMULINK environment and simulated for 1500 seconds with fixed step ode4 type. Reference Point Position vs Actual Position acting on the system is given in Figure 4.18.

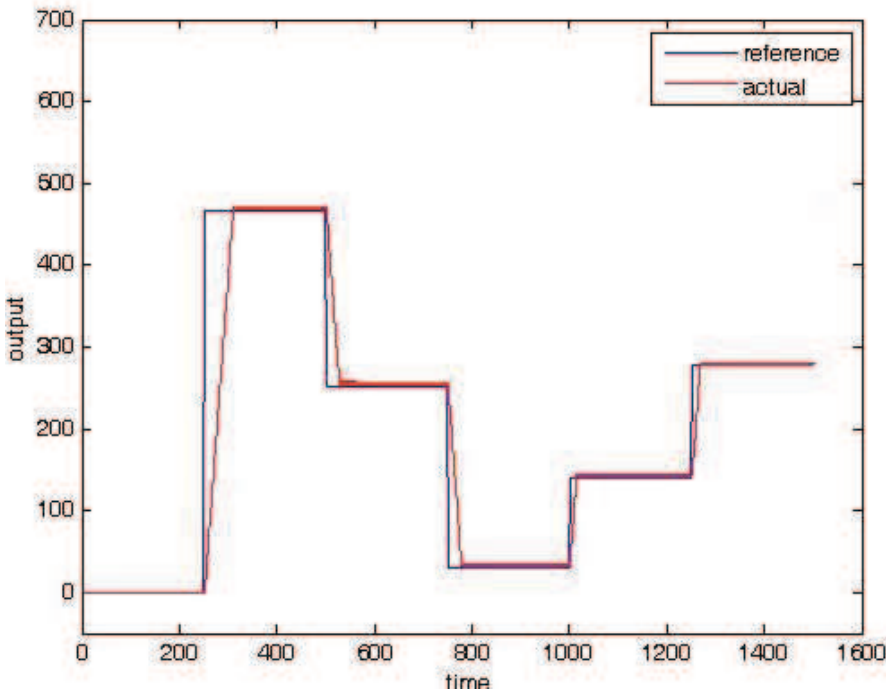


Figure 4.18: Reference Input vs Actual Output

4.6.1 Neural Network Process Model

In the model-based fault diagnosis of the MSD system, a Neural Network model of the actual output position, which is exercised under normal operation, is used. A two input one output feedforward neural network is implemented offline using Matlab Neural Network Toolbox. The inputs and output (for the discrete time value i) are:

Inputs= $F_{tot}(i)$, $x(i-1)$;

Output= $x(i)$;

Same procedure described in section 4.5.4 is used in the training stage. However in this case, 25 neurons (in the hidden layer) and 300 epochs are used in the training stage. Outputs of the neural network and actual system in normal condition is shown in Figure 4.19.

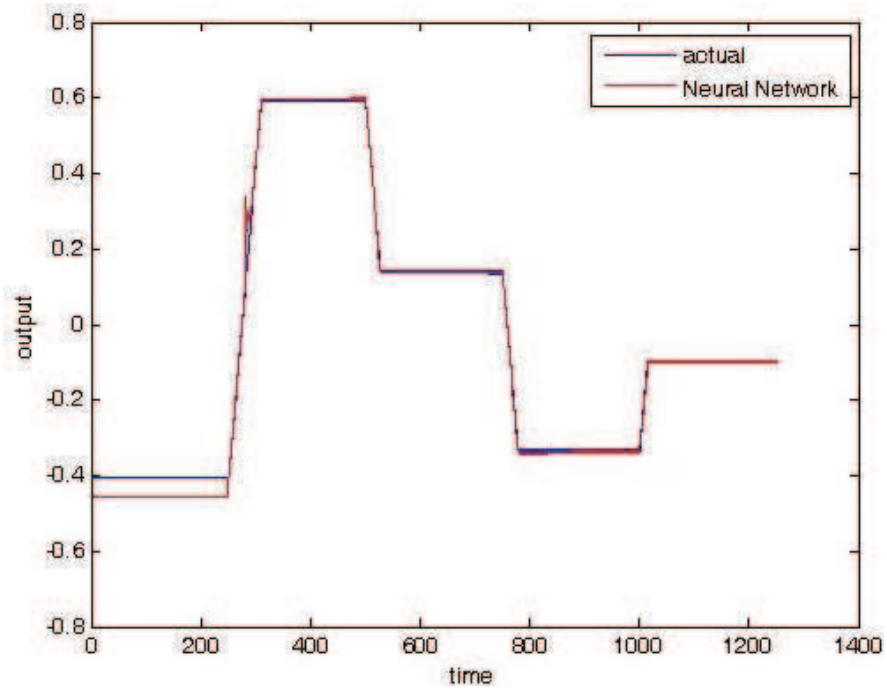


Figure 4.19: Actual Output vs NN Output

4.6.2 Residual Generation and Feature Extraction

Residual Generation

Remembering Section 4.5.5, three different residuals applied on the deviation between the actual system and neural network model of the system are used in fault diagnosis procedure. These are RMS Value, Standard Deviation and Power Value which are defined in (4.21), (4.25), (4.24).

Feature Extraction

MSD system is simulated for four different cases which are system under normal operation, fault 1 (f1) , fault 2 (f2) and fault 3 (f3) cases. f1 case is a decrease in k value, f2 case is a decrease in m value and f3 case is an increase in Coulomb friction. The system is simulated for each cases and residuals related to the fault classes are obtained. Three dimensional representation of the residuals is given in Figure 4.20.

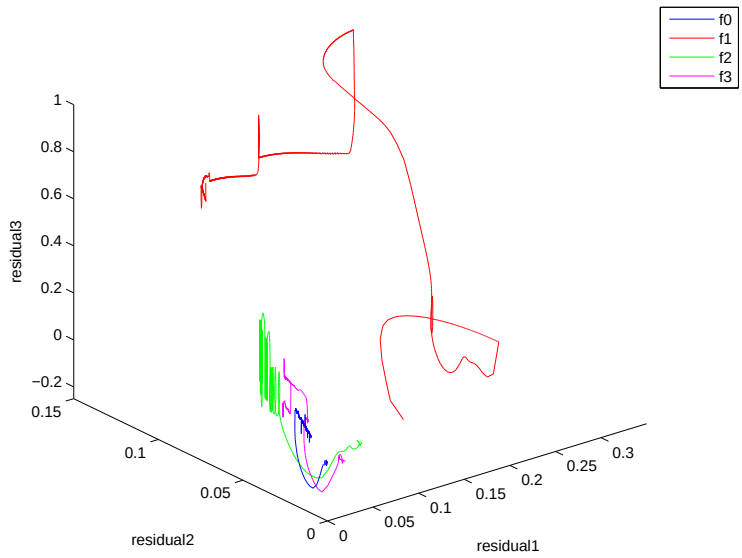


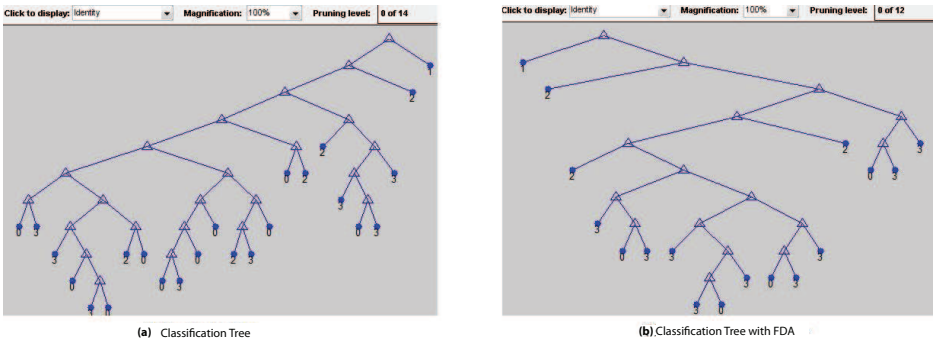
Figure 4.20: Residuals in Three Dimensional Space

4.6.3 Results and Discussion

Applying the same classification tree approaches used in lead-acid battery, a maximum tree is obtained with 24 leaf nodes (Figure 4.21(a)) whereas CT with FDA has 17 lead nodes (Figure 4.21(b)).

However in the second case study it is assumed that there is no noise or disturbance effect on the system. So the pruning stage does not change the

size of the tree. Pruned trees for Classification Tree and Classification Tree with FDA are shown in Figure 4.21.



(a) Classification Tree (b) Classification Tree with FDA

Figure 4.21: Comparison of the Pruned Trees

Classification Tree is shown in Figure 4.22a and Classification Tree with FDA is shown in Figure 4.22b.

In Table 6 the classification tree and CT with FDA are compared. FDA has longer response time and poorer performance compared to the CT.

Table 6: Comparison of CT and CT with FDA

	CT	FDA with CT
Elapsed Time(seconds)	2.789	5.722
Misclassification Rate for F1	0.004	0.005
Misclassification Rate for F2	0	0.008
Misclassification Rate for F3	0.008	0.024

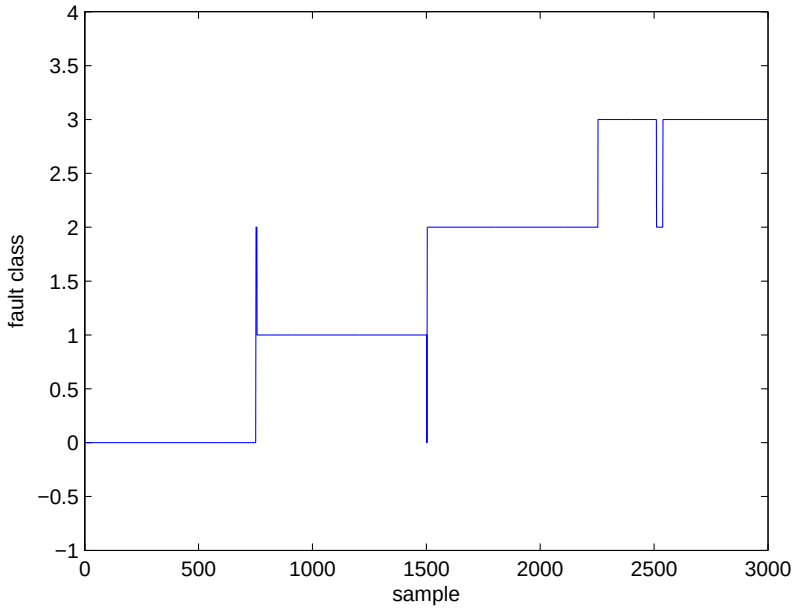
Comparing the performance of the tree in lead-acid battery and MSD systems, the MSD system has shorter response time and lower misclassification rate, since there is no noise effect on the system and the simulation time is smaller than the Lead-Acid Battery system.

4.7 Conclusion

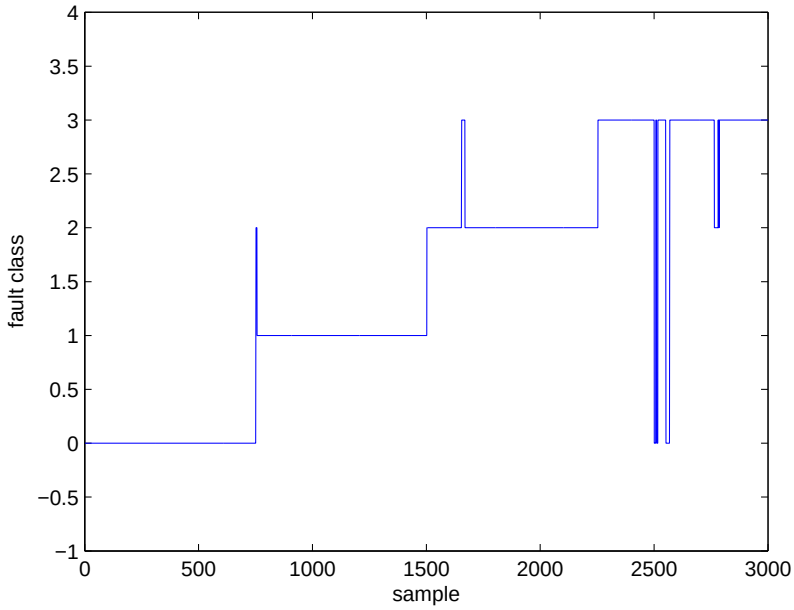
In this chapter a general information is given about classification tree and classification tree with FDA. The tree building and pruning methods using cross-validation technique are covered. Error rate is calculated for the obtained trees.

In the second subsection, rechargeable lead-acid battery principles are demonstrated. The important parameters for the fault diagnosis are represented. Based on these parameters, the neural network modeling of the actual battery is performed. Then the classification tree approach is applied to the residuals obtained from the neural network model produced. It is obvious that both of the approaches identifies the faultless case precisely. However in the faulty cases their performances are close to each other. On the other hand, normal classification tree can reach the optimum tree size in shorter time than the tree with FDA.

In the third subsection, nonlinear mass-spring-damper system is modeled. The algorithm used in lead-acid battery system is applied on this system. However in this case classification tree has smaller error rate compared to the classification tree with FDA. Also it has shorter computational time but larger tree size.



(a) Classification Tree



(b) Classification Tree with FDA

Figure 4.22: Classification Tree Application on MSD System

Chapter V

5 Conclusions

Fault Diagnosis and Detection in industrial processes are crucial in terms of the production cost and quality of the product. There are two important stages in fault diagnosis: residual generation and residual evaluation. Residual generation is based on the analytical model of the healthy system and residual evaluation step consists of the classification of faults occurred in the system using the generated residuals.

Therefore model based fault diagnosis methods have been very popular in last thirty years. These can be grouped as qualitative and quantitative model based diagnosis methods. Quantitative methods which are applied to only linear systems include parameter estimation, parity relation and observer based methods. On the other hand, qualitative methods which are applied generally to nonlinear systems consist of neural network (NN), fuzzy logic and neuro-fuzzy approaches.

Neural networks can be applied to fault diagnosis and detection as a process model as well as a fault classifier to differentiate different faults in the system. In this thesis three cases studies are investigated for fault diagnosis and in modeling of these cases neural network tool is used. In the first case study which is the water-tank system, neural network is also used as fault classifier. Its performance is compared with the adaptive neuro-fuzzy inference system. Although neural networks have longer training time, they

have better modeling capability.

In the second and third cases (lead-acid battery models and mass-spring damper systems, respectively), statistical methods such as classification tree and fisher discriminant analysis are applied on the residuals generated from NN process model.

The classification tree with FDA has longer training time but smaller tree size, since FDA extracts the most significant components in the original process data and achieves optimal discrimination among different faults as well as reduces the dimension of the original data. However misclassification rates of both approaches for different faults, which are highly affected from noise and disturbances, are close to each other.

A Matlab Codes

A.1 Data Acquisition and Normalization of Data for NN

load aku2pidli001; % no fault data (taken by four data blocks)

load aku2pidli002;

load aku2pidli003;

load aku2pidli004;

load aku2susuz001; % fault 1 data

load aku2derindeshr001; % fault 2 data

l1= aku2pidli001;

l2= aku2pidli002;

l3= aku2pidli003;

l4= aku2pidli004;

h1=aku2susuz001;

h2=aku2derindeshr001;

vo=[l1.Y(7).Data l2.Y(7).Data l3.Y(7).Data l4.Y(7).Data]; % actual voltage measured by DSpace

soc=[l1.Y(2).Data l2.Y(2).Data l3.Y(2).Data l4.Y(2).Data]; % actual SoC measured by DSpace

cu=[l1.Y(3).Data l2.Y(3).Data l3.Y(3).Data l4.Y(3).Data]; % actual current measured by DSpace

% Data Measured for Fault 1 Case

h1vo=[h1.Y(7).Data];

h1soc=[h1.Y(4).Data];

h1cu=[h1.Y(1).Data];

% Data Measured for Fault 2 Case

```
h2vo=[h2.Y(7).Data];
h2soc=[h2.Y(4).Data];
h2cu=[h2.Y(1).Data];
%Normalization of Data
cu1=(cu-mean(cu))/(max(cu)-min(cu));
vo1=(vo-mean(vo))/(max(vo)-min(vo));
soc1=(soc-mean(soc))/(max(soc)-min(soc));
cu2=(h1cu-mean(h1cu))/(max(h1cu)-min(h1cu));
vo2=(h1vo-mean(h1vo))/(max(h1vo)-min(h1vo));
soc2=(h1soc-mean(h1soc))/(max(h1soc)-min(h1soc));
cu3=(h2cu-mean(h2cu))/(max(h2cu)-min(h2cu));
vo3=(h2vo-mean(h2vo))/(max(h2vo)-min(h2vo));
soc3=(h2soc-mean(h2soc))/(max(h2soc)-min(h2soc));
```

A.2 Training of the Neural Network

```
% Inputs and outputs are taken from the Matlab simulation:
input1= simout2(:,1)'; % actual current
input2= simout2(:,2)'; % delayed Soc
input3= simout2(:,3)'; % delayed voltage
output1= simout2(:,4)'; % actual voltage
input=[input1;input2;input3];
net = newff(input,output1,30,'tansig' 'purelin','trainlm'); % feedforward
backprop.
net.trainParam.epochs =15000;
net.trainParam.lr=0.30; % (learning rate)
net.trainParam.maxfail=13250;% (maximum failure)
net.trainParam.mc=0.6; % (momentum)
```

```
net = train(net,input,output1);
gensim(net,1) % generation of NN block for simulink
```

A.3 Generation of Classification Tree

```
function m=cartmain(trainx,trainy,testx,testy)
    tic;
    T=optimaltreefit(trainx,trainy)
    YFITnum=treeevaluate(T,testx)
    m=misclass(YFITnum,str2num(testy));
    plot(m)
    toc
    function YFITnum=treeevaluate(T,testx)
    % Test the classification tree using test data
    % returns numerical values of classes
    YFIT=treeval(T,testx);
    YFITnum=double(YFIT)-ones(size(YFIT));
    function T=optimaltreefit(trainx,trainycart)
    % Inputs: trainx–numerical
    % trainycart– string
    %
    %Output: T–Classification Tree
    %
    % Get the original tree
    T=treefit(trainx,trainycart,'prune','on');
    % Calculate the Resubsitition Error rate
    resubcost=test(T,'resub');
    % Optimize the tree using cross-validation
```

```
[c,s,n,best]=treetest(T,'cross',trainx,trainycart);
% Prune the tree
T=treeprune(T,'level',best);
view(T)
% Plot the Best Tree
plot(n,c,'b-',n,resubcost,'r–')
figure(gcf);
xlabel('number of terminal nodes');
ylabel('cost(misclassification error)')
legend('Cross validation')
%best tree
[mincost,minloc]=min(c);
cutoff=mincost+s(minloc);
hold on
plot([0 20],[cutoff cutoff],'k:')
plot(n(best+1),c(best+1),'mo')
legend('Cross-validation','Resubstitution','Min+1 std. err.','Best choice')
hold off
c(best+1)
function m=misclass(actual,target)
sizeofset=size(actual);
boundaryofclasses=zeros(20,2);
temp=target;
currentclass=1;
boundaryofclasses(currentclass,1)=1;
counter=1;
while counter<sizeofset(1)
```

```
while target(counter)==0  counter<sizeofset(1)
counter=counter+1;
end
boundaryofclasses(currentclass,2)=counter-1;
currentclass=currentclass+1;
boundaryofclasses(currentclass,1)=counter;
target=target-ones(sizeofset);
end
boundaryofclasses(currentclass-1,1)=counter;
target=temp;
numberofclass=currentclass-1;
misclass=zeros(numberofclass+1,1);
temp=sign(abs(actual-target));
for i=1:numberofclass misclass(i,1)=sum(temp(boundaryofclasses(i,1):boundaryofclasses(i,2),1
boundaryofclasses(i,1)+1); end
misclass(numberofclass+1,1)=sum(temp)/sizeofset(1);
m=misclass;
```

A.4 Generation of Classification Tree with FDA

```
function m=fdacartmain(trainx,trainy,testx,testy)
tic;
Wp=fdam(trainx,trainy);
fdatrainx=trainx*Wp;
fdatestx=testx*Wp;
T=optimaltreefit(fdatrainx,trainy);
YFITnum=tree_evaluate(T, fdatestx);
m=misclass(YFITnum,str2num(testy));
```

```
plot(m)
toc
function m=fdam(trainx,trainy)
% Fisher Discriminant Analysis
% trainx: numerical, training data, predicting variables
% trainy: string, training data, classes
%
% function returns Wp –loading vector
rowsize=size(trainx)*[1;0];
columnsize=size(trainx)*[0;1];
% backup trainy value
temp=trainy;
% convert to numerical
trainy=str2num(trainy);
%max no of classes=20
boundaryofclasses=zeros(20,2);
currentclass=1;
boundaryofclasses(currentclass,1)=1;
counter=1;
% find class boundaries
while counter<rowsize
while trainy(counter)==0  counter<rowsize
counter=counter+1;
end
boundaryofclasses(currentclass,2)=counter-1;
currentclass=currentclass+1;
boundaryofclasses(currentclass,1)=counter;
```

```
trainy=trainy-ones(rowsize,1);
end
boundaryofclasses(currentclass-1,2)=counter;
numberofclass=currentclass-1;
% restore train_y
trainy=temp;
% total mean and covariance
avgT=mean(trainx);
sT=(rowsize-1)*cov(trainx);
% calculate class averages and covariances for max class=20
% sb– between class, sw–within class
s=zeros(columnsize,columnsize,20);
avg=zeros(20,columnsize);
sb=zeros(columnsize,columnsize);
sw=zeros(columnsize,columnsize);
for i=1:numberofclass
avg(i,:)=mean(trainx(boundaryofclasses(i,1):boundaryofclasses(i,2),:));
s(:,:,i)=(boundaryofclasses(i,2)-boundaryofclasses(i,1))*...
cov(trainx(boundaryofclasses(i,1):boundaryofclasses(i,2),:));
sw=sw+s(:,:,i); sb=sb+(boundaryofclasses(i,2)-boundaryofclasses(i,1)+1)*(avg(i,:)-
avgT)'*(avg(i,:)-avgT);
end
% test if sT=sb+sw;
% sT-sb-sw
[V,D]=eig(sb,sw);
diag(D)
order=input('enter the FDA order you chose:');
```

```
m=V(:,1:order);
```

A.5 Generation of Data for Classification Tree

```
%Data Acquisition for Fault Cases :
load hatasiz;
load hatabir;
load hataiki;
% Three Different Residual Generation
sayi1=[hatasiz(:,1);hatabir(:,1);hataiki(:,1)];
sayi2=[hatasiz(:,2);hatabir(:,2);hataiki(:,2)];
sayi3=[hatasiz(:,3);hatabir(:,3);hataiki(:,3)];
% Generation of Training Data
k=0;
for j=1:2:120
for i=1:1:60
array2(i+50*k,1:3)=[sayi1(50*j+i,1) sayi2(50*j+i,1) sayi3(50*j+i,1)];
array2=array2(:,1:3);
end
k=k+1;
end
% Generation of Test Data
k=0;
for j=0:2:119
for i=1:1:60
array1(i+50*k,1:3)=[sayi1(50*j+i,1) sayi2(50*j+i,1) sayi3(50*j+i,1)];
array1=array1(:,1:3);
end
```

```
k=k+1;
end
trainx=array1(1:3000,1:3);
testx=array2(1:3000,1:3);
trainy=[repmat('0',750,1);repmat('1',750,1);repmat('2',750,1);repmat('3',750,1)];
testy=trainy;
% Call the program for Classification:
fdacartmain(trainx,trainy,testx,testy)
cartmain(trainx,trainy,testx,testy)
```

References

[1] R. Beard, "Failure Accommodation in Linear System Through Self-Reorganization(PhD thesis)," *MIT, Massachusetts,USA*, 1971.

[2] H. Jones, "Failure Detection in Linear Systems(PhD thesis)," *MIT, Massachusetts,USA*, 1973.

[3] R. K. Mehra and P. J., "An innovations approach to fault detection and diagnosis in dynamic systems," *Automatica*, vol. 7, pp. 637–640, 1971.

[4] R. N. Clark, F. D. C., and W. W. M., "Detecting Instrument Malfunctions in Control Systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 11, pp. 465–473, 1975.

[5] L. A. Mironovsky, "Functional Diagnosis of Linear Dynamic Systems-A Survey," *Automation Remote Control*, vol. 41, pp. 1122–1143, 1980.

[6] R. Isermann, "Supervision, Fault Detection and Fault Diagnosis Methods-an Introduction," *Control Engineering Practice*, vol. 5, no. 5, pp. 639–652, 1997.

[7] J. Chen and R. J. Patton, "Robust Model Based Fault Diagnosis for Dynamic Systems," *Kluwer Academic Publishers*, 1999.

[8] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and its Applications to Modeling and Control," *IEEE Trans. System, Man and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[9] R. Patton, "Robustness in Fault Diagnosis for Dynamic Systems," *Annuan Reviews in Control*, vol. 21, pp. 103–123, 1997.

[10] J. Gertler, "Analytical Redundancy Methods in Fault Detection and Isolation," *IFAC/IMACS Safeprocess Conference*, pp. 9–21, 1991.

[11] R. Patton, P. M. Frank, and C. R. N., "Issues in Fault Diagnosis of Dynamic Systems," *Springer Verlag, London*, vol. 21, 2000.

[12] P. M. Frank and X. Ding, "Frequency Domain to Optimally Robust Residual Generation and Evaluation for Model-Based Fault Diagnosis," *Automatica*, vol. 30, no. 4, pp. 789–804, 1994.

[13] A. H. Jazwinski, "Stochastic Processes and Filtering Theory," *Mathematics in Science and Engineering*, vol. 64, 1970.

[14] I. E. Potter and M. C. Suman, "Thresholdless redundancy management with arrays of skewed instruments," *Integrity in Electronic Flight Control Systems*, vol. 15, pp. 1–25, 1977.

[15] J. Gertler and D. Singer, "A new structural framework for parity equation based failure detection and isolation," *Automatica*, vol. 26, p. 381388, 1990.

[16] R. Babuska, "Fuzzy Modeling for Control," *Kluwer*, 1998.

[17] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man Mach. Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[18] B. Kosko, "Fuzzy Systems as Universal Approximators," *Int. Conf. Fuzzy Syst.*, vol. 7, no. 1, pp. 1153–1162, 1992.

[19] X. Wang, "A Course in Fuzzy Systems and Control," *Prentice-Hall,Upper Saddle River, NJ*, 1997.

[20] S. R. Jang, S. T., and E. Mizutani, "Neuro-Fuzzy and Soft Computing," *Prentice-Hall,Upper Saddle River, NJ*, 1997.

[21] D. Nauck and R. Kruse, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets and Systems*, vol. 101, pp. 261–271, 1999.

[22] J. Gertler, "Fault detection and diagnosis in engineering systems," *Marcel Dekker,New York*, 1998.

[23] F. Uppal, R. Patton, and C. Lopez-Toribio, "Soft computing approaches to fault diagnosis for dynamic systems: A survey," *Proceeding of IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes , Budapest, Hungary*, vol. 1, pp. 303–315, 2000.

[24] P. Frank and B. Koppen-Seliger, "New developments using Al in fault diagnosis," *Engineering Applications of Artificial Intelligence*, vol. 10, no. 1, pp. 3–14, 1997.

[25] G. Zhang, "Neural networks for classification: A survey," *IEEE Transactions on Systems Man and Cybernetics: Part C - Applications and Reviews*, vol. 30, no. 4, pp. 451–462, 2000.

[26] V. Venkatasubramanian, R. Rengaswamy, and S. Kavuri, "A review of process fault detection and diagnosis Part II: Qualitative models and search strategies," *Computers and Chemical Engineering*, vol. 27, pp. 313–326, 2003.

[27] B. Upadhyaya, K. Zhao, and B. Lu, "Fault monitoring of nuclear power plant sensors and field devices," *Progress in Nuclear Energy*, vol. 3, pp. 337–342, 2003.

[28] R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Man and Cybernetics*, vol. 23, pp. 665–683, 1993.

[29] J. Hertz, A. Krogh, and R. G. Palmer, "Introduction to The Theory of Neural Computing," *Addison-Wesley Publishing Company*, 1991.

[30] P. Balle and D. Fuessel, "Introduction to The Theory of Neural Computing," *Engineering Applications of Artificial Intelligence*, vol. 13, pp. 695–704, 2000.

[31] J. N. Morgan and J. Sonquist, "Problems in the analysis of survey data, and a proposal," *J. Am. Stat. Assoc*, pp. 415–434, 1963.

[32] L. Breiman, J. Friedman, R. A. Olshen, and C. Stone, "Classification and Regression Trees," *Wadsworth,Belmontf*, 1984.

[33] R. Kumar, V. K. Jayaraman, and R. D. Kulkarni, "An SVM Classifier Incorporating Simultaneous Noise Reduction and Feature Selection: Illustrative Case Examples," *Pattern Recognition*, pp. 41–49, 2005.

[34] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[35] M. Sugeno and G. T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets and Systems*, vol. 28, pp. 15–33, 1988.

[36] J. S. R. Jang, C. T. Sun, and M. E., "Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligent," *Prentice Hall*, 1996.

[37] E. Russel, L. Chiang, and R. D. Braatz, "Data-Driven Techniques for Fault Detection and Diagnosis in Chemical Processes," *Springer-Verlag,London*, 2000.

[38] R. O. Dudo and P. E. Hart, "Pattern Classification and Scene Analysis," *New York*, 1973.

[39] HowStuffWorks, "What is the difference between a normal lead-acid car battery and a deep cycle battery?," Online, available at `http://auto.howstuffworks.com/question219.htm`.

[40] HyperPhysics, "Lead-acid Batteries," Online, available at `http://hyperphysics.phy-astr.gsu.edu/Hbase/electric/leadacid.html`.

[41] L. D. and T. B. Reddy, "Handbook of batteries," *New York:McGraw-Hill*, 2002.

[42] D. Marquardt, "An algorithm for least squares estimation of nonlinear parameters," *SIAM Journal of Applied Mathematics*, pp. 431–441, 1963.