NEURAL NETWORKS AND THE SMOOTH VARIABLE STRUCTURE FILTER

Training of Neural Networks Using the Smooth Variable Structure Filter with Application to Fault Detection

By

Ryan Ahmed

A Thesis Submitted to

College of Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of Masters of Applied Science (M.A.Sc.)

in the Department of Mechanical Engineering

McMaster University

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from McMaster University, I agree that the Libraries of this University may make it freely available for inspection. I further agree that the permission for copying this thesis in any manner, in whole or in part for scholarly purposes, may be granted by the professors who supervised my thesis work or, in their absence, by the Head of the Department or the Faculty Dean in which my thesis work was conducted. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and McMaster University in any scholarly use which may be made of any material in my thesis. Requests for permission to copy or to make other use of material in this thesis, in whole or part, should be addressed to:

Head of the Department of Mechanical Engineering

McMaster University

Faculty of Engineering

1280 Main Street West

Hamilton, Ontario L8S 4L6

Canada

Masters of Applied Science (2011) McMaster University

(Department of Mechanical Engineering) Hamilton, Ontario, Canada

TITLE: Training of Neural Networks Using the Smooth Variable Structure Filter with Application to Fault Detection

AUTHOR: Ryan Ahmed

SUPERVISOR: Professor S. Habibi

NUMBER OF PAGES: 196, xiii

Abstract

Artificial neural network (ANNs) is an information processing paradigm inspired by the human brain. ANNs have been used in numerous applications to provide complex nonlinear input-output mappings. They have the ability to adapt and learn from observed data.

The training of neural networks is an important area of research and consideration. Training techniques have to provide high accuracy, fast speed of convergence, and avoid premature convergence to local minima.

In this thesis, a novel training method is proposed. This method is based on the relatively new Smooth Variable Structure filter (SVSF) and is formulated for feedforward multilayer perceptron training. The SVSF is a state and parameter estimation that is based on the Sliding Mode Concept and works in a predictor-corrector fashion. The SVSF applies a discontinuous corrective term to estimate state and parameters. Its advantages include guaranteed stability, robustness, and fast speed of convergence.

The proposed training technique is applied to three real-world benchmark problems and to a fault detection application in a Ford diesel engine.

SVSF-based training technique shows an excellent generalization capability and a fast speed of convergence.

"To my parents, Laila and Mohamed Ali, who provided me with love and support and will be the inspiration to all of what I do in my life"

Acknowledgements

The author would like to express his gratitude to his supervisor, Dr. S.R. Habibi, for his supervision, advice, and guidance from the very early stage of this research until the writing phase of this thesis.

Financial support provided by the School of Graduate Studies, the Department of Mechanical Engineering and the Ontario Government's Graduate Scholarship Program is acknowledged.

I convey special acknowledgement to my fellow graduate student, Mohammed El-Sayed for his assistance, technical expertise, and support during this research.

I gratefully thank Andrew Gadsden, Kevin McCullough, Mohammed Al-Ani, and Haran Arasaratnam for their help.

I would also like to express my gratitude to my family, my mother Laila and my father Mohamed Ali, and Heba, Max and Michelle for their endless encouragement and support.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Preliminary Remarks

In the last three decades, a range of intelligent algorithms have been developed and applied to pattern classification problems. Amongst these, artificial neural networks (ANNs) have been prevalent as they are adaptive and show exceptional non-linear input-output mapping ability [1].

ANNs are information processing models inspired by the human brain. The human brain has over 100 billion neurons that communicate with each other and help us to see, think, and generate ideas and thoughts. Human brain learns by creating connections among these neurons using electrical and chemical signals. Inspired by the human brain, ANNs are a mathematical rendition of neurons that communicate with one another and learn from experience. Training of an ANN is achieved using training data sets that represents a specific input-output mapping. ANNs mimic the information processing model of the human brain. It is typically implemented in applications requiring adaptation and intelligence such as face and speech recognition, pattern classification, and fault detection.

ANN training is the process of using input-output training data sets to obtain the optimal network weights that represents connections among neurons. Several training algorithms have been proposed. The first order gradient descent back-propagation (BP) algorithm was introduced in the 80's. It however shows poor performance as it is slow, its input-output mapping is inaccurate, and training might easily converge to a local minimum. Accordingly, a number of alternative training techniques have been proposed to overcome the drawbacks of the BP algorithm. The second-order BP algorithms (e.g.: Quasi-Newton, Levenberg-Marquardt) are amongst these techniques. The second order BP training algorithm provides a fast speed of convergence and an accurate mapping. State estimation techniques such as the famous Kalman filter (KF) and the extended Kalman filter (EKF) have also been used to train ANNs by formulating the network as a filtering problem. These techniques can provide better generalization and avoid premature convergence in local minima.

Recently, a new filtering approach referred to as the Smooth Variable Structure Filter (SVSF) has been proposed. The SVSF is a state and parameter estimation strategy that has a predictor-corrector form. Its advantages include accuracy, robustness, guaranteed stability, and performance indicators that provide a measure of uncertainties in the filter model.

The aim of this research is to develop a novel ANNs learning algorithm based on the relatively new SVSF and to provide a comparative assessment of its performance.

1.2 Research Objectives

The main objective is to develop a new ANNs training methodology based on the SVSF. In particular, the training of the feed-forward multilayer perceptron networks (MLP) is considered. Research objectives can be stated as follows:

- Developing a novel SVSF-based MLP training algorithm for pattern classification problems in multi-streaming, global mode (GSVSF).
- Application of the proposed algorithm to three widely used, benchmark problems and comparison of its performance with other well-known algorithms such as the EKF, and the first and the second order BP algorithms.
- Application of the new training algorithm to detect and classify faults in a FORD Diesel engine. Two faults were considered, namely piston chirp and missing bearing faults.

The performance of the new SVSF-based training method is compared in all cases to other classical training techniques.

1.3 Thesis Organization

The dissertation is organized as follows. Chapter 2 involves a literature review of various neural networks training techniques including the traditional first order BP algorithm, second-order BP, Levenberg-Marquardt, Quasi-Newton, and the EKF. In addition, it also reports on the use of various engine fault detection and diagnostic techniques involving artificial intelligence methods.

Chapter 3 describes the common methods used for the training of feed-forward MLP. The first and the second order BP algorithms, the EKF-based MLP training in global (GEKF) multi-streaming mode are discussed.

Chapter 4 describes the SVSF estimation method and presents the new SVSF-based training method.

Chapter 5 considers the application of the proposed algorithm to three benchmark problems. Its performance is compared with the first and second order BP algorithms and the EKF.

Chapter 6 considers the application of the proposed algorithm to fault detection and diagnosis using real data from practical and industrial systems. The Chapter considers the application of the proposed training algorithm to the detection and

classification of fault conditions in a FORD diesel engine. The performance of the SVSF-based training method is then compared to the above mentioned classical methods.

Chapter 7 provides conclusions and recommendations for future research.

Chapter 2: Neural Networks Training Techniques for Pattern Classification with Application to Fault Detection Problems

Introduction

Artificial neural networks (ANNs) are mathematical models that process information and adapt in a fashion inspired by the human brain. ANNs are capable of modeling relationships between a set of inputs and outputs. ANNs are applied in numerous applications such as pattern classification [2], pattern recognition [3], function approximation, data processing, and robotics applications [4].

The objective of this research is to propose a new ANNs training technique based on the relatively new Smooth Variable Structure Filter (SVSF) for its application to fault detection and isolation. This chapter provides an introduction and literature review of three topics pertinent to this thesis, namely, ANNs training techniques, fault detection and isolation methodologies, and the SVSF estimation strategy.

This chapter is organized as follows: Section 1 provides an introduction to ANNs and their various training techniques. Section 2 presents an introduction to different

Fault detection and isolation (FDI) techniques, including model and signal–based methods, with a focus on Artificial intelligence. Section 3 presents an overview of various publications that involve the SVSF as a parameter and state estimation and discusses the motivation for using the SVSF for the training of ANNs.

2.1 ANNs and their Training Techniques

Over the past three decades, various supervised learning techniques have been proposed and applied for pattern classification problems. Amongst these, artificial neural networks (ANNs) have been prevalent in the field of pattern classification especially for fault detection and diagnosis applications. ANNs show enhanced generalization capability, adaptation competency, and potent non-linear input-output mapping [1]. In fact, a neural network with sufficient number of neurons can approximately model any continuous function with an acceptable degree of accuracy [3,5].

Accordingly, a list of real-life applications where ANNs are extensively applied are as follows [6]: speech recognition [7], automatic vehicle control [8], detection of heart abnormalities [9,10,11], finger prints recognition [12], detection of explosives [13], underwater targets allocation using sonar signals [14], handwritten character recognition [15], chemical engineering applications [16,17], experiment calibration

[18,19], business applications such as bank failure prognosis and stock market predictions [20,21], metallurgical applications [22].

ANNs are sometimes compared to statistical methodologies for regression, prediction, and pattern classification [6]. Some researchers classify ANNs as a class of non-linear regression [23,3]. Lapedes and Farber show that ANNs' performance can be better than conventional statistical techniques in forecasting of two noise-free time series [24,23]. [25] and [26] report that ANNs provide enhanced performance compared to Box-Jenkins for time series with short memory, while for long memory time series, the Box-Jenkins shows better results. In addition, ANNs have been compared to other conventional statistical approaches for pattern classification and show enhanced performance over conventional discriminant analysis methods in [27,28].

One of the main drawbacks of ANNs compared to statistical analysis methods is their black-box structure. After training an ANN, even though the trained network can operate properly, the internal structure of the network often has no physical significance and is very difficult to understand. However, some advanced techniques have been proposed to extract knowledge embedded in the network [29]. These methods provide an understandable explanation of the knowledge learned by the network during the training phase.

ANNs can learn (or be trained) from experience (or by example) using training algorithms that can establish relationships between input-output datasets. During the learning (training) phase, ANNs can adaptively change their internal structure. Several ANNs training algorithms have been proposed. These consider accuracy, speed, computational complexity, and memory requirements.

In the next section, a brief introduction to ANNs and the neuron model is provided. It is followed by a literature review of training techniques.

2.1.1 Human and ANNs

ANNs are computational models that imitate the structure and function of biological neural networks found in human's central nervous system. They consist of numerous artificial neurons connected to each other to process information. In humans, the structure of the biological neuron is as shown in Figure 2.1. The neuron collects signals from input channels named dendrites, processes information in its nucleus, and then generates an output in a long thin branch called the axon.

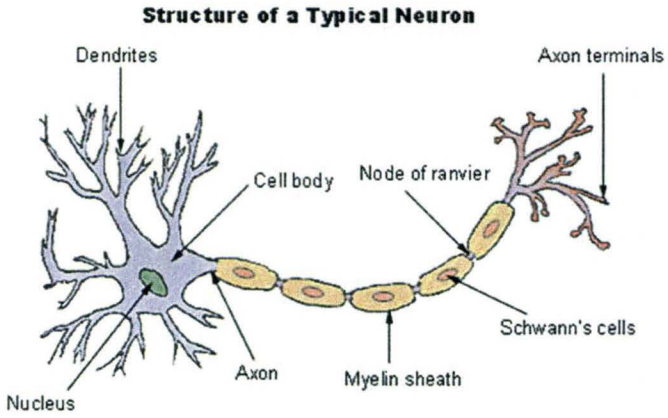

Figure 2.1. Biological Neuron Structure [30]

At the end of the axon, a synapse is used to conduct electrical and chemical signals to the dendrites of another neuron. Neurons can thus communicate with one another as shown in Figure 2.2. Human learning occurs adaptively by varying the bond strength between these neurons.

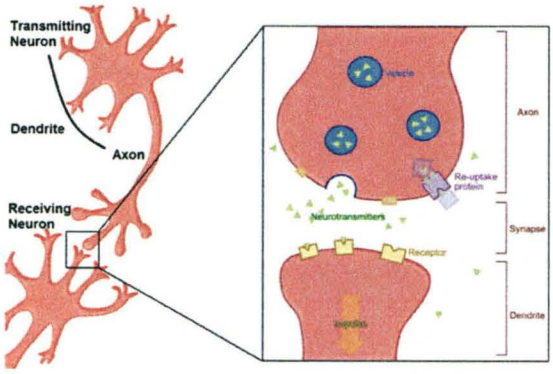

Figure 2.2. Two Biological Neurons Communicating with Each Other

2.1.2 Artificial Neuron Model

The neuron model as shown in Figure 2.3 is inspired by the human nervous system and represents a computational unit with multiple inputs and one output. The mathematical model of an artificial neuron with three inputs ($k = 3$) is as follows:

$$n = \sum_{i=1}^{k=3} P_i W_i = P_1 W_1 + P_2 W_2 + P_3 W_3 \tag{2.1}$$

$$a = f(n) \tag{2.2}$$

Where, W_1, W_2, W_3 are the network's synaptic weights that resemble the bonding strength among neurons. P_1, P_2, P_3 represent the neuron's inputs and a denotes the neuron's output. A weighted sum of the network's inputs is calculated generating a signal n followed by the application of an activation function $f(.)$ to generate the output a. The activation function can be linear or non-linear depending on the application.

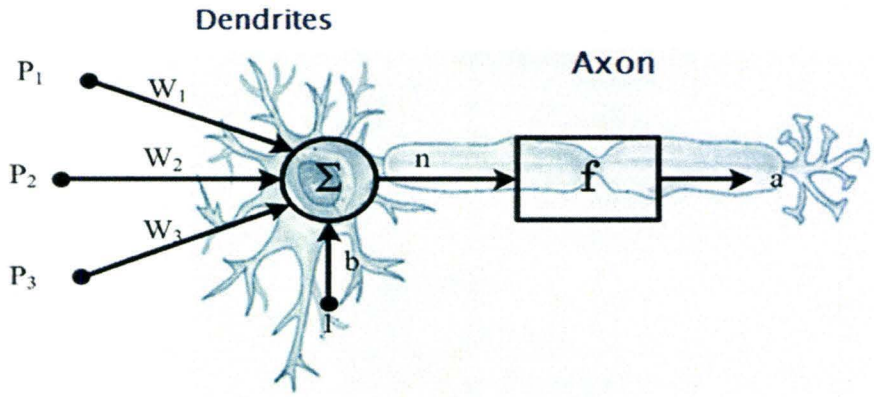

Figure 2.3. Transition from Human Neuron to Artificial Neuron Model

The history of artificial neural networks starts in 1943. The first artificial neuron model was developed by McCullah and Pits [31], the neuron has the same structure as above but the computational unit consists of a linear threshold function. The output is either 0 or 1 depending on if the neuron cell fires or not [32]. The neuron output is 1 if the sum of all the neuron's weighted inputs is more than a fixed threshold and 0 otherwise.

Artificial neural network can be made up by using a number of neurons interconnected together through synaptic weights. The first artificial neural network was developed by Rosenblatt in 1959 and made up of artificial neurons as proposed by McCullah and Pitts [33,31]. The only difference between Rosenblatt's neurons and that of McCullah and Pitts is the neuron's outputs that are scaled from -1 to 1 instead of 1 to

0. Rosenblatt's model is widely known as the "perceptron". ANNs structures were further developed in the 1960s by Rosenblatt [34] and by Widrow et al. [35].

Neural networks research diminished during the 1970s after the publication of a study titled "the Rosenblatt's untimely death" in 1969. The study shows that Rosenblatt's network fails to classify linearly inseparable functions such as the XOR [36].

The introduction of hidden layers in the perceptron networks by Schalkoff in the 1980's has overcome this problem. Adding hidden layers to the network can significantly enhance the network's computational capability [37]. The discovery has led to what is known as the feedforward multilayer perceptron network which is the core of this research project.

2.1.3 ANNs Training Techniques Literature Review

ANNs learning process operates by adaptively changing the network weights by using an error signal as shown in Figure 2.4 . Assuming a training data set $\{x(n), y(n)\}$, where $x(n)$ is the network input and $y(n)$ is the desired output. ANN training is performed by continuously adapting the weights according to the error signal The error signal $E(n)$, as shown in Figure 2.4, represents the difference between the network's actual output $z(n)$ and the target (or desired) output $y(n)$ from the training data set. Various algorithms have been presented to train ANNs, they differ in the way they propagate the error back to update the weights.

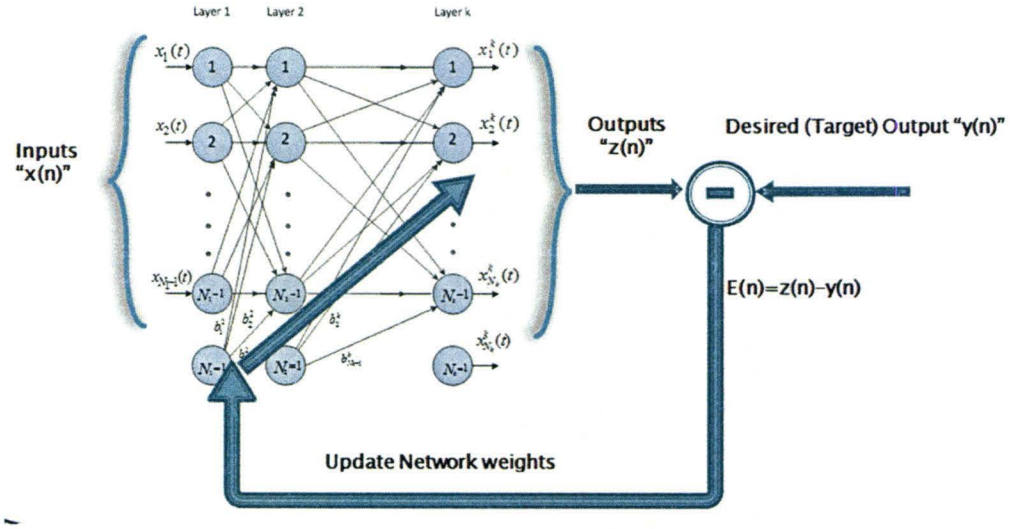

Figure 2.4. ANNs Learning Process

In the literature, various ANNs training techniques have been implemented. In general, ANNs training techniques are classified to four main categories as follows:

(1) First-order gradient descent Back propagation (BP) and its further enhancements (e.g. back propagation with momentum and batch-based back propagation).

(2) Optimization based techniques (e.g. Quasi-Newton, Newton's method, and Levenberg Marquardt).

(3) Stochastic-based methods (e.g. genetic algorithms, Simulated annealing (SA), and Alopex methods)

(4) Estimation-based methods (e.g. using Kalman filter, Extended Kalman filter, Unscented Kalman filter, Particle filter, and the Smooth Variable Structure Filter).

Since 1980s, several ANNs training techniques have been proposed. ***Back propagation (BP)*** is one of the first used in training of multilayer perceptrons [38]. It was reported by Rumelhart, Hinton, and Williams in 1986 [39]. BP is a first-order stochastic gradient descent method that iteratively searches for link weights that minimize the output error in a supervised manner. However, since early versions of BP involve a constant learning rate, a slow speed of convergence is attained. In fact, several enhanced training algorithms have been developed to improve training performance, mapping accuracy, and speed of convergence compared to the BP algorithm [40]. For instance, the nonlinear least squares ***Gauss–Newton method*** is a good candidate to iteratively solve supervised neural-network training problems [39]. Nevertheless, it has been shown in [41] that the Jacobian Matrix may become rank deficient in some cases, thus resulting in the numerical instability of the Gauss-Newton algorithm [42]. The second-order ***Levenberg-Marquardt*** training algorithm [43] has shown to circumvent the previous problems. Watrous [44] verified the application of a ***Quasi-Newton*** method to neural network training. Quasi-Newton method demonstrated better convergence performance than the standard BP algorithm but it requires large memory storage to store the Hessian Matrix [45].

The Quasi-Newton and Levenburg-Marquardt demonstrate better performance than BP as they involve second-order derivative information. In addition, these algorithms are implemented in a batch (multi-streaming) mode where weights are updated based on more than one training sample in the training set. This is in contrast to the conventional early versions of BP where weights are updated by involving only one training sample (a serial mode) [46]. Even though second-order algorithms have proven to outperform the classical first-order BP, they may suffer from poor convergence properties due to problems with local minima [40].

The Kalman filter (KF) is the most popular state estimation tool. It provides a statistically optimal estimate for linear systems in the presence of Gaussian white noise. In the case of nonlinear systems, the extended Kalman filter is applied by linearizing the system or measurement matrices around the current state estimate at each time. An ***EKF-based neural network training*** technique was first introduced by Singhal and Wu in 1989 [47]. The EKF provides a powerful neural network training capability compared to conventional first-order gradient-based algorithms, such as the BP [40]. In literature, the EKF has been widely applied for training of both feed-forward [48] and recurrent networks [49,50] in both a global form (GEKF) or in a decoupled form (DEKF). Although the EKF demonstrates a close performance compared to a second-order derivative, batch-based method, it can avoid local minima problems by encoding second-order

information in terms of a state error covariance matrix [40]. Accordingly, the EKF represents an efficient and practical alternative to second-order training methods.

Various enhanced ANNs training techniques have been proposed in several studies. A new hybrid learning algorithm that combines the EKF and particle filter has been presented in [51]. The new training scheme provides faster speed of convergence than the stand-alone EKF. An advanced EKF training technique has been proposed in [52]. The advanced form of Kalman filter-based parameter estimation method obtains a more accurate estimate of how a Gaussian distribution evolves under a nonlinear transformation. It has proven to offer performance advantages over standard EKF training. Reference [46] provides suggestions on how to initialize the EKF parameters in addition to presenting a new decoupling strategy that reduces the update rate of the error covariance matrix. Wan et al. [53] stated the effective use of the ***Unscented Kalman Filter (UKF)*** of Julier et al. [54] for feed-forward neural networks training.

A novel, fast ANNs training algorithm that works by decreasing the number of synaptic weights in a third-order ANNs is presented in (Zhang 2004) [55]. The algorithm uses a trigonometric approach for training of feedforward ANNs on a pattern recognition application. The new algorithm is able to reduce the complexity of the network while preserving its high classification precision.

Another training technique based on genetic algorithms is presented by Kawata et al. [56]. The new algorithm can overcome problems associated with gradient descent

BP algorithms in particular the slow convergence speed. The algorithm can simultaneously searches several subspaces to improve computational performance. A feedforward ANNs training algorithm has been presented by Looney in [57]. The algorithm provides a faster speed of convergence compared to the BP algorithm by pruning out unimportant synaptic weights and by iteratively adjusting a momentum term during optimization at every epoch.

Another training technique that has been proposed by Dubrovin el al. can evaluate network weights in a non-iterative fashion [58]. It works by applying cluster regression approximations to form a network topology. The algorithm is tested on a practical problem and results show that it provides a faster speed of convergence compared to classical BP training techniques.

Yang el al. proposed a new ANNs training methodology using the widely known Taguchi Method also known as the orthogonal array method for optimum ANNs design [59]. Taguchi method can be used for determining the most suitable network structure, including the number of hidden layers and the number of neurons in each layer. This method results in faster speed of convergence by dynamically adapting the gradient descent BP methods and aid in minimizing lengthy trial and error process to select network parameters.

It is known that as the size of the network is increased, the number of training samples must also increase to achieve good results. Accordingly, for large networks,

training might become a time-consuming process. Chang et al proposed an updated version of the Taguchi-based training methodology to solve the aforementioned problem by using orthogonal arrays to perform sample selection [60]. Applying orthogonal arrays can significantly decrease the required training samples while maintaining the network's accuracy.

2.2 Fault Detection and Isolation

Fault detection and Isolation (FDI) techniques are used for detecting fault conditions, isolating them, and generating an alarm signal whenever a malfunction occurs in the monitored system. Therefore FDI plays an important role in modern engineering systems due to increasing demand for safety and reliability, especially in the automotive and the aerospace sectors.

In the literature, while different classical FDI techniques have been implemented, Artificial intelligence based methods such as Neural Networks and Fuzzy logic have been prevalent. These methods have proven to increase the system's reliability and decrease the probability of producing false alarms.

In this section, a general overview of fault detection techniques is presented followed by classification of FDI techniques as discussed in the literature. These techniques are classified into three main groups, namely model-based FDI, signal-based FDI, and intelligent techniques. ANNs can be used in all three groups.

2.2.1 General overview of fault detection techniques

A fault is an unpredictable change in a system's behaviour that deteriorates its performance. Two types of faults are considered: ***Intermittent*** and ***permanent faults.*** Intermittent faults occur at irregular intervals and last for limited periods of time while the latter exist permanently from their inception until the faulty component or system is replaced or repaired.

Fault diagnosis algorithms start by ***fault detection*** which is to detect the presence of a fault within the system under diagnosis. It is followed by the ***fault isolation*** process that is used to determine the location of the fault. The next step is ***fault identification*** that involves estimating the severity of the fault [61].

2.2.2 Fault Detection Systems (FDS)

FDS can be applied using ***analytical*** [62,63] or **hardware redundancy** [64]. In analytical (or sometimes called functional or model based) techniques, fault detection is achieved by establishing an intrinsic relationship between measured variables and a mathematical model of the system under diagnosis [5]. State estimation methods are then used to track changes in the system behaviour with respect to a baseline model.

Hardware redundancy is implemented by incorporating additional hardware into the system to detect faults. Hardware redundancy is divided into two categories namely, ***static*** or ***dynamic*** hardware redundancy. In the former, more than one sensor (or

actuator) are used to measure (or control) the system. Faults can be detected by acquiring different measurements followed by performing a consistency check amongst signals. For the later, standby modules are subsequently activated in case of a failure condition to provide a fail-safe system.

Hardware redundancy has several drawbacks and increases system overhead. Duplicating components in the system increases weight, equipment and maintenance costs, and the space requirement. Consequently, hardware redundancy is limited to safety critical applications (e.g.: aerospace and nuclear reactors) [61]. Reference [65] provides recommendations for how to select hardware or analytical redundancies for different applications.

2.2.3 Classification of Fault Detection Techniques

Fault detection techniques have been divided into three main categories: Signal-based fault detection, Model-Based fault detection, and artificial intelligent techniques. In the following sections, an overview of these three methodologies is presented.

2.2.3.1 Signal-based Fault Detection

Signal based fault detection involves extracting the fault signature by comparing system measurements against their nominal operational trends. Analysis is performed in the time domain or in the frequency domain for extracting features or trends that can be attributed to fault conditions [61].

In the literature, various signal-based FDI techniques especially for the internal combustion engine fault detection have been discussed. Most of these FDI techniques use either noise levels as well as pressure, or vibration signals to detect faults.

In 1979, Chung et al. implemented an engine fault detection methodology using sound measurements by acquiring sound intensities using microphones. This technique is one of the oldest practices that were implemented at General Motors Research Laboratories. The method can effectively generate a thorough mapping of an engine's noise using cross-spectral analysis. This method is able to identify the noise source by using a noise source ranking methodology. In 2002, Leitzinger provided a comparison between laser Doppler vibro-meters, microphones, and accelerometers to detect engine faults. The research concludes that microphones provide an easy, non-contact measuring system but they might generate inconsistent results and produce false alarms. In addition, the research shows that accelerometers and laser Doppler vibro-meters provide more reliable measurements. Acoustic tests on internal combustion engines in a production environment using two overhead microphones to measure sound pressure are described in [66]

In literature, the ***Qualitative trend analysis (QTA)*** is one of the most widely used feature extraction techniques. QTA is a data-driven FDI methodology that works by

extracting features (trends) from the measured signals and accordingly takes decisions. QTA has been extensively applied for process fault detection and diagnosis [67].

Alternatively, Feature extraction can be performed using ***Discrete Wavelet-based techniques*** [68]. Fault detection using discrete wavelet transforms (DWT) is broadly deliberated in [69]. DWT techniques involve two main steps: First, measured signal decomposition followed by signal edge detection that occurs due to faults.

Artificial Neural Networks (ANNs)-based feature extraction methods have been broadly discussed through the literature. Since ANNs are the core of this research, Signal-based fault detection using ANNs will be explicitly discussed later in section 2.2.4.1 (Artificial intelligent methods).

After feature extraction using one of the previously mentioned methodologies, a trend interpretation algorithm is applied to arrive at meaningful conclusions pertaining to the fault conditions. The ***Hidden Markov Model*** is one of the most common trend interpretation algorithms. It is applied for matching extracted feature signal trends with those of a nominal one [70].

2.2.3.2 Model-based fault detection

Model-based fault detection is mainly based on residual generation. Residuals represent inconsistencies between the actual physical system measurements and the system's mathematical model output. Figure 2.5 shows a block diagram of the most common

form of model-based fault detection. It consists of a plant under consideration, a baselined mathematical model block that represents the healthy system with no faults, a residual generator, and a decision making block that performs residual evaluation.

The fault detection algorithm involves applying the same input to both the actual physical system as well as the baselined nominal model. Their outputs are compared using the residual generation algorithm. Ideally, the residual block output should be zero while a non-zero value indicates a fault condition. The decision block interprets the residual value by comparison against a threshold to diagnose the fault condition. It then produces an alarm signal if a fault condition is detected. The Residual evaluation stage (or decision block) is critically important as it presents a trade-off between the tendency to produce a false alarm and the potential failure of flagging an actual fault condition.

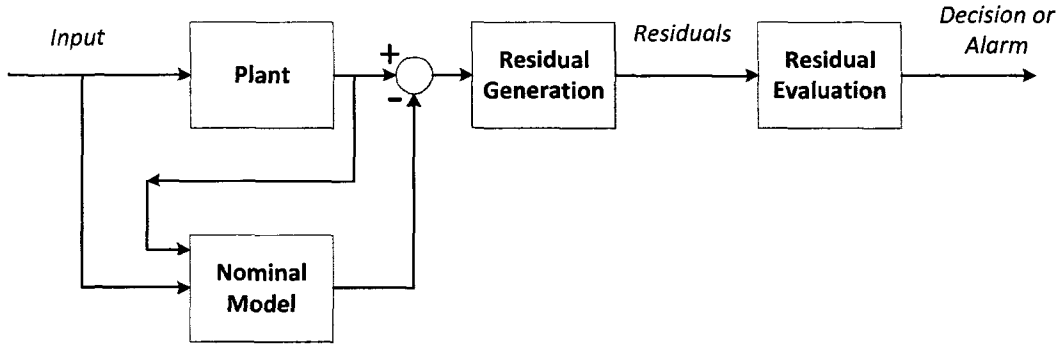

Figure 2.5. General Model-based FDI Block Diagram [71]

A list of advantages and drawbacks of model-based FDI is presented in [72,73] as follows:

- **Advantages**

  (1) The FDI methodology is able to detect the systems' transient faults.

  (2) Model-based FDI provides an excellent fault detection capability if an accurate system model is provided. They provide a minimum probability of generating false alarms and minimum probability of missing a fault condition.

  (3) Models can be generated based on physical principals.

- **Disadvantages**

  (1) Model-based FDI is not applicable for complex dynamic systems where attaining an accurate system model is complicated task.

  (2) Modeling uncertainties might lead to generation of false alarms.

  (3) Large systems have high computational requirements.

Model-based fault detection and isolation approaches can be divided into two main categories depending on the fault diagnosis data being measured as follows:

Discrete event model-based approaches

These methods are applied when the system under diagnosis can be described as a discrete-event system (DES). In DESs, the operation of a system is characterised by a chronological sequence of events. Each event takes place at a specific period of time and

performs a change in the system state. Finite state machines (FSM) and Petri nets (PN) are used to model DESs. PNs are widely used as a modeling tool for DESs compared to FSM. PNs provide clearer graphical descriptions, process synchronization, and simple mathematical formulation. DES Fault detection task is performed by comparing the discrete-events model output against the observed events from the system under diagnosis [74,75].

Differential or difference equations model-based approaches

These FDI approaches are applicable when the system's state variables can be numerically obtained or simulated. Differential or difference equations can be derived to obtain a physical model of the system.

This research is concerned with the second category of differential model-based systems since it deals with continuous time systems.

In general, three model-based FDI can generate residuals by using observers, parameter estimation, and parity space comparison as follows.

- **Observer and Filter-based FDI**

In the literature, observer-based residual generation techniques have been broadly applied in several FDI applications. Observers or filters are systems used to estimate the system's states and its output. Observer-based FDI techniques provide the ability of

designing a residual generator that is robust to model uncertainties [76]. Examples of observer-based FDI applications can be found in several studies [77,78,79].

Observer-based FDI can be classified into ***deterministic*** or ***stochastic*** settings approaches depending on how the system outputs are estimated from measurements. In deterministic setting approaches, the system outputs are estimated from the measurements by using linear or non-linear observers, high gain non-linear observers [80], and sliding mode observers [81]. In Stochastic setting, different estimation techniques have been applied to FDI problems including the Kalman filter [82], extended Kalman filter (EKF) [83], Unscented Kalman filter (UKF) [84], and the relatively new smooth variable structure filter (SVSF) [85].

A model-based engine fault detection using cylinder pressure estimates, combustion heat release, and torque estimates from non-linear observers is implemented by Kao and Moskwa [86,87]. Results from the proposed methodology have shown good performance, with fast convergence, and stability.

A neural network-based adaptive observer for aircraft engine parameter estimation is provided in [88]. This adaptive observer combines the Kalman filter with Neural Networks and is able to compensate for non-linearities that cannot be handled by the filter.

An adaptive observer based fault detection scheme is presented in [89]. With the aid of nonlinear adaptive observer theory, an approach of constructing adaptive residual generators is developed. Accordingly, this approach enhances robustness of residual generators with respect to model uncertainties. This FDI methodology has been applied to detect and classify Actuator faults based on estimations, results show good accuracy.

Patton and Chen introduced a new approach to the design of optimal observer-based residual generators for detecting incipient faults in flight control. The new approach reduced the probability of generating false alarms [90].

An observer-based fault detection system in robots using nonlinear and fuzzy logic residual evaluation is discussed in [91]. This FDI methodology makes use of non-measurable process information as an alternative of setting up numerous sensors. A fuzzy-based approach is used for threshold selection with the objective of improving robustness in fault detection thus minimizing the probability of producing false alarms. The suggested approach has been successfully implemented and experimental results show the benefits of the adaptive threshold.

A fault diagnostic scheme for aircraft engine sensor fault is presented in [92]. The proposed methodology can distinguish between modeling uncertainties and occurrence of faults in order to reduce false alarms. It encompasses a dynamic threshold algorithm that detects parametric uncertainties using a bank of observers. A robust fault detection observer is proposed for a class of nonlinear third order hydraulic systems [93].

Observer-based fault detection for a drive-train of a Jaguar car involving an automatic transmission is presented in [94]. A model representing the drive-train of the vehicle is derived using nonlinear polynomials that relate manifold pressure, engine speed, and the wheel speed. A non-linear full-order observer is designed and used for generating residuals by comparing the model output to the systems output. The proposed FDI methodology can detect three fault scenarios with high accuracy.

- **FDI using parameter estimation**

In this approach, faults are detected by estimating the system parameters online using parameter estimation techniques then comparing them with the parameters of a known healthy system. Consider a monitored system in the form:

$$y(t) = f(u(t), e(t), \theta, x(t)) \tag{2.3}$$

Where, $u(t)$ represents the system input vector, $y(t)$ is the output vector, θ represents systems parameters affected by fault occurrence and they cannot be measured, $e(t)$ denotes modeling errors and external disturbances (noise), and $x(t)$ denotes systems states.

The parameter-based FDI algorithm, as shown in Figure 2.6, is summarized by the following process [95]:

(1) Development of a baseline model for the healthy fault-free system model,

$$y(t) = f(u(t), \theta) \tag{2.4}$$

(2) Establishing a physical meaning for model parameters θ_i by linking them to physical parameters p_j.

(3) Model parameters θ_i estimation using inputs-outputs datasets.

$$\hat{\theta}(t) = g(y(1), \dots, y(t), u(1), \dots, u(t)) \tag{2.5}$$

(4) Physical parameters estimation $\hat{p}(t)$ from the model parameters $\hat{\theta}(t)$.

(5) Comparing the baseline model parameters $p_i(t)$ to the estimated parameters $\hat{p}(t)$ followed by generation of residuals $r(t)$ where

$$r(t) = \hat{p}(t) - p(t) \tag{2.6}$$

For fault-free systems, the residuals should be almost zero if the model is accurate and good estimates are provided by the parameter estimation technique [96].

(6) Fault conditions are detected by comparing parameters values to predetermined thresholds or by using advanced statistical methodologies.

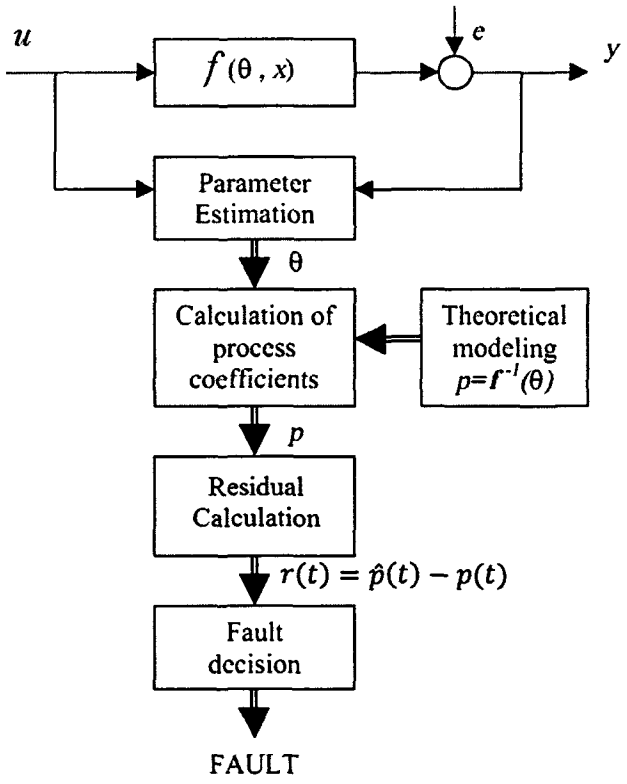

Figure 2.6. Parameter Estimation-based Fault Detection [95]

Parameter estimation FDI can be applied to both linear and non-linear systems only if estimated parameters have a physical significance. Reference [97] provides a brief summary of fault-detection methods using parameter estimation techniques.

A nonlinear fault detection, isolation, and recovery (FDIR) for satellites models is presented in [98]. FDIR involves the construction of residual generators that are based on least-squares parameter estimation techniques. In addition, a fault recovery

procedure is also presented; it works by estimating system parameters and adaptively redesigning and reconfiguring the controller.

Motor faults as inter-turn short circuit and increased winding resistance have been detected in [99]. An adaptive Kalman filter is applied for recursively estimating system parameters for fault detection.

- **Parity space FDI**

In this FDI methodology, parity-functions are used to generate residuals using input-output data. Parity space FDI techniques check the consistency between different set of measurements collected over a specified interval of time. A detailed discussion of the parity space FDI technique is presented in [100]. The definition of Parity functions and Parity equations was first presented by Potter and Suman in 1977 [101]. This paper represents the first introduction of the analytical redundancy concept. Then, a generalized form of parity space for fault detection and residual generation was first introduced by Chow and Willsky in 1984 [102]. Parity space FDI were originally developed and applied to linear systems then extended to non-linear ones in [103].

The most important advantage of using parity equations is that they involve simple mathematics [100]. In addition, all residuals and parity functions can be represented by only one vector which makes it easy to manipulate. However, Parity

space techniques are more sensitive to uncertainties and sensor measurements noise than observer-based ones [61].

A universal Parity space approach, which represents an extension of the original parity equations, was introduced by Hofling in 1993 [104]. This research introduced parity space equations that are valid for continuous linear systems besides being valid for discrete systems.

Fault detection using Parity space approaches have been discussed in several survey papers [105,106,107]. In [107], Patton and Chen provided a review of parity space FDI approaches for the aerospace industry. This paper concludes that residual generation using parity space provides a robust method for differentiating change caused by external effects versus system's uncertainties.

A parity space approach for fault detection and isolation (FDI) of a cryogenic rocket engine combustion chamber is presented in [108]. This methodology can generate residuals based on three measurements followed by the application of spherical co-ordinates to map these residuals to faults. The fault detection methodologies are evaluated based on three main indicators, namely, probability to generate false alarms, missed alarms, and time taken to perform fault detection.

Parity space-based FDI to detect faults in rotary systems is presented by Bachschmid et al in [109]. This FDI methodology can detect rotor cracks that occur due to thermal effects using vibration measurements. Furthermore, the authors present a

more advanced FDI technique that detects the depth of the crack in [110]. The proposed method can identify the position of the rotor cracks and depth using vibration measurements with high accuracy. By comparing the static bending moment to the identified periodical bending moment (which indicates the presence of cracks), the crack depth is estimated. Experimental results show that the proposed methodology is able to detect crack depths with high accuracy.

2.2.4 Artificial Intelligence Techniques

Since the 1990s, Artificial intelligence-based fault detection and diagnosis approaches have been broadly established and applied to various complex engineering problems. Patton and Lopez [111] provided a general overview of artificial intelligence-based techniques used for fault detection and diagnosis. Artificial intelligence-based FDI are divided into two main categories: ***Neural Networks-based*** and ***Expert system-based***. The following sections discuss these two techniques in more details.

2.2.4.1 Artificial Neural Networks (ANNs)-based FDI

ANNs have been successfully applied to a broad spectrum of real world data intensive applications including system identification, pattern recognition, classification, filtering, data clustering, and feature extraction. Accordingly, ANNs provide a powerful tool for fault detection and prognosis. Throughout the literature, three main FDI approaches where ANNs are prevalent are as follows:

- **ANNs for fault modeling and residual generation**

In this approach, an ANNs-based diagnostic system learns the non-linear dynamics using input–output training data sets. A mathematical model representing the healthy system is obtained and the baseline model acts as a reference point for fault diagnosis. More precisely, an occurrence of fault is detected whenever a measurement violates the expected value obtained from the ANN model. The process starts by applying the same input to both the actual physical system and the trained ANN representing the healthy system model. The outputs are compared and an alarm signal is generated if a fault is detected. Ideally the residual block output should be zero and non-zero otherwise.

As stated in the literature, models that are generated by using Neural Networks can be more accurate than those generated by traditional system identification and estimation techniques [112]. This is due to the fact that ANNs have powerful self-learning and self-adapting characteristics, effective online adaptation algorithms besides their parallel and pipeline processing characteristics, good noise rejection capabilities, and excellent nonlinear approximation properties [5]. Furthermore, ANNs provide the aptitude to include models with partly known physical structure, resulting in semi-physical models (*Wang et al.* [113]). There, a blend of physical modelling of main features followed by modeling of secondary effects by ANNs results in an enhanced overall performance.

The NN-based non-linear system identification process can be classified into three categories according to the network structure [114]. The first category is the ***Recurrent Multi-Layer Perceptron (RMLP)***. This network has been established by K. Funahashi and Y. Nakamura [115]. In this research, they proved that any finite time continuous function can be approximately modeled by the internal states of a RMLP network given a sufficient number of output units, hidden layers, and proper initial conditions. In addition, a real-time RMLP that captures the non-linear dynamic behaviour is presented in [116,117]. Talebi and Korasani applied a recurrent Neural Network-based sensor and actuator FDI to detect faults in a satellite's attitude control subsystem using partial state measurements [118].

The second category is by using a ***static neural network with tapped delay lines (TDL)***, the TDL increase the ability of the network to capture non-linear dynamics [119,120]. The third category is ***dynamic neural networks.*** As shown in Figure 2.7, the neuron transfer function is presented as a discrete or a continuous time dynamic system by incorporating an ARMA-filter within the neuron itself [121]. In this category, the ANN training algorithm adjusts the network parameters including weights and filter coefficients using the provided input-output training data set. Yazdizadeh and Khorasani introduced another form of dynamic neural networks. It consists of an all-pole filter

augmented after the neuron activation function in order to generate a dynamic relationship between neuron inputs and outputs [114].

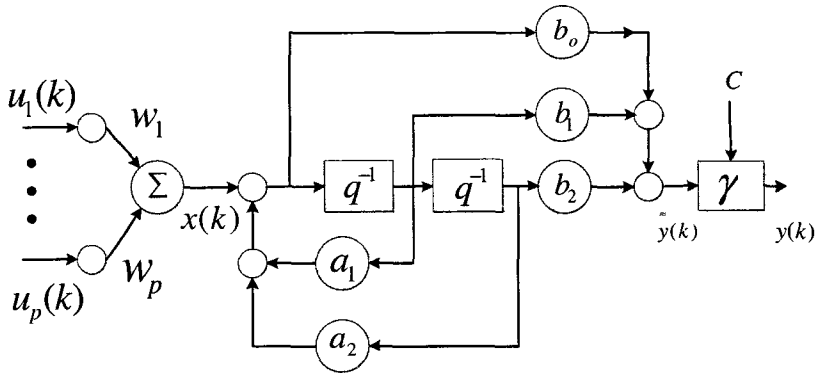

Figure 2.7 Dynamic Elementary Processor with P inputs [12]

Numerous FDI applications that involve ANNs have been reported. Space shuttle main engine (SSME) modeling using feed-forward Neural Network with a sigmoidal activation function is presented in [122]. The Gamma model was introduced by Principe and Motter. The Gamma neural model works by involving an adaptive short memory to store past signal trends. The Gamma model is able to capture substantial features of the system dynamics and is used for non-linear system identification [123,124]. Leonard and Kramer discussed the application of the radial basis function networks (RBFNs) for fault diagnosis and classification [125]. (Naidu et al.) applied back-propagation neural networks for sensor failure detection in process control systems [126]. Terra and Tinos applied Neural Network-based FDI to a 3-joint PUMA manipulator [127]. They used a

multilayer perceptron (MLP) trained with back propagation to reproduce the robot's dynamic behaviour. The ANN outputs are compared with measurements of the joints positions and velocities for residual generation.

- **Neural Networks for decision making (Residual Classification ANN):**

After residual generation (from the previous stage), a trained Neural Networks can be trained to analyze the residuals and decide whether a fault has occurred or not. As shown in Figure 2.8, a two stage neural networks for fault detection and isolation is presented in [61]. The first stage generates the residuals by one of the three previously mentioned methods (i.e.: RMLP, static with TDL, Dynamic NN) followed by a decision-making stage that identifies multiple features and generates fault detection and isolation information.

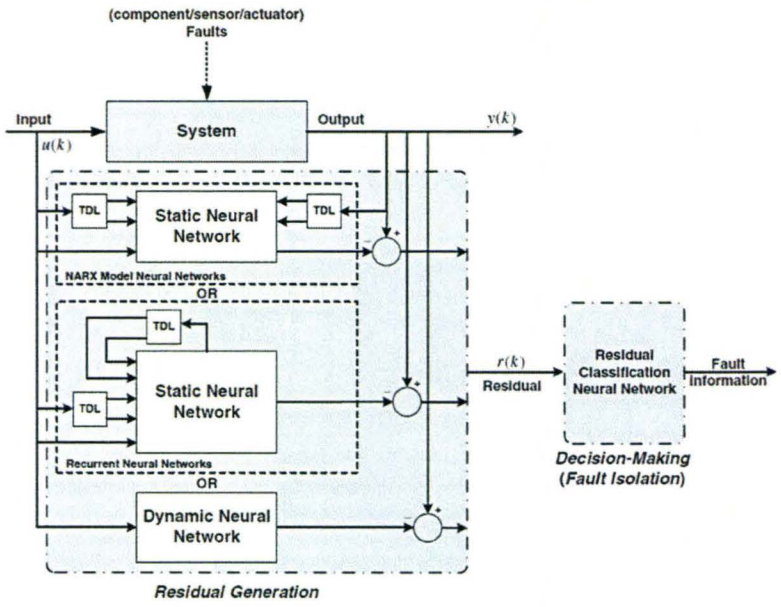

Figure 2.8. Two stage fault detection and isolation [112]

A residual generation and residuals evaluation-based ANNs FDI technique to detect and classify sensor faults is presented in [128]. In this paper, the application of a probabilistic neural network to check temperature sensor readings is performed. This FDI process includes two phases, the first phase involves a comparison between the measurements and trained network predictions to generate residuals. In the second phase, another neural network is used to analyze probable fault conditions by classifying residual patterns. The second phase ANNs has to be trained first to map known residual patterns to assigned faults that might occur in the measurement system.

A neural network-based residual evaluation technique to detect and classify faults online in an industrial actuator benchmark problem is presented in [129]. The actuator in this benchmark problem is a brushless synchronous DC motor. Two faults are considered, namely, actuator current fault due to an end-stop switch and position sensor (potentiometer) fault. Results show that the proposed ANN algorithm can predict and classify faults with an acceptable accuracy.

An FDI methodology to detect faults in robotic manipulators for non-trained trajectories is presented in [130]. Two-level neural networks are used for residual generation and residual evaluations. False alarms that occur due to modeling errors are avoided since the FDI methodology does not use a model.

- **Neural Networks for fault feature extraction and pattern recognition:**

Pattern recognition has been a subject of concern in several engineering disciplines. Pattern recognition is the process of mapping patterns to various groups or categories [131]. It aims at classifying patterns to groups that share the same set of properties [132]. It has been implemented in many applications such as machine vision [133] , speech recognition [134], image processing and analysis [135], medical diagnosis [136], and fault detection and diagnostics [137].

Pattern recognition can be divided into two subcategories, *supervised* and *unsupervised pattern recognition* [132]. In supervised pattern recognition, which is known as pattern classification, input-output training data is clearly provided and labeled. A classifier is then designed to properly classify inputs to categories (or classes) through an iterative learning procedure. In unsupervised learning (clustering), the training data set is not labeled (inputs don't belong to a known class) and the algorithm is supposed to catch intrinsic patterns that can be used later to classify new input patterns.

This research project is concerned with supervised pattern classification. Throughout the literature, various pattern classification techniques have been presented such as linear discriminant methods (support vector machines (SVM) and Fisher's linear discrimination [138]), statistical classification techniques (Bayesian and k-nearest neighbour) [139], and nonlinear discriminant methods (artificial neural networks (ANNs) and decision trees) [140].

Artificial Neural Networks for pattern classification is our main research focus. In pattern recognition approaches, Neural Networks are used as a fault classifier to detect the probability of a fault occurrence or a system abnormality.

Published work in this area of research includes a robust multi-sensor FDI technique applied to marine diesel engine is presented in [141]. Here, four engine faults

are induced and since all of these malfunctions affect pressure, temperature, and vibration measurements, an ensemble of three neural networks are trained, one for each of the three mentioned signals. Consequently, a system that is resistant to sensor failure is obtained. The output of the ensemble is created by taking a majority vote over the three networks.

The Neural Network-based fault detection algorithm has also been applied to locomotive diesel engines to detect lubrication faults using vibrations frequency domain analysis in [142]. Experimental result on more than 50 lubrication pumps shows the effectiveness of the proposed methodology.

Neural networks with N-version programming have been applied in [143]. The idea of N-version programming is to train multiple networks with different measurements signals (e.g.: pressure, vibration, and temperature) for detecting fault conditions and selecting using a voting system for fault classification. This technique can be applied for safety critical systems [144].

Detection of motor bearing faults using signal-based ANN FDI technique is discussed in [145]. Since motors faults are closely related to the bearing assembly, fault detection in motor bearings is important. In this research, networks are trained using vibration data and results show the effectiveness of the proposed ANN method.

A comparison between different fault detection techniques has been published. Rengaswamy provided a comparative study between model-based and Neural Network-

based diagnostic methods in an industrial case study showing the relative advantages and drawbacks of the two approaches [146]. Advantages and drawbacks of neural networks-based FDI as reported by Venkatasubramanian et al., and Katipamula et al, and Rengaswamy in [72,73,147] are listed below:

Advantages:

(1) Neural networks-based FDI are appropriate for complex systems where attaining a system model is hard.

(2) ANNs are appropriate for systems where training data is easy to acquire.

(3) ANNs have less computational requirements.

(4) Systems' physical knowledge is not essential.

Disadvantages:

(1) A knowledge of all possible faults that might occur must be known beforehand to train the network.

(2) If a fault is not included in the training data, fault detection is not attained.

(3) After network training, it is hard to relate network's weights to physical parameters (Block-box modeling)

2.2.4.2 Expert systems (knowledge-based approaches)

Expert systems or knowledge based approaches use heuristics to associate symptoms to faults. The diagnostic expert system imitates the way humans think and how decisions are made using available information.

In expert systems FDI, the a-priori knowledge is presented in the form of causal relationships between faults and symptoms [148].

Figure 2.9 shows a simple fault-symptom-tree, it consists of a knowledge base that stores relationships between faults F_j and symptoms S_j. These relations are learned (or trained) experimentally and stored in the diagnosis system database in the form of rules:

IF <condition> THEN <conclusion>

The condition part is called the premise which contains facts on symptoms S_i and, the conclusion part indicates the events E_j. A combination of different events may indicate a fault F_j, these combinations are in the form of AND, OR connections as follows:

$$IF <S_1\ OR\ S_2>\ THEN <E_1>$$

$$IF <S_1\ AND\ S_2>\ THEN <E_2>$$

$$IF <E_1 \; AND \; E_2> \; THEN <F_1>$$

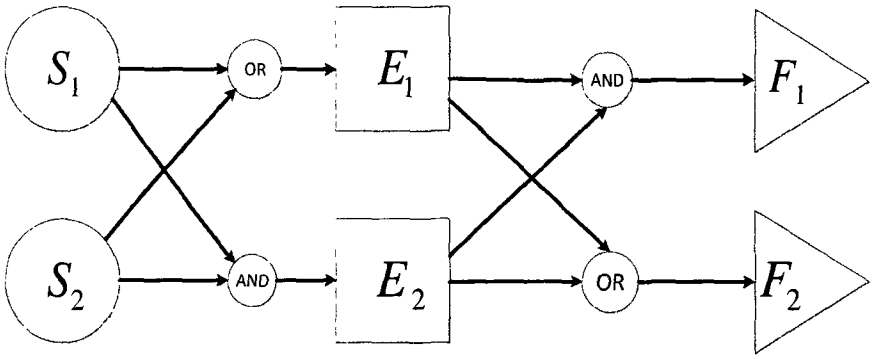

Figure 2.9. Fault symptom tree [82]

In the classical fault-tree analysis, symptoms, events, and faults are considered as binary variables. The condition part of the rules can be calculated by Boolean equations [149].

Fuzzy logic was first introduced in 1967 [150], fuzzy logic gained considerable attention among researchers due to its ability to model human reasoning in a linguistic form. Fuzzy logic has been applied in numerous applications especially in Fault detection and isolation. In fuzzy logic FDI, unlike crisp logic-based FDI, where the system state is either 0 (fault-free) or 1 (faulty system), the system's state can range in degree between 0 and 1. Fuzzy logic FDI is used to express transitions from normal operation to faulty cases. In other words, system under consideration might not be "fault-free" and at the same time they are not totally "damaged" but they are in an intermediate state. Fuzzy

logic FDI provides each item with a certain membership degree in a given set as shown in Figure 2.10.

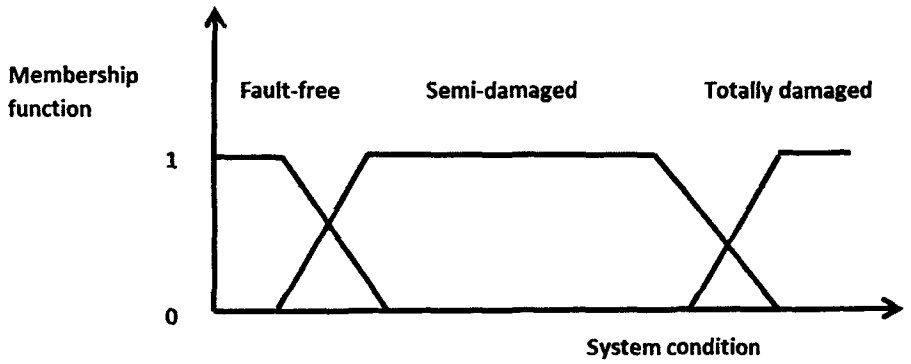

Figure 2.10. Fuzzy membership functions for System Condition

A fuzzy logic-based diagnosis has some advantages; it allows for an easy incorporation of a-priori known rules, enables the user to understand the inference of the system and provide a high degree of transparency and an increased robustness.

In the literature, several heuristic knowledge-based FDI have been published. Fuzzy Logic-based fault detection in a three phase induction motor has been presented in [151]. This FDI system can perform an on-line condition monitoring using a fuzzy logic controller.

A fault detection strategy on a gas turbine engines using fuzzy logic has been presented in [152]. Engine model parameter residuals acquired during the flight cycle

are used as FDI system inputs to prognoses high-pressure spool failure. The presented FDI methodology shows reliable prediction horizons using few system parameters. The FDI system is modular and can be applied to any engine with similar characteristics.

A fuzzy logic-based FDI in a hydraulic system is discussed in [153]. Simulation results show the ability of the proposed FDI technique to detect the severity of the fault at a high level of accuracy using residuals generated by a non-linear observer. The difference between the desired and actual velocities is used for residual calculation then these residuals are evaluated using a fuzzy logic inference controller to detect the severity of faults.

A similar approach applied to industrial robots is presented in [91] with the addition of an adaptive threshold that is used to minimize the probability of generating false alarms. Other applications of Fuzzy rule-based diagnosis system include the consideration of turbo machines using observed vibrations in [154], an internal combustion engines as in [155,156], and DC motors in [157].

2.3 The Smooth Variable Structure Filter

The recently proposed SVSF has been successfully applied for parameter and state estimation in several studies [158,85]. The SVSF has demonstrated better performance over the EKF in target tracking applications in terms of computational complexity and robustness to model uncertainties [159].

The SVSF provides an advantage that it is able to extract information on modeling uncertainties using two performance indicators, referred to as the primary and the secondary indicators. The primary indicators are the estimation errors or the error covariance matrix, while the second one includes the chattering signals due to the discontinuous gain application.

The SVSF has been applied to control application and was combined with sliding mode control (SMC) in [160]. A comparative study of the SVSF and the EKF in a nonlinear state estimation problem has demonstrated that both techniques are comparable in terms of accuracy and estimation convergence when applied to a system with a known model [161].

The EKF is marginally more accurate than the SVSF given an accurate model but its performance deteriorates considerably in the presence of uncertainties. Therefore, a combined variable structure and Kalman filtering approach for parameter estimation has been proposed in [162]. In this study, a blend of SVSF-EKF has been introduced to add robustness to the EKF and provide a guaranteed stability. A comparison between the EKF, the SVSF, the Particle Filter, and the unscented Kalman filter on a bearing-Only target tracking problem is demonstrated in [163].

The SVSF concepts are further investigated in [164]. In this research, an enhanced form known as the general TOEPLITZ/OBSERVABILITY Smooth Variable Structure Filter is proposed. This filter introduces the Toeplitz and the Observability

matrices to overcome the SVSF's limitations that arise due to the use of the Luenberger technique regarding sensitivity to noise and modeling errors. In addition, the chattering signals are used to create a monitoring algorithm that is capable of detecting added uncertainties in the system. Furthermore, a novel parameter estimation technique referred to as the Iterative Bi-Section/Shooting Method (IBSS) is derived. The proposed SVSF estimation technique is used to estimate the system's states provided that the model structure only is known.

In this research project, the SVSF is applied for the training of the feed-forward neural network. The SVSF is applied in global (GSVSF), multi-streaming mode. The GSVSF performance is compared against the standard first and second-order derivative BP algorithms, and the EKF on a standard benchmark problem. In addition, the proposed training algorithm is applied to detect and classify faults on a FORD diesel engine using vibration data.

Chapter 3: Multi-Layer Feedforward Networks and their Training Techniques

3.1 Introduction

A multilayer feedforward network, also known as multilayer perceptron network (MLP), is a class of artificial neural networks. It has various applications including pattern recognition, interpolation, and system identification. In addition, MLPs are able to solve extremely complex problems which can imitate human behaviour such as speech processing, machine translation, and image recognition. MLP is a mathematical model that maps a set of inputs to output patterns. MLP entails supervised learning techniques in which a teacher or a pattern of training data set is required to perform the network training.

This research focuses on feed-forward MLP and its training techniques. This chapter, as shown in Figure 3.1, is organized as follows; section 3.2 discusses various MLP configurations including feedforward and feedback MLP networks. Section 3.3 involves feedforward MLP networks and their mathematical representation including single and multi-neuron models in addition to signal follow diagrams. Section 3.4 provides a description of various training algorithms including the first-order back (BP) propagation, Newton's method, Levenberg-Marquardt (LM), and Quasi-Newton (QN). Finally, section

3.5 describes the extended Kalman filter (EKF) and its application to MLP training. These algorithms are applied to benchmark problems later in chapters four and five and compared against the SVSF.

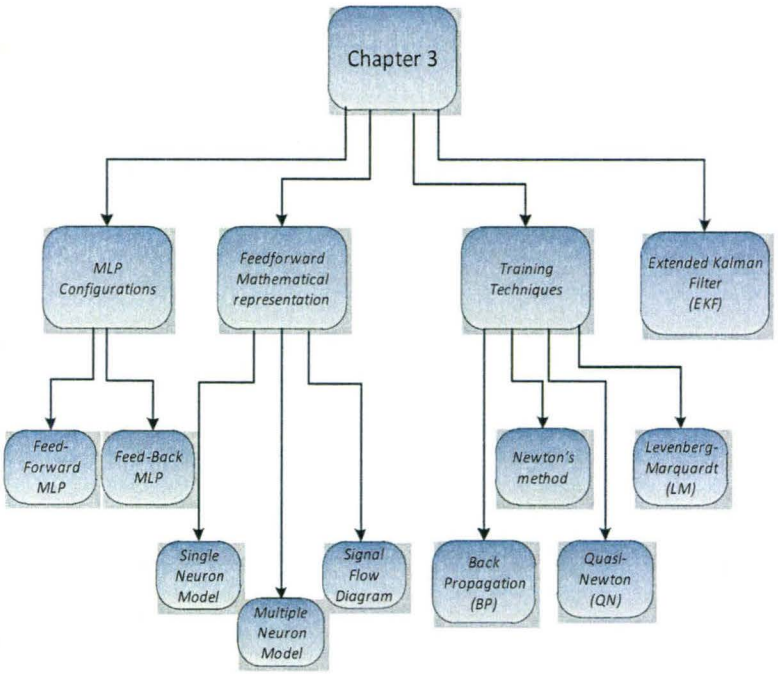

Figure 3.1. Chapter Three Hierarchy

3.2 MLPs Network Configuration

A MLP network, as shown in Figure 3.2, consists of one input layer, a number of hidden layers, and one output layer. Each layer consists of a number of neurons where each neuron is fully (or partially) connected to all neurons in its preceding layer. The network shown in Figure 3.2 is a fully connected multilayer perceptron (MLP) network; a

fully connected network is characterized by a complete connection between every neuron (node) in a layer and its preceding layer.

MLP networks have two categories; feedforward MLP and feedback MLP. In a feedforward MLP, as in Figure 3.2, signals propagate across layers in a forward fashion. In feedback MLP or sometimes called recurrent multilayer perceptron RMLP as shown in Figure 3.3, a signal is fed back from the output layer or from one of the hidden layers and inserted as an input to the network after adding a delay (memory) element (z^{-1}). The delay element introduces an internal state and thus RMLP can introduce dynamic behaviour. Consequently, RMLP are extensively used in handwriting recognition and system identification.

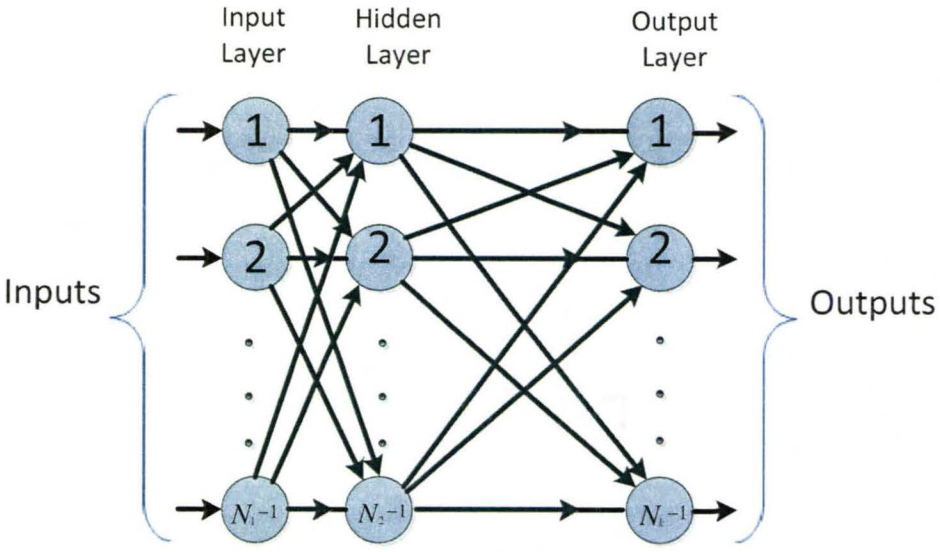

Figure 3.2. Feedforward Multilayer Perceptron Network

Two types of signals are recognized in any MLP, they involve ***function*** signals and ***error*** signals [39]. A function signal originates from the input and crosses the network hidden layers and eventually exits as an output. On the other hand, an error signal arises from the output and propagates in a backward fashion across the network. Error signals are discussed later in this chapter; they are used during network training to adjust network synaptic weights.

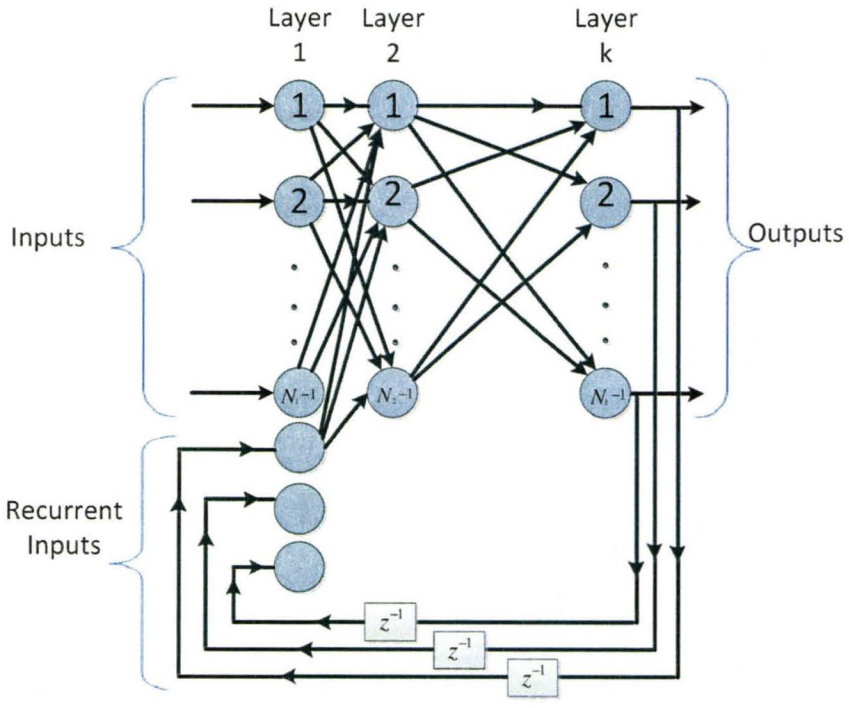

Figure 3.3. Feedback (Recurrent Multilayer Perceptron) Network

3.3 Feed-Forward Multilayer Perceptron Networks

A multi-layer feed-forward network consists mainly of sensory units that constitute the input layer, one or more hidden layers and an output layer. As shown in Figure 3.4, a fully connected MLP where each node is connected to all nodes in the adjacent layer by links (weights), and computes a weighted sum of the inputs. An offset (bias) is added to the resultant sum followed by a nonlinear activation function. The input signal propagates through the network in a forward direction on a layer-by-layer basis. Consequently, the network represents a static mapping between the inputs and the outputs.

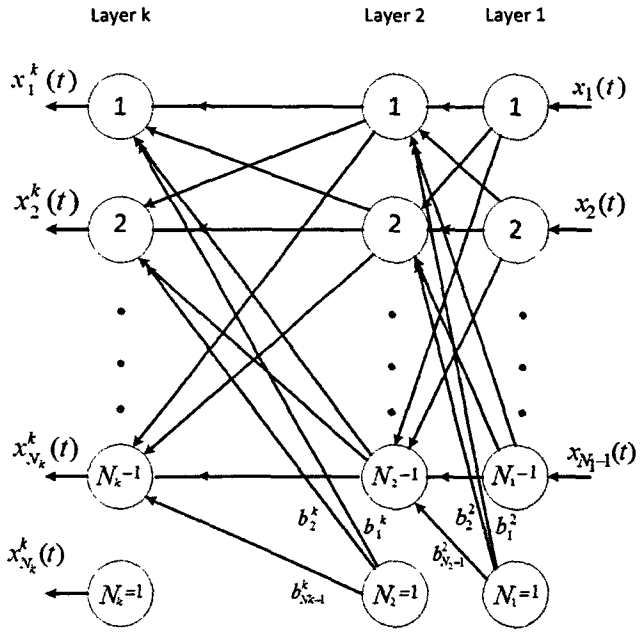

Figure 3.4. Schematic of feed-forward multilayer perceptron network

3.3.1 Single and Multiple Neuron Mathematical Representation

Let k denote the total number of layers, including the input and output layers. Node(n,i) denotes the i^{th} node in the n^{th} layer, and $N_n - 1$ is the total number of nodes in the n^{th} layer. As shown in Figure 3.5, the operation of $node(n+1,i)$ is described by the following equation:

$$x_i^{n+1}(t) = \varphi\left(\sum_{j=1}^{N_n-1} w_{i,j}^n\, x_j^n(t) + b_i^{n+1}\right) \tag{3.1}$$

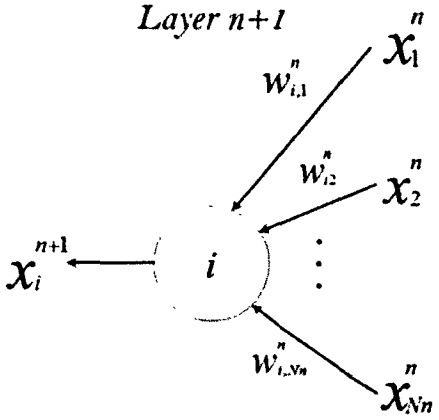

Figure 3.5. Node (n+1, i) representation

Where, $x_i^n(t)$ denotes the output of $node(n,j)$ for the t training pattern, $w_{i,j}^n$ denotes the link weight from $node(n,j)$ to the $node(n+1,i)$. b_i^n is the node offset (bias) for $node(n,i)$. The function $\varphi(.)$ is a nonlinear sigmoid activation function defined by:

$$\varphi(w) = \frac{1}{1 + e^{-aw}} \qquad a > 0 \; and \; -\infty < w < \infty \tag{3.2}$$

For simplicity, the node bias is considered as a link weight by setting the last input N_n to $node(n+1, i)$ to a value of one as follows:

$$x_{N_n}^n(t) = 1, \qquad 1 \leq n \leq k$$

$$w_{i,N_n}^n = b_i^{n+1}, \qquad 1 \leq n \leq k-1$$

Consequently, Equation (3.1) can be rewritten in the following form:

$$x_i^{n+1}(t) = \varphi\left(\sum_{j=1}^{N_n} w_{i,j}^n \, x_j^n(t)\right) \tag{3.3}$$

3.3.2 Signal Flow Representation

The MLP can also be viewed as shown in Figure 3.6 using signal-flow graph. This representation is useful in highlighting connections between neurons for both function and error signals. In addition, most of the training techniques in section 3 will be discussed using signal flow diagrams. The signal-flow diagram notation is as follows [39].

Consider a hidden neuron j being fed by inputs $y_i(n)$. The inputs are then multiplied by the synaptic weights denoted as $w_{ji}(n)$ (in addition to bias $b_j(n)$) and summed together generating a local field signal $\gamma_j(n)$. Thus for a single neuron j, the local filed $\gamma_j(n)$ can be expressed as follows:

$$\gamma_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{3.4}$$

The local field signal $\gamma_j(n)$ represents the weighted summation at epoch n for all weights directed to neuron j. The local field function signal is then processed by the non-linear activation function $\varphi_j(.)$ to generate the output $y_j(n)$.

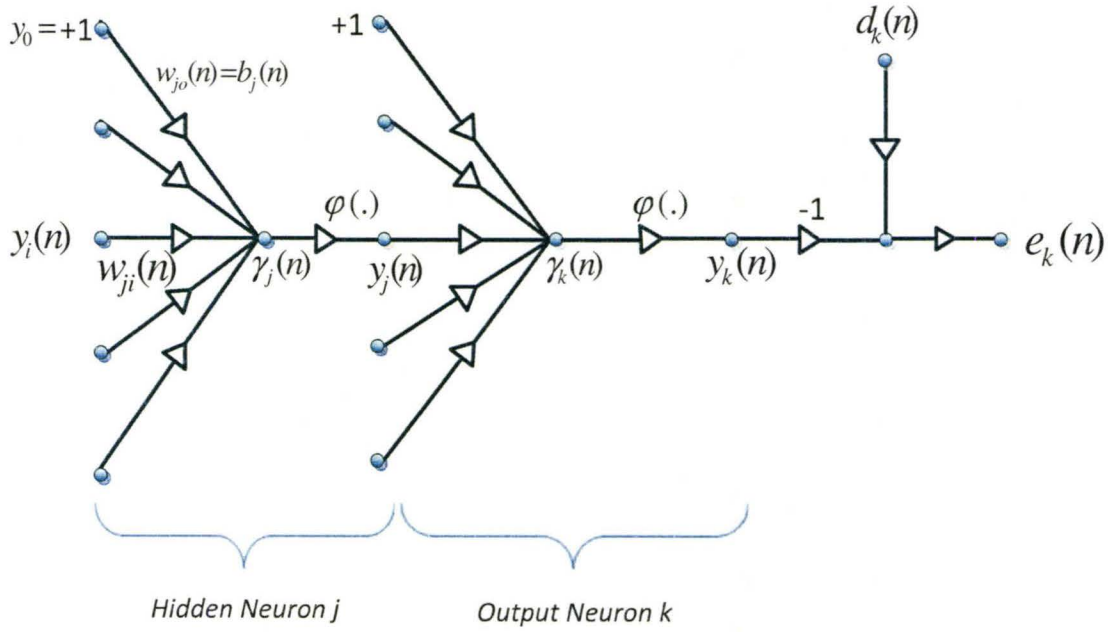

Figure 3.6. Signal flow from a hidden neuron j to the output neuron k [39]

The activation function $\varphi_j(.)$ might be linear or non-linear but it has to be continuous as the derivative is a major requirement in the training process (gradient

calculation). The most frequently used activation function (used throughout the thesis) is the ***sigmoid nonlinear*** function expressed as follows:

$$y_j(n) = \varphi_j\left(\gamma_j(n)\right) = \frac{1}{1 + \exp(-a\,\gamma_j(n))} \tag{3.5}$$

Where, $a > 0$ and $-\infty < \gamma_j(n) < \infty$

Two cases are to be considered here:

Case 1:

If the neuron j denotes the network's input layer, then the signal $y_j(n)$ is denoted as $x_j\,(n)$

$$\text{ie: } y_j(n) = x_j(n) \tag{3.6}$$

Where, $x_j(n)$ is obtained from the training data set.

Case 2:

If the neuron j is in the output layer of the network, then:

$$y_j(n) = o_j(n) \tag{3.7}$$

Where $o_j(n)$ is the j^{th} component of the actual network's output. The error signal $e_k(n)$ is calculated by subtracting the desired output (obtained from the training data set) $d_k(n)$from the actual network output $o_k(n)$ as follows:

$$e_k(n) = d_k(n) - o_k(n) \tag{3.8}$$

The error signal is then used to update network weights as discussed in the following section.

3.4 MLP Training Techniques

In this section, most of the popular neural networks training techniques are discussed. They include the widely used back-Propagation algorithm, Newton method, Quasi-Newton, and Levenberg-Marquardt. These are classical training techniques that have been extensively used throughout the literature in various applications. These training techniques are being compared to the relatively new Smooth Variable Structure Filter (SVSF) later on.

3.4.1 Back Propagation Algorithm

Back propagation (BP) is one of the most famous classical ANNs training techniques widely used in the 80's. It's a supervised learning algorithm that mainly applies the delta rule (or gradient descent rule) to minimize the output error. The delta rule used to update network weights can be generally expressed as follows:

$$\Delta w_{ji} = \alpha(d_j - y_j)\varphi'(h_j)x_i \tag{3.9}$$

Where, α denotes the learning rate, d_j is the target signal, y_j is the neuron output, φ is the activation function, φ' is the first order derivative of the activation function φ, h_j is the weighted sum of the neuron's inputs , and x_i is the i^{th} input.

Note that: $h_j = \sum x_i w_{ji}$ and $y_i = \varphi(h_j)$

The BP algorithm consists mainly of two paths, namely a forward and a backward path. The forward path involves the application of a training data set pattern on the input, the signal then propagates across the network on a layer-by-layer basis until the network's output is generated. The backward path involves the calculation of the output error using the training data set target and propagation of the error back to update weights and thus perform training.

3.4.1.1 BP Training Procedure

Figure 3.7 represents the operation of BP algorithm on a network with three inputs, one hidden layer with three neurons, and three output nodes in the output layer. The upper portion demonstrates the forward pass (feedforward network running phase) that represents function signals stated earlier. During the forward pass, network weights remain unchanged and function signals are generated on a layer-by-layer basis until the actual network's output is generated. The lower portion depicts the backward pass. The backward pass includes error signals where all local gradients are calculated in order to update network weights.

Assume N training samples in the following form:

$$\{x(n), d(n)\}^{N}{}_{n=1}$$

Where, $x(n)$ represents the network input and $d(n)$ depicts the desired target at the n^{th} training sample. Assuming a sequential training fashion, back propagation algorithm is summarized as follows [39]:

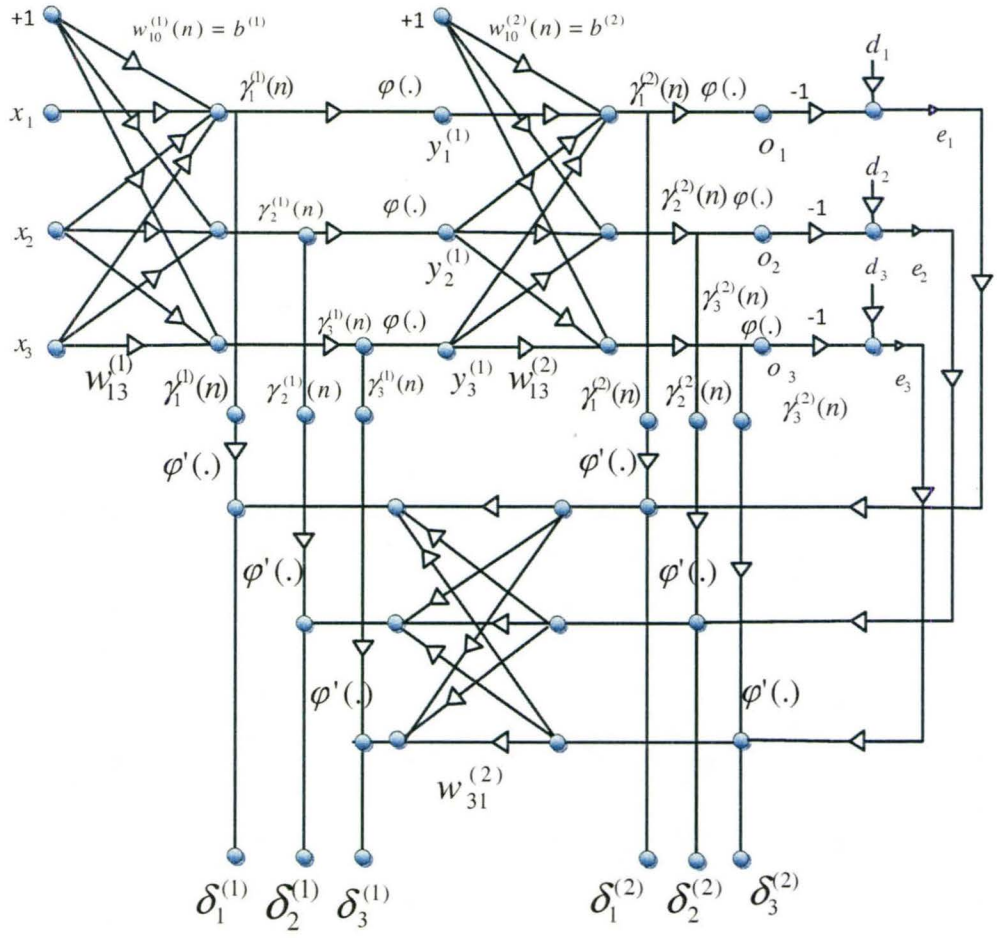

Figure 3.7. Signal flow using back propagation algorithm [39]

Step 1: Initialization

The network weights and biases are randomly initialized using a uniform distribution with zero mean and unity standard deviation. A random initialization strategy reduces the probability of premature early convergence to local minima.

Step 2: Forward Run

It involves applying an input signal to generate the output by first computing local fields (weighted sum before activation function) followed by applying the activation function on a layer-by-layer basis as discussed in the previous section. Let the local fields for neuron j in layer l for the training sample n be denoted as: $\gamma_j^{(l)}(n)$ and expressed as:

$$\gamma_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \tag{3.10}$$

A non-linear sigmoidal activation function $\varphi(.)$ is then applied to generate the neuron's j output as follows:

$$y_j^{(l)} = \varphi(\gamma_j(n)) \tag{3.11}$$

Where,

- $w_{ji}^{(l)}(n)$ is the weight from neuron i in layer $l-1$ directed towards neuron j in layer l.

- $y_i^{(l-1)}(n)$ is the i^{th} neuron's function signal of the layer $(l-1)$ calculated at the n^{th} epochs.

As stated in section 3.2, for an easier representation of the network bias, the bias is considered as a network's weight while setting an input to unity. As shown in Figure 3.7 for $i = 0$:

$$w_{j0}^{(l)}(n) = b_j^{(l)}(n) \tag{3.12}$$

$$y_0^{(l-1)}(n) = +1 \tag{3.13}$$

For the two cases stated earlier in the previous section, if the neuron j is located in the input layer, the input layer is treated as $l = 0$, and relabelled such that:

$$y_j^{(0)}(n) = x_j(n) \tag{3.14}$$

While if the neuron j is positioned in the output layer, the variable l is set to L where L is the number of layers, and the output is relabelled such that:

$$y_j^{(L)}(n) = o_j(n) \tag{3.15}$$

In our example as shown in Figure 3.7, L is set to 2 and each layer is made to have 3 neurons with $m_0 = m_1 = m_2 = 3$

Step 3: Error Calculation

For a feedforward MLP network, assume the output of neuron j at iteration k is $o_j(n)$. Assume that the target at n's training sample is $d_j(n)$. The error signal for sample n is defined as:

$$e_j(n) = d_j(n) - o_j(n) \tag{3.16}$$

Step 4: Gradient Calculation (Backward Pass)

Network's local gradients are calculated in this step. Two cases are considered here depending on the neurons' location in the hidden layer or on the output layer as follows:

Case 1: If the neuron j *is located in the output layer 'L':*

$$\delta_{\mathrm{j}}^{(\mathrm{l})}(\mathrm{n}) = \mathrm{e}_{\mathrm{j}}^{(\mathrm{L})}(\mathrm{n})\varphi'_{\mathrm{j}}(\gamma_{\mathrm{j}}^{(\mathrm{L})}(\mathrm{n})) \tag{3.17}$$

Case 2: if the neuron j is placed in the hidden layer 'l':

As shown in Figure 3.7, local gradients are calculated using the following equation [40]

$$\delta_j^{(l)}(n) = \varphi'_{\mathrm{j}}\left(\gamma_j^{(l)}(n)\right)\sum_k \delta_k^{(l+1)}(n)\, w_{kj}^{(l+1)}(n) \tag{3.18}$$

Where the derivative $\varphi'(.)$ for the sigmoidal activation function is calculated as:

$$\varphi'\left(\gamma_j(n)\right) = \frac{a\exp(-a\gamma_j(n))}{[1 + exp(-a\gamma_j(n))]^{\,2}} \tag{3.19}$$

In a simpler form, by removing the exponential term from the previous equation as follows:

$$\varphi'\left(\gamma_j(n)\right) = \frac{a\left[1+\exp\left(-a\gamma_j(n)\right)-1\right]}{[1+exp(-a\gamma_j(n))]^{\,2}}$$
$$= a\left[\frac{1}{1+exp(-a\gamma_j(n))} - \frac{1}{[1+exp(-a\gamma_j(n))]^{\,2}}\right] \qquad (3.20)$$
$$= a y_i(n)[1-y_j(n)]$$

Where, $y_j(n) = \varphi_j\left(\gamma_j(n)\right)$

Step 5: Weight Update

The network's synaptic weights are adjusted using the delta rule as discussed earlier and as follows:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \mu\left[w_{ji}^{(l)}(n-1)\right] + \beta\delta_j^{(l)}(n)y_i^{(l-1)}(n) \qquad (3.21)$$

Where, β is the learning rate and μ is the momentum constant. The momentum introduces the old weight change in computing the new weight change. Accordingly, the momentum is used for stabilizing oscillations in case of problems with a narrow minimum zone thus speed up convergence. The momentum constant works as follows, in areas where the error surface is flat the algorithm uses large steps and traverse these areas rapidly. On the other hand, when the surface is rough, the step size is decreased thus the algorithm speed is enhanced. The momentum constant is selected as follows:

$$0 < \mu < 1$$

Step 6: Shuffling Training Data and Learning Rate Annealing

Training data sets are shuffled every epoch (following each weight update) in addition to annealing of learning rate and momentum. Shuffling is important to make sure that training data set correctly represents the data mix thus helps in removing any drift or bias that might occur if the same order of the training data set is applied every epoch. Annealing means that the values are decreased by a constant value from one epoch to another. For example, throughout the thesis, the learning rate β is primarily set to one and linearly reduced to 10^{-4} in order to prevent over-training.

Steps 2-5 are iteratively repeated until a stopping criteria is met as follows.

3.4.1.2 Training Algorithm Stopping Criteria

During neural networks training, the training data set is divided into 50%, 25%, 25% of the overall content for training, validation, and testing, respectively. Training portion is used to calculate the gradient and for weight and bias update. Validation set is used to perform cross-validation checks which are executed to assess training quality as training proceeds. Cross-validation is implemented to overcome over-fitting (over-training) [165]. Over-fitting may happen when the training algorithm focuses on training set peculiarities at the cost of losing generalization ability [166]. In other words, the trained network mean squared error (MSE) might be low during training but afterwards during testing the network would exhibit poor generalization performance.

In this paper, cross-validation is performed by using an early stopping technique. Early stopping is accomplished by constantly observing the mean square error. In normal cases, validation error should be decreasing as training proceeds but if over-fitting occurs, the error on the validation set starts to increase. Training is stopped whenever one of following three conditions occurs: If the GL_α stopping criterion discussed below is achieved or if the training progress is below a specified value or if maximum number of epochs are reached. The three stopping criteria are further discussed below.

For the first stopping criterion, training is terminated when the Generalization Loss (GL) becomes higher than a certain threshold. According to [165], the generalization loss at epoch t is defined as the relative increase of the validation error over the minimum-so-far (in percent).

$$GL(t) = 100\left(\frac{E_{valid}(t)}{E_{Optimal}(t)} - 1\right) \tag{3.22}$$

$$E_{Optimal}(t) = \min_{t' \leq t} E_{valid}(t') \tag{3.23}$$

Where, $E_{valid}(t)$ is the validation set squared error at epoch t, where E is the error function defined later in (3.25). $E_{Optimal}(t)$ is the least validation set squared error obtained up to epoch t. When high generalization loss occurs, it indicates that the algorithm is focusing on training signal peculiarities and ignoring general regular

behavior. Accordingly, training should be terminated whenever a GL exceeds a pre-determined threshold value α (set to 0.01).

The second stopping criterion is called training progress. It is performed by calculating the squared error of training and validation data set after a pre-specified number of epochs called the strip length k (set to five epochs in this paper). The sequence is numbered as $n+1, n+2, .. n+k$ provided that n is divisible by k. Training progress is calculated using the following formula:

$$P_k(t) = 1000.\left(\frac{\sum_{t' \in t-k+1 \ldots t} E_{train}(t')}{k \,.\, min_{t'. \in t-k+1 \ldots t} E_{train}(t')} - 1\right) \tag{3.24}$$

This ratio specifies the relation between the average and minimum training error that occurs within one strip.

The network performance function (error function) used is the mean squared error between the network and target outputs. For the i^{th} output node, the squared error per training sample is calculated as follows:

$$E(o,t) = \sum_i (o_i - t_i)^2 \tag{3.25}$$

Where o_i is output target node and t_i is the target. MSE is calculated using the whole data set. The mean squared error is selected as it is independent of the training

data set size. Error normalization is performed and squared error percentage is calculated as follows [165]:

$$E = 100\frac{o_{max} - o_{min}}{N.P}\sum_{P=1}^{P}\sum_{i=1}^{N}\left(o_{p_i} - t_{P_i}\right)^2 \tag{3.26}$$

Where, N represents the number of output nodes. P indicates number of training samples used. o_{max} and o_{min} represent maximum and minimum values for the output nodes, respectively. In addition to MSE variations during training, training algorithms are evaluated according to classification performance [167]. According to the benchmark rules, training performance is measured based upon classification accuracy. It represents the percentage of wrongly classified data points.

The third stopping criterion is when the maximum number of epochs is attained. The selection of the maximum number of epochs depends on the network's application and on the complexity of the problem. Training is limited to 1000 epochs in this research.

3.4.2 Optimization-Based Techniques

The basic idea of training is to iteratively update network's weights in order to minimize the output error using a given training data set. Consequently, training process can be visualized as an optimization problem whose objective is to find the network's best

weights that minimize error. Various optimization techniques have been proposed for network training in the literature; Newton's method for instance is one of the most prominently applied optimization techniques. In addition, Quasi-Newton, Levenberg-Marquardt, and Gauss Newton are also commonly applied to train feedforward MLP networks.

In order to apply any optimization techniques, a cost function (or objective function) must be defined. The cost function determines the effectiveness of the optimization technique. The selection of the cost function is crucial as it would reflect the required objective. In other words, in neural networks, selection of the error function specifies the objective of the search. If this goal is not properly specified, poor results will be attained. The 'mean square error' is one of the most prevalent cost functions used in regression problems.

Numerous optimization techniques have been proposed for network training [168,169,170] . Ideally, they seek the global minimum. In practice, optimization algorithms vary in probability of discovering the global minimum, their rate of convergence, and their robustness [171]. Algorithms might get stuck in local minima and fail to find the global minimum. Local and global minima are as shown in Figure3.8 below.

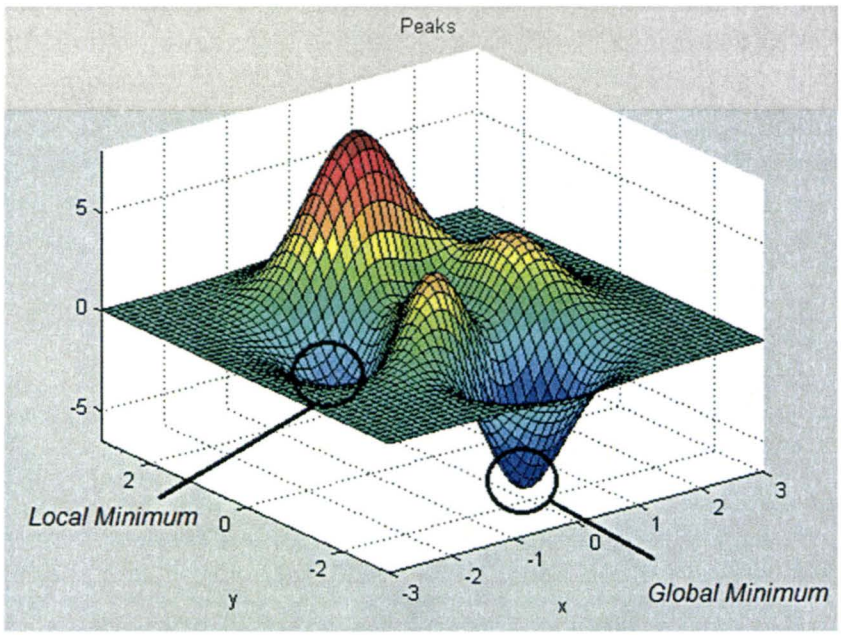

Figure3.8. Local and Global minima [172]

Optimization techniques are more widely used in neural networks training compared to first-order BP algorithm due to their comparatively faster speed of convergence. In this section, a detailed discussion of various optimization techniques that are frequently used in neural networks training is discussed. More specifically, Quasi-Newton and Levenberg Marquardt methods are described. The performance of these techniques is later compared to a new method proposed in this research based on the Smooth Variable Structure filter (SVSF).

As a start, Newton's method is being highlighted as it provides the basis of all other second-order optimization techniques because it makes use of first and second

order derivative information. Consequently, it represents a benchmark where all other optimization techniques are compared against.

3.4.2.1 Newton's method

Optimization techniques in general are categorized into first order and second order. In first-order optimization techniques, the error surface is approximated to be linear (line search). The cost function and its first order derivatives only are used. While in second-order optimization techniques, a quadratic error surface is assumed and the second order derivative of the cost function is also utilized. Newton's method is a form of second-order optimization method that achieves a faster speed of convergence compared to the back propagation algorithm. Second-order derivatives are stored in a matrix form widely known as the ***Hessian matrix.*** The Hessian matrix is a square matrix that involves a given function's local curvature by including the function's second-order partial derivatives. Let D_i be the differentiation operator with respect to the i^{th} element, for a function $f(x)$, the Hessian matrix can be obtained as follows:

$$H(f)_{ij}(x) = D_i D_j f(x) \tag{3.27}$$

Second order optimization techniques are very effective in the "neighbourhood of the global minimum". Using Newton's method, if the approximated error function is originally quadratic, the solution (global minimum) can be achieved in one step. On a

larger scale, second order techniques are not efficient as they might get stuck in a local minimum rather than find the global one.

Newton's method uses and stores the full Hessian matrix which is a computationally expensive process. Newton's method uses a quadratic error function as follows [171]:

$$E(w) = E_o + g^T w + \frac{1}{2} w^T H w \tag{3.28}$$

Where,

- g: is the gradient vector ($\frac{\partial E}{\partial w}$)
- H: is the cost function's second-order derivatives matrix (Hessian matrix) with elements $h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$ (*H must be a positive definite matrix)*

By differentiating equation (3.28) with respect to w and equating the resultant to zero, then:

$$g + Hw = 0 \tag{3.29}$$

By re-arranging, and given that H is positive definite, the solution is obtained as follows:

$$w^* = -H^{-1} g \tag{3.30}$$

Thus if the function $E(w)$ is originally quadratic, the global minimum can be attained in only one epoch. Various iterations might be required if the function is not perfectly quadratic. When several iterations are required, a step size parameter τ must

be included and it is being selected by the user. For multiple iterations, the following formula used is:

$$w(t+1) = w(t) - \tau H^{-1} g \tag{3.31}$$

The advantages of Newton's method are:

1. Provides fast speed of convergence in the neighbourhood of the minimum. For quadratic error functions, given that $e = w - w^*$, the optimization process is quadratic and takes the form:

$$|e_{k+1}| \leq C|e_k|^2 \tag{3.32}$$

2. It acts as a benchmark for comparison against other optimization techniques as it makes use of the full Hessian matrix.

The disadvantages of Newton's method are:

1. It requires the Hessian matrix calculation and it must be positive definite. This is not usually the case in non-linear systems.
2. The Hessian matrix H might be rank deficient in neural networks applications. In this case, saturation of the network's hidden neurons might occur.
3. Selection of the step size is crucial, as τ should be selected as follows:

$$0 < \tau < 2/\beta_{max} \tag{3.23}$$

Where, $\beta_{\max}$ is the Hessian matrix's largest Eigen value. Eigen values of a matrix A represent the solution $\boldsymbol{\lambda}$ to the equation:

$$\det(\mathrm{A} - \boldsymbol{\lambda}\mathrm{I}) = 0 \tag{3.34}$$

If the step size value is selected to be large for fast convergence, then the system might overshoot beyond the neighbourhood of the minimum. Accordingly, the step size must be chosen small enough so that the system does not overshoot the minimum.

As a result of the above deficiency, the Quasi-Newton algorithm has been proposed.

3.4.2.2 Quasi-Newton (Variable metric) algorithm

Quasi-newton (QN) method overcomes the problems related to the Newton's method. It basically overcomes the problem of computing and getting the Hessian matrix's inverse. By utilizing first order gradients only, an approximation of the Hessian matrix is obtained. The most famous Quasi-Newton algorithm (also used throughout the thesis) is known as BFGS and was developed in 1970 [173]. BFGS stands for Broyden, Fletcher, Goldfarb, and Shanno.

QN technique provides fast speed of convergence compared to the back propagation algorithm. However, one of its main drawbacks is that there is no guarantee

that the algorithm will converge to the global minimum as it might get stuck in a local minimum [39,171]. QN algorithm is summarized as follows:

Step 1: Random initialization of the matrix S

The S matrix is used later in calculating the direction vector. $S(0)$ must be a positive definite matrix.

Step 2: Calculation of the direction vector

Define the direction vector $s(n)$ as follows:

$$s(n) = -S(n)g(n) \tag{3.35}$$

Where $g(n)$ is the first-order gradient vector. The matrix $S(n)$ is a positive definite matrix and throughout every epoch it is being tuned such that the direction vector $s(n)$ points to the same direction as the newton's method (direction to the global minimum).

Step 3: Weight update equation

Weight update is performed by knowing the direction vector $s(n)$ and the learning rate parameter $\mu(n)$as follows:

$$w(n+1) = w(n) + s(n)\mu(n) \tag{3.36}$$

The $\mu(n)$ is set to 1 in case of applying the BFGS-Quasi Newton algorithm.

Step 4: Getting second order (Curvature) information

Second order information is obtained without the need of computing the Hessian matrix. This requires the knowledge of four major variables:

$$w(n),\ \ w(n+1),\ \ g(n),\ \ g(n+1)$$

First, the difference between the current weights and the previous weights is calculated:

$$\Delta w(n) = w(n+1) - w(n) \tag{3.37}$$

followed by the difference between two successive gradients:

$$q(n) = g(n+1) - g(n) \tag{3.38}$$

Note that $q(n)$ can be approximated as:

$$q(n) \cong (\frac{\partial}{\partial w} g(n))\Delta w(n) \tag{3.39}$$

Step 5: Hessian matrix approximation

If the objective function is quadratic, the two following equations provide the exact Hessian matrix. However, if the objective function is not quadratic (which is the case of non-linear neural networks), the following equations can be used to get an approximation of the Hessian matrix H:

$$H \cong [q(0), q(1), \dots, q(W-1)] * \tag{3.40}$$

$$[\Delta w(0), \Delta w(1), \dots, \Delta w(W-1)]^{-1}$$

The inverse Hessian matrix H^{-1} can in turn be approximated as follows:

$$H^{-1} \cong [\Delta w(0), \Delta w(1), \dots, \Delta w(W-1)][q(0), q(1), \dots, q(W-1)]^{-1} \quad (3.41)$$

Step 6: Update the matrix $S(n+1)$ from its previous value $S(n)$ using the following recursive formula:

$$(n+1) = S(n) + \frac{\Delta w(n)\Delta w^T(n)}{q^T(n)q(n)} - \frac{S(n)q(n)q^T(n)S(n)}{q^T(n)S(n)q(n)} + \delta(n)[q^T(n)S(n)q(n)][\mathrm{v}(n)\mathrm{v}^{\mathrm{T}}(n)] \quad (3.42)$$

Under condition that:

$$0 \leq \delta(n) \leq 1 \ \textit{for all } n$$

Where,

$$\mathrm{v}(n) = \frac{\Delta w(n)}{\Delta w^T(n)\Delta w(n)} - \frac{S(n)q(n)}{q^T(n)S(n)q(n)} \quad (3.43)$$

Step 7: Iterative repetition until the global minimum is achieved.

The algorithm is iteratively repeated until a stopping condition is reached.

3.4.2.3 Levenberg Marquardt

The Levenberg-Marquardt applies the two main advantages of the gradient descent and second-order methods. The Levenberg-Marquardt switches from the gradient descent method to Newton's method as it reaches the vicinity of the minimum.

Levenberg-Marquardt is a blend of the gradient descent algorithm and Newton's algorithm. It overcomes the drawbacks of Newton's method where the Hessian matrix has to be positive definite. In general, when the optimization process starts at a remote

point away from the optimal global minimum, the Hessian matrix tends to be indefinite and thus Newton's method might provide poor convergence properties [171]. Accordingly, the gradient method is used first as it guarantees convergence to the vicinity of a minimum irrespective of where the optimization process starts. After that, the algorithm shifts to Newton's method as it is much faster than gradient methods.

The Levenberg Marquardt algorithm is governed by the following equation:

$$w_{k+1} = w_k - (H + \alpha I)^{-1} g \tag{3.44}$$

The parameter α starts as a large value, thus, the algorithm behaves as a gradient descent algorithm. As the algorithm goes on, the parameter α decreases and as it approaches zero the algorithm converts to the Newton's method for the final convergence to the minimum (global or local) [39].

The Levenberg-Marquardt method described above stores the Hessian matrix in addition to getting the Hessian matrix's inverse. However, an approximation known as the "outer-product approximation" is applied. As such, the Levenberg-Marquardt method can be viewed as a first-order algorithm. Instead of fully computing the Hessian matrix, an approximation is computed to reduce the Levenberg-Marquardt computation complexity. Outer product approximation is used when the cost function is the mean-squared error function. The approximated Hessian matrix is computed using the average of the gradient vector's outer products as follows. Given that the desired (Target) signal

from the training data set is d and the actual network output is y, the error function E can be defined as:

$$E = \langle (d-y)^2 \rangle \tag{3.45}$$

Where $\langle . \rangle$ denotes the average for the entire training data set. Recall that the gradient g_j for the weight w_j is defined as:

$$g_j = \frac{\partial E}{\partial w_j} \tag{3.46}$$

Substituting equation (3.45) in (3.46):

$$g_j = \frac{\partial E}{\partial w_j} = 2 \langle (d-y) \frac{\partial y}{\partial w_j} \rangle \tag{3.47}$$

Recall that the Hessian matrix H is defined as $h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j}$. Substituting equation (3.47) in the previous equation then:

$$h_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} = 2 \langle -\frac{\partial y}{\partial w_i}\frac{\partial y}{\partial w_j} + (d-y)\frac{\partial^2 y}{\partial w_i \partial w_j} \rangle \tag{3.48}$$

Note that for obtaining the derivative of equation (3.48), the product rule is applied as follows:

$$(f.g)' = f'.g + f.g'$$

Equation (3.48) can be re-written in the following form:

$$H = 2(-P + Q) \tag{3.49}$$

Where the first term, $P = \langle \frac{\partial y}{\partial w_i} \frac{\partial y}{\partial w_j} \rangle = \langle g g^T \rangle$, embodies the first-order (gradient's) average outer product while the second term $Q = \langle (d - y) \frac{\partial^2 y}{\partial w_i \partial w_j} \rangle$ involves the second order terms. For all cases, the term P is real and symmetric. Accordingly, the matrix is positive definite. Such that:

$$\forall w \neq 0, \quad w^T P w \geq 0 \tag{3.50}$$

In order for the matrix P to have a full rank, the number of training samples must be more than the total number of network weights [171]. The approximation is based on the fact that the second order terms Q have a negligible value compared to the first-order ones P in the vicinity of the global minimum. Consequently, the second-order terms should be ignored on the condition that the residual errors $(d - y)$ have zero mean and be uncorrelated with second order derivatives.

3.5 The Extended Kalman Filter (EKF)

The Kalman filter (KF) is the most popular state estimation tool. It provides optimal estimates for linear systems in the presence of Gaussian white noise. In the case of nonlinear systems, the extended Kalman filter (EKF) is applied by linearizing the system around the latest state estimate at each time interval. An EKF-based neural network training technique was first introduced by Singhal and Wu in 1989 [47].

The EKF provides a powerful neural network training capability compared to conventional first-order gradient descent algorithms, such as the BP [40]. In literature,

the EKF has been extensively applied for training of both feed-forward [48] and recurrent networks [49,50] in both a global form (GEKF) or in a decoupled form (DEKF). Although the EKF demonstrates a close performance compared to a second-order derivative, batch-based optimization method, it can avoid local minima problems by encoding second-order information in terms of a state error covariance matrix [40]. Accordingly, the EKF represents an efficient and practical alternative to second-order training methods.

The EKF has been tailored for training feed-forward neural networks by formulating the network as a filtering problem [174]. Accordingly, feedforward multilayer perceptron network behaviour can be described by a nonlinear discrete-time state space representation [175] such that:

$$w_{k+1} = w_k + \omega_k \tag{3.51}$$
$$y_k = C_k(w_k, u_k) + v_k \tag{3.52}$$

Equation (3.51) represents the system equation. It demonstrates the neural network as a stationary system with an additional zero mean, white system noise ω_k with a covariance described by $[\omega_k \omega_l{}^T] = \delta_{k,l} Q_k$. Neural network weights and biases w_k are regarded as the system's state. Equation (3.52) is the measurement (observation) equation. It is a nonlinear equation relating the network's desired (target) response y_k to the network's input u_k and weights w_k. The nonlinear function C_k represents the

measurements function, it is a nonlinear function based on the sigmoidal function and the number of layers. A zero-mean, white measurement noise v_k is added with a covariance defined as $[v_k v_l{}^T] = \delta_{k,l} R_k$.

As previously mentioned, the KF provides an optimal state estimate for linear systems. However, for network training, the EKF must be used due to the nonlinearity of sigmoidal function φ.

Consider a feed-forward multi-layer perceptron network with two hidden layers as shown in Figure 3.9. All activation functions of the first, second and output layers are nonlinear sigmoidal functions denoted as φ_I, φ_{II} and φ_o respectively. The network transfer function in terms of network weights, inputs, and activation functions can be mathematically defined as:

$$y_i(k) = \varphi_o\left(w_{o_i}(k)\,\varphi_{II}\left(w_{II}(k)\,\varphi_I\left(w_I(k)u(k)\right)\right)\right) \quad \text{for } i=1,2..m \tag{3.53}$$

Where m denotes the number of output neurons, w_I, w_{II}, w_o are weight matrices for the first hidden layer, the second hidden layer and the output layer, respectively. where,

- $w_I \in \Re^{((z\times l_1)\times 1)}$ is the vector representing weights from the input to the first hidden layer
- $w_{II} \in \Re^{((l_1\times l_2)\times 1)}$ from first to second hidden layer
- $w_o \in \Re^{((l_2\times m)\times 1)}$ from second hidden layer to output layer

Where, l_1 and l_2 represent the number of hidden neurons in the first and the second hidden layers respectively, and z specifies the number of input neurons

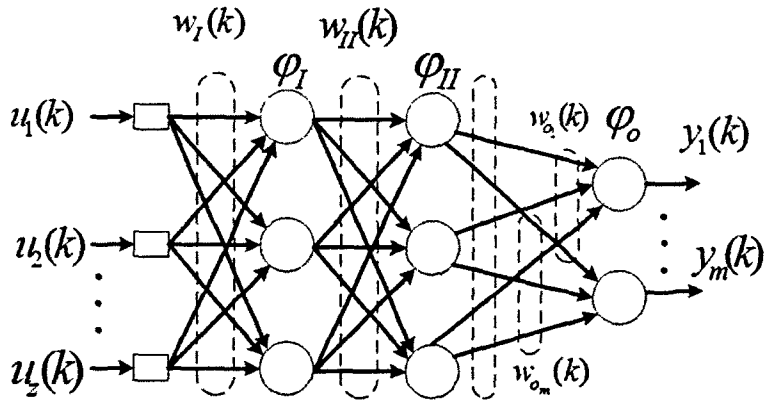

Figure 3.9. Feed-forward multilayer perceptron with 'z' inputs, 2 hidden layers and 'm' outputs

Linearization is performed by differentiating the network transfer function with respect to the network synaptic weights (i.e. deriving the Jacobian). The Jacobian matrix $C_{k|linearized}$ can be mathematically expressed as follows:

$$C_{k|linearized} = \begin{bmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} & \cdots & \frac{\partial y_1}{\partial w_{N_T}} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} & & \frac{\partial y_2}{\partial w_{N_T}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial y_m}{\partial w_1} & \frac{\partial y_m}{\partial w_2} & \cdots & \frac{\partial y_m}{\partial w_{N_T}} \end{bmatrix} \tag{3.54}$$

Where, N_T denotes the total number of synaptic weights (including bias). By differentiating Equation (3.53) with respect to the different weight groups w_I, w_{II}, w_o and for *i, l=1, 2 ... m* the following is obtained:

$$\frac{\partial y_i}{\partial w_{o_l}} = \begin{cases} \acute{\varphi}_o\left(w_{o_i}\varphi_{II}\left(w_{II}\varphi_I(w_I u)\right)\right)\ \varphi_{II}\left(w_{II}\varphi_I(w_I u)\right), & if\ i = l \\ 0, & otherwise \end{cases} \tag{3.55}$$

$$\frac{\partial y_i}{\partial w_{II}} = \acute{\varphi}_o\left(w_{o_i}\varphi_{II}\left(w_{II}\varphi_I(w_I u)\right)\right) w_{o_i}\acute{\varphi}_{II}\left(w_{II}\varphi_I(w_I u)\right)\varphi_I(w_I u) \tag{3.56}$$

$$\frac{\partial y_i}{\partial w_I} = \acute{\varphi}_o\left(w_{o_i}\varphi_{II}\left(w_{II}\varphi_I(w_I u)\right)\right) w_{o_i}\acute{\varphi}_{II}\left(w_{II}\varphi_I(w_I u)\right)w_{II}\acute{\varphi}_I(w_I u)u \tag{3.57}$$

By placing (3.49), (3.50), and (3.51) in one matrix:

$$C_{k|linearized} = [\frac{\partial y}{\partial w_o} \quad \frac{\partial y}{\partial w_I} \quad \frac{\partial y}{\partial w_{II}}] \tag{3.58}$$

$C_{k|linearized}$ is the *m-by-*N_T measurement matrix of the linearized model around the current weight estimate.

The EKF-based neural network training introduced by Singhal and Wu is known as the global extended Kalman filter (GEKF) [47]. In the GEKF algorithm, all network weights and biases are simultaneously processed and a second-order information matrix correlating each pair of network weights is obtained and updated [40]. Consequently,

the GEKF computational complexity is $O(m\,{N_T}^2)$. A storage capacity of $O({N_T}^2)$ is required, which is relatively high compared to the standard BP algorithm.

The DEKF-based neural network training algorithm illustrated below and represents the most general EKF-based neural network training method. The GEKF is a special form of the DEKF where the weight group number g is set to one. Weights that are directed to the same neuron are considered to belong to a specific weight group. Neural network training using the DEKF algorithm can be expressed as follows [176]:

$$\Gamma_k = [\sum_{i=1}^{g} (C_k^i)^T\, P_k^i\, C_k^i \; + R_k]^{-1} \tag{3.59}$$

$$K_{kalman_k}^{\;\;i} = P_k^i\; (C_k^i)^T\, \mathbf{\Gamma_k} \tag{3.60}$$

$$\alpha_k = d_k - \hat{d}_k \tag{3.61}$$

$$\widehat{w}_{k+1}^i = \widehat{w}_k^i + K_{kalman_k}^{\;\;i}\, \alpha_k \tag{3.62}$$

$$P_{k+1}^i = P_k^i - K_{kalman_k}^i\, C_k^i\, P_k^i + Q_k^i \tag{3.63}$$

Where, the following nomenclature applies:

$\mathbf{\Gamma} \in \Re^{(\mathrm{m}\times\mathrm{m})}$ is the global scaling matrix (or global conversion factor).

$\mathbf{C} \in \Re^{(\mathrm{m}\times n_i)}$ is the gradient matrix, it involves weights gradient w.r.t every output node.

$\alpha \in \Re^{(\mathrm{m}\times 1)}$ is the innovation which is the difference between desired (target) and actual network output.

$\mathrm{P} \in \Re^{(n_i\times n_i)}$ is the error covariance matrix.

$\mathrm{Q} \in \Re^{(n_i\times n_i)}$ is the process (system) noise covariance matrix.

$K_{kalman} \in \Re^{(n_i \times m)}$ is the Kalman gain matrix.

R $\in \Re^{(m \times m)}$ is the measurement noise covariance matrix.

The above DEKF training algorithm operates in a serial mode fashion. In the serial mode, one training sample is involved in the error calculation, gradients computation and synaptic weight update. A problem known as recency phenomenon arises with serial mode when training tends to be influenced by the most recent samples [40]. Consequently, a trained network fails to remember former input-output mappings and thus serial-mode training results in deteriorated performance. The recency phenomenon can be circumvented by using the multi-streaming training technique [177,178,179].

In the multi-streaming EKF training, multiple training samples are batched and processed. It involves training M identical neural networks using several training samples followed by weight update using the overall networks' errors. The algorithm of equations (3.59) to (3.63) can be applied to multi-streaming mode by replacing the matrix dimension m in Γ, C, α, K_{Kalman}, and R with $M \times m$ [46].

Chapter 4: Neural Networks Training Using the Smooth Variable Structure Filter

In this chapter, a new proposed neural networks training technique is proposed. The technique is based on a relatively new filtration strategy known as the Smooth Variable Structure Filter (SVSF). The SVSF is a state and parameter estimation method that works in a predictor-corrector fashion and provides guaranteed stability and robust dynamic adaptation to modeling uncertainties. The SVSF has been successfully applied to train feedforward multilayer perceptron (MLP) networks and provides an excellent generalization capability and minimum mean squared error compared to other training techniques.

This Chapter is divided into two sections, namely, Section 4.1 provides a description of the SVSF and discusses its derivation. Section 4.2 discusses the formulation of the SVSF for the training of feedforward multilayer perceptron networks. The algorithm is used later in Chapter 5 to train feed-forward multilayer perceptron network on benchmark problems.

4.1 The SVSF Concepts and Applications

State and Parameter estimation methods have been implemented in numerous engineering applications including those related to fault detection and diagnosis. These methods can detect if faults occur in a system and identify their source. In their application to fault detection and diagnosis, they are classified as model-based methods, and operate as follows. A physical model of the system is obtained used in conjunction with the filter for tracking model parameters given measurements from the system. Consequently, the filter can therefore identify deviations in system parameters and departures from normal operating conditions.

As stated in chapter two, common parameter estimation methods such as the Kalman filter, the Unscented Kalman filter, and the Particle filter have been extensively used for fault detection and diagnosis applications. The SVSF is a relatively new filter that can guarantee stability given bounded uncertainties. One of its important advantages is that the SVSF provides an extra performance indicator besides the estimation error and the error covariance matrix.

In this section, the SVSF theory and stability guarantee criteria are discussed. The SVSF gain calculation and the smoothing boundary layer introduction to overcome the chattering problem are provided, and finally, the SVSF algorithm for parameter and state estimation is described.

4.1.1 The SVSF theory

The SVSF was initially inspired by the variable structure control (VSC) and systems theory where the state space of a function $f(x,u)$ is separated by a discontinuous hyperplane [180]. For instance, assume a control signal $u_i(x,t)$ which is a function of the state vector x. By assuming a discontinuity surface $S_i(x)$ that separates two continuous regions, a control signal $u_i(x,t)$ can be defined as follows:

$$u_i(x,t) = \begin{cases} u_i^+(x,t) & when \quad S_i(x) > 0 \\ u_i^-(x,t) & when \quad S_i(x) < 0 \end{cases} \tag{4.1}$$

Where in each segment of the state space, the function $f(x,u)$ is continuous and differentiable. By forcing system states to be directed to the hyper plane and slide across it, a category of VSC widely known as the sliding mode control (SMC) is obtained. The SMC uses a discontinuous control signal to remain on the sliding surface and while on this surface achieves robustness to disturbances and modeling uncertainties.

The Variable Structure Filter (VSF) is a filtration strategy that is an extension to the sliding mode concept [181]. In the VSF, as shown in Figure 4.1, an estimate of the state trajectory is generated and is forced towards the actual state trajectory. The estimates then remain within a neighbourhood of the actual state trajectory, known as the existence subspace. The width of the existence subspace is a function of uncertainty in the filter model and varies with time. In the existence subspace, a switching gain is

applied so that the state estimates switch back and forth along the true (desired) state trajectory. The period for the states to reach the existence subspace is known as the reachability phase [181].

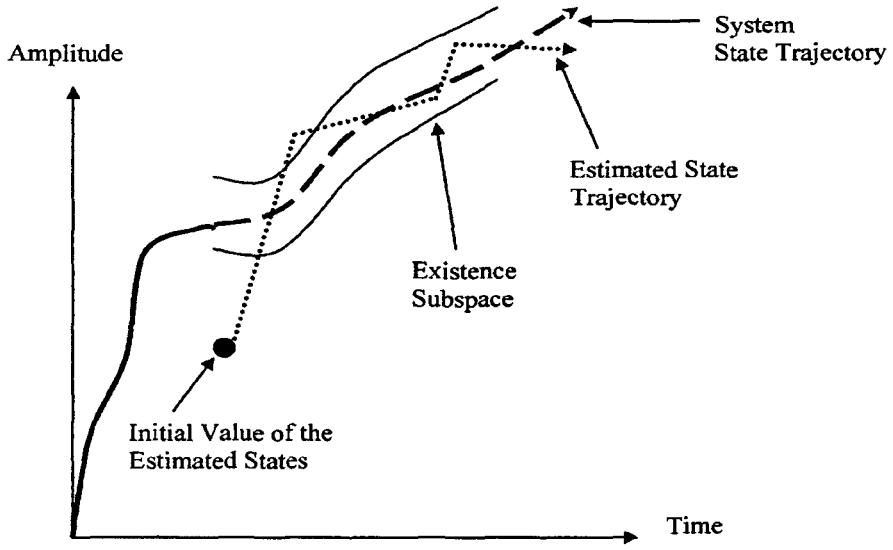

Figure 4.1. SVSF State Estimation

Based on the degree of uncertainties, the thickness of the existence subspace is determined. Consequently, for bounded uncertainties, a bounded existence subspace can be assumed and the state estimates are restricted to chatter within the existence subspace along the true state trajectory.

The SVSF can be applied to both linear and non-linear dynamic systems. Two versions of the SVSF have been developed in

the literature; an original form of the SVSF without involving the state error covariance matrix $P_{k+1|k}$ and one that includes the covariance matrix [182]. In this research project, the original form of the SVSF will be discussed and applied to train feed-forward MLP networks.

Consider the following nonlinear system where f is a nonlinear function and H is the linearized output matrix of the following form:

$$x_{k+1} = f(x_k, u_k, w_k) \tag{4.2}$$
$$z_k = Hx_k + v_k \tag{4.3}$$

Where x_k are the system's states, u_k is the system's input, w_k is the system noise, v_k is the measurement noise, and z_k is the measured output. Recall that the EKF assumes that the system and measurement noise are white Gaussian noise with covariance matrices R_k and Q_k.

The SVSF algorithm is analogous to the Kalman filter algorithm stated earlier in Chapter three. First, using the system model, an a-priori state estimate $\hat{x}_{k+1|k}$ is calculated as follows:

$$\hat{x}_{k+1|k} = \hat{f}(\hat{x}_{k|k}, u_k) \tag{4.4}$$

Secondly, a discontinuous switching corrective term K_k is calculated and used to force the state estimates to be directed towards the actual (correct) state trajectory.

This is achieved by updating the a-priori state estimate and generating an a-posteriori state estimate $\hat{x}_{k+1|k+1}$ as follows:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_k \tag{4.5}$$

In Figure 4.2, by applying the corrective SVSF gain $\mathrm{K_k}$, the path of the estimated state is reversed whenever it crosses the actual state trajectory.

As such the actual state trajectory can be considered as the switching hyperplane as shown in Figure 4.2, the estimation error is decreased and the state estimate approaches the true state trajectory.

$e_{z_{k|k}}$ and $e_{z_{k|k-1}}$ are the a-posteriori and a-priori measurement errors, respectively. They are defined as:

$$e_{z_{k|k}} = z_k - \hat{z}_{k|k} \tag{4.6}$$
$$\mathrm{e_{z_{k|k-1}}} = \mathrm{z_k} - \mathrm{\hat{z}_{k|k-1}} \tag{4.7}$$

The SVSF's stability can be proven by using a discrete Lyapunov function, the Lyapunov function is used to validate stability of a given dynamic system. Let the discrete Lyapunov function be defined as:

$$V_k = e_{z_{k|k}}{}^{\circ} e_{z_{k|k}} \tag{4.8}$$

The estimation process is stable if:

$$\Delta V_k = e_{z_{k|k}} \circ e_{z_{k|k}} - e_{z_{k-1|k-1}} \circ e_{z_{k-1|k-1}} < 0 \qquad (4.9)$$

Thus, the SVSF corrective term K_k can be derived as follows. Re-arranging equation (4.9) then the condition of stability can be restated as:

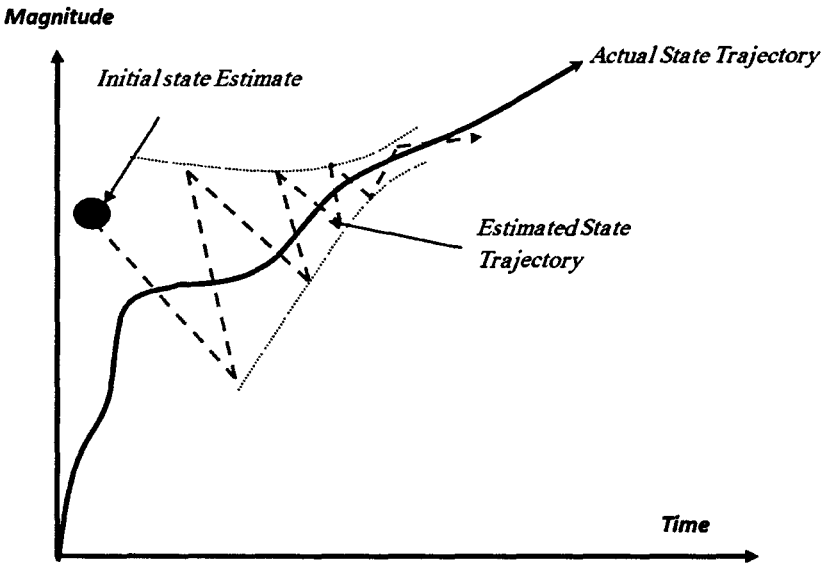

Figure 4.2. SVSF State Estimation Concept

$$\left|e_{z_{k|k}}\right| < \left|e_{z_{k-1|k-1}}\right| \qquad (4.10)$$

Given the condition in equation (4.10), let the corrective gain be defined as:

$$K_k = \widehat{H}^+ \left| \left| e_{z_{k|k-1}} \right|_{ABS} + \gamma \left| e_{z_{k-1|k-1}} \right|_{ABS} \right|_{ABS} \circ sgn(e_{z_{k|k-1}}) \tag{4.11}$$

Where $\gamma \in R^{mxm}$ is a diagonal matrix with elements within the range of 0 to 1. From equation (4.11),

$$\left| \widehat{H} K_k \right|_{ABS} = \left| e_{z_{k|k-1}} \right|_{ABS} + \gamma \left| e_{z_{k-1|k-1}} \right|_{ABS} \tag{4.12}$$

Given $\gamma's$ definition, then:

$$\left| \widehat{H} K_k \right|_{ABS} < \left| e_{z_{k|k-1}} \right|_{ABS} + \left| e_{z_{k-1|k-1}} \right|_{ABS} \tag{4.13}$$

Then,

$$\left| \widehat{H} K_k \right|_{ABS} - \left| e_{z_{k|k-1}} \right|_{ABS} < \left| e_{z_{k-1|k-1}} \right|_{ABS} \tag{4.14}$$

Since from (4.11):

$$sgn\left(\widehat{H} K_k\right) = sgn(e_{z_{k|k-1}}) \tag{4.15}$$

Then from equation (4.13):

$$\left| e_{z_{k|k-1}} - \widehat{H} K_k \right|_{ABS} < \left| e_{z_{k-1|k-1}} \right|_{ABS} \tag{4.16}$$

From equation (4.3), (4.5), (4.7) get,

$$e_{z_{k|k}} = e_{z_{k|k-1}} - \widehat{H} K_k \tag{4.17}$$

Substituting equation (4.17) into (4.16) then,

$$\left|e_{z_{k|k}}\right|_{ABS} < \left|e_{z_{k-1|k-1}}\right|_{ABS}$$

Consequently, the stability condition is satisfied and the corrective SVSF gain from equation (4.11) results in stability.

Substituting from equations (4.3), (4.6), and (4.7), then equation (4.10) can be expanded to:

$$\left|He_{x_{k|k}} - v_k\right|_{ABS} < \left|He_{x_{k-1|k-1}} - v_{k-1}\right|_{ABS} \tag{4.18}$$

Then, for a completely observable and controllable system, if the output matrix H is a positive, diagonal matrix, and v_k is white noise then:

$$E\left[\,|e_{x_{k|k}}|_{ABS}\right] < E[|e_{x_{k-1|k-1}}|_{ABS}] \tag{4.19}$$

Where $\boldsymbol{E}[.]$ is the expectation.

Consequently, as the estimation algorithm proceeds, the expectation of the estimation error decreases. In other words, since the output estimates will eventually converge, this will lead to the convergence of the state estimates too.

Due to the application of a *sgn(.)* function during the SVSF gain calculation, a discontinuous, high frequency, chattering arises in the state estimates. Chattering can be reduced or eliminated by introducing a smoothing function with a boundary layer thickness Ψ. The smoothing function thickness can be tuned during the estimation process while the existence subspace β width is unknown and a function of uncertain

dynamics related to modeling uncertainties. The smoothing function Ψ, as shown in Figure 4.3 and Figure 4.4, can remove the effect of chattering if its width is greater than the existence subspace β. It will otherwise fail to do so. To elaborate further, consider the following two cases.

Case 1: ψ larger than the existence subspace β

Consider Figure 4.3 where the smoothing subspace ψ is selected to be larger than the existence subspace β. In this case, after the state estimates are forced to the existence subspace, they remain within the smoothing boundary layer and the chattering effect is removed by applying a continuous corrective action through a saturation term *sat(.)* as follows:

$$K_{k+1} = H^{-1}\left(\left|e_{z,k+1|k}\right| + \gamma\left|e_{z,k|k}\right|\right) \circ sat\left(e_{z,k+1|k}, \psi\right) \tag{4.20}$$

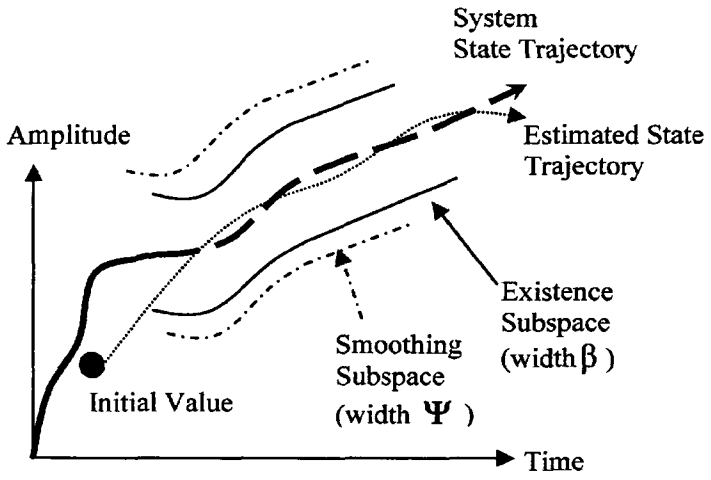

Figure 4.3. Smoothing Boundary Layer [183]

Where the saturation function $sat(.)$ is defined as follows:

$$sat\left(e_{z,k+1|k},\psi\right)=\begin{cases}1, & e_{z,k+1|k}/\psi \geq 1\\ e_{z,k+1|k}/\psi, & -1 < e_{z,k+1|k}/\psi < 1\\ -1, & e_{z,k+1|k}/\psi \leq -1\end{cases} \tag{4.21}$$

Case 2: ψ smaller than the existence subspace β

As shown in Figure 4.4, if the smoothing layer is smaller than the existence subspace, the estimates will exit the smoothing boundary layer and the saturation function effectively reverts to a sign function. The corrective action becomes discontinuous and chattering occurs indicating that the upper boundary of uncertainties has been incorrectly underestimated.

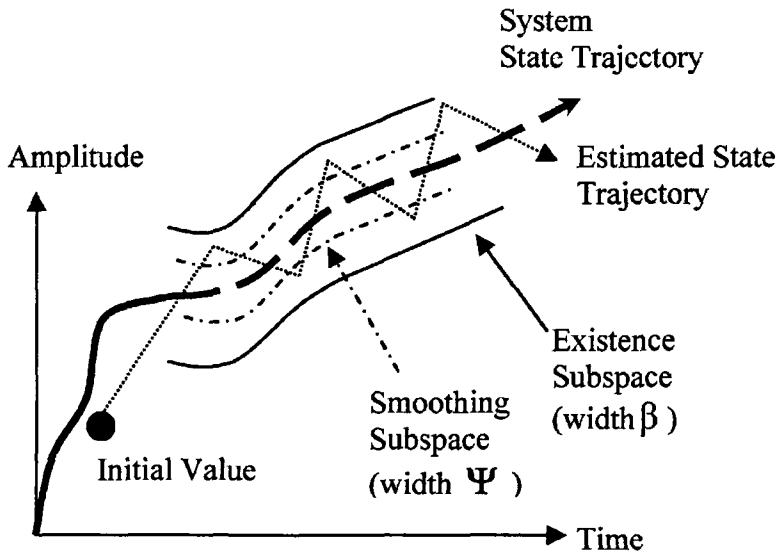

Figure 4.4. Chattering Concept Illustration [183]

4.2 The SVSF algorithm

The SVSF algorithm has two forms, an original algorithm as stated in [181] that does not include a covariance derivation and a revised from that includes a state error covariance matrix $P_{k+1|k}$ as discussed in [184]. In the following subsection, the original form is presented first followed by the revised form.

4.2.1 The original SVSF algorithm

According to the previous section, if the SVSF gain satisfies the stability condition of $\left|e_{z_{k|k}}\right| > \left|e_{z\ k+1|k+1}\right|$, the SVSF algorithm is stable and provides convergence. The SVSF algorithm can be summarized as follows, assuming a non-linear dynamic system with state and measurement equations defined as follows:

$$x_{k+1|k} = f(x_{k|k}, u_k, w_k) \tag{4.22}$$
$$z_{k+1|k} = Hx_{k+1|k} + v_k \tag{4.23}$$

Where f is a non-linear function and H is the measurement matrix. For linear systems, the system equation is defined as follows:

$$x_{k+1|k} = Fx_{k|k} + Gu_k + w_k \tag{4.24}$$

Where F, G are the state and input matrices respectively.

The SVSF process is as follows,

1. First step involves calculation of an a-priori state estimates $\hat{x}_{k+1|k}$

$$\hat{x}_{k+1|k} = F\hat{x}_{k|k} + Gu_k \tag{4.25}$$

Followed by the predicted measurement $\hat{z}_{k+1|k}$,

$$\hat{z}_{k+1|k} = H\hat{x}_{k+1|k} \tag{4.26}$$

The measurement error $e_{z,k+1|k}$ can be calculated as the difference between the actual measurements and its predicted value:

$$e_{z,k+1|k} = z_{k+1} - \hat{z}_{k+1|k} \tag{4.27}$$

2. Next comes the calculation of the SVSF gain as a function of the a-priori measurement error $e_{z_{k+1|k}}$, the a-posteriori measurement error $e_{z_{k|k}}$, the convergence rate γ, and the smoothing boundary layer thickness ψ such that:

$$K_{k+1} = H^{-1}\left(\left|e_{z,k+1|k}\right| + \gamma\left|e_{z,k|k}\right|\right) \circ sat\left(e_{z,k+1|k}, \psi\right) \tag{4.28}$$

3. The a-posteriori state estimate $\hat{x}_{k+1|k+1}$ is then calculated as follows:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1} \tag{4.29}$$

4. Finally, the new updated measurement estimates are used for the next iteration:

$$\hat{z}_{k+1|k+1} = H\hat{x}_{k+1|k+1} \tag{4.30}$$
$$e_{z_{k+1|k+1}} = z_{k+1} - \hat{z}_{k+1|k+1} \tag{4.31}$$

4.2.2 Revised SVSF form

By including the state error covariance matrix $P_{k+1|k}$, and using the same dynamic system described in equations (4.23), (4.24), the revised SVSF algorithm is as follows [184]. In addition to calculating the predicted state estimates $\hat{x}_{k+1|k}$ as discussed under step 1 in the previous subsection, the state error covariance matrix $P_{k+1|k}$ is also calculated as follows:

$$P_{k+1|k} = HP_{k|k}H^T + Q_k \tag{4.32}$$

The SVSF gain is revised in step 2 as:

$$K'_{k+1} = H^+ diag\left[\left(\left|e_{z_{k+1|k}}\right| + \gamma\left|e_{z_{k|k}}\right|\right) \circ sat\left(\bar{\psi}^{-1}e_{z_{k+1|k}}\right)\right] diag\left(e_{z_{k+1|k}}\right)^{-1} \tag{4.33}$$

Where, ψ is a diagonal matrix representing boundary layer thicknesses for m number of measurements.

$$\bar{\psi}^{-1} = \begin{bmatrix} \frac{1}{\psi_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{\psi_m} \end{bmatrix} \tag{4.34}$$

The state estimate update equation of step 3 is altered as:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K'_{k+1}e_{z,k+1|k} \tag{4.35}$$

Note that the revised gain of equation (4.33) does not alter the a-priori estimate and that,

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K'_{k+1} e_{z,k+1|k} = \hat{x}_{k+1|k} + K_{k+1} \tag{4.36}$$

As such the stability proof of the SVSF is not altered and the revised form only allows an iterative process for calculation the error covariance matrix. The covariance matrix is updated under step 4 as:

$$P_{k+1|k+1} = (I - K'_{k+1}H)P_{k+1|k}(I - K'_{k+1}H)^T + K'_{k+1}R_{k+1}K'^T_{k+1} \tag{4.37}$$

The revised SVSF algorithm is iteratively repeated similar to the original SVSF discussed in section 1.2.1. In the following, the SVSF is tailored to train feed-forward multilayer perceptron networks. The original algorithm is used in a multi-streaming global mode.

4.3 The SVSF-based Neural Network Training

The SVSF can be applied for training nonlinear feed-forward neural networks through estimation of the network weights. In this section, the formulation of the SVSF-based feedforward MLP training algorithm is presented.

4.3.1 The SVSF Training Algorithm

The SVSF can be applied for training nonlinear feed-forward neural networks by estimating network weights. In the same fashion as the Kalman filter, the SVSF has been

adapted to train feed-forward neural networks by visualizing the network as a filtering problem where F, G, and C_k are the system, input, and output matrices, respectively as follows:

$$\widehat{w}_{k+1|k} = F\widehat{w}_{k|k} + Gu_k \tag{4.38}$$

$$y_k = C_k(w_{k|k}, u_k) \tag{4.39}$$

Where $F = I,\ G = 0,\ and\ C_k$ is similar to the Extended Kalman filter case discussed earlier. The global SVSF training algorithm is iterative and is summarized by the following steps, assuming a training data set $\{x_k, z_k\}$.

Step 1: Network weights initialization

The a-priori state estimates or network weights, $\widehat{w}_{k|k}$, are randomly initialized ranging from -1, 1.

Step 2: Calculation of the predicted (a-priori) weight estimates $\widehat{w}_{k+1|k}$ from (4.38)

For neural networks training, the system matrix F is an identity matrix and G matrix is set to zero. Consequently, when the algorithm is initialized, the a-priori weight matrix is the same as the a-posteriori from the previous step as follows:

$$\widehat{w}_{k+1|k} = \widehat{w}_{k|k} \tag{4.40}$$

Step 3: Jacobian Matrix calculation (Linearization) of the measurement matrix C_k

The algorithm for the Jacobian matrix calculation is as follows, assuming the use of nonlinear sigmoidal activation functions denoted as φ_{I}, φ_{II} and $\varphi_{\mathrm{o}}$for the first, second and output layers respectively shown in Figure 4.5, the network transfer function can be mathematically defined as:

$$y_i(k) = \varphi_o\left(w_{o_i}(k)\,\varphi_{II}\left(w_{II}(k)\,\varphi_I\left(w_I(k)u(k)\right)\right)\right) \qquad (4.41)$$

for i=1,2..m

Where m denotes the number of output neurons, w_I, w_{II}, and $w_o$are the group weight matrices for the first hidden layer, the second hidden layer and the output layer, respectively. Linearization is performed by differentiating the network transfer function with respect to network synaptic weights (i.e., deriving the Jacobian). The Jacobian matrix $C_{k|linearized}$ can be mathematically expressed as follows:

$$C_{k|linearized} = \begin{bmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} & \cdots & \frac{\partial y_1}{\partial w_{N_T}} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} & & \frac{\partial y_2}{\partial w_{N_T}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial y_m}{\partial w_1} & \frac{\partial y_m}{\partial w_2} & \cdots & \frac{\partial y_m}{\partial w_{N_T}} \end{bmatrix} \qquad (4.42)$$

Where N_T denotes the total number of synaptic weights (including bias) and z specifies number of input neurons. By differentiating (4.41) with respect to different weight groups w_I, w_{II}, w_o and *for i, l=1, 2 ... m* the following is obtained:

$$\frac{\partial y_i}{\partial w_{o_l}} = \begin{cases} \acute{\varphi}_o\left(w_{o_i}\varphi_{II}\left(w_{II}\varphi_I(w_I u)\right)\right)\varphi_{II}\left(w_{II}\varphi_I(w_I u)\right), & if\ i = l \\ 0, & otherwise \end{cases} \tag{4.43}$$

$$\frac{\partial y_i}{\partial w_{II}} = \acute{\varphi}_o\left(w_{o_i}\varphi_{II}\left(w_{II}\varphi_I(w_I u)\right)\right)w_{o_i}\acute{\varphi}_{II}\left(w_{II}\varphi_I(w_I u)\right)\varphi_I(w_I u) \tag{4.44}$$

$$\frac{\partial y_i}{\partial w_I} = \acute{\varphi}_o\left(w_{o_i}\varphi_{II}\left(w_{II}\varphi_I(w_I u)\right)\right)w_{o_i}\acute{\varphi}_{II}\left(w_{II}\varphi_I(w_I u)\right)w_{II}\acute{\varphi}_I(w_I u)u \tag{4.45}$$

By placing (4.43), (4.44), and (4.45) in a matrix form, then:

$$C_{k|linearized} = \left[\frac{\partial y}{\partial w_o} \quad \frac{\partial y}{\partial w_I} \quad \frac{\partial y}{\partial w_{II}}\right] \tag{4.46}$$

$C_{k|linearized}$ is the *m*-by-N_T matrix of the linearized model around the current weight estimate. After linearization, the measurement equation is in the following form:

$$z_{k+1|k} = C_{k|linearized}w_{k+1|k} \tag{4.47}$$

Thus by multiplying the linearized Jacobian measurement matrix $C_{k|linearized}$ by the a-priori network weights $\hat{w}_{k+1|k}$ then network output vector prediction $\hat{z}_{k+1|k}$ is

generated. In this SVSF formulation, the states are treated as the weights of the system and the output vector is a linearly related to the states by the matrix $C_{k|linearized}$.

Step 4: Perform state transformation

State transformation is applied so that the output vector is related to the states using an identity matrix I. A new state vector $\hat{x}_{k+1|k}$is formulated and used instead of the old state vector $\hat{w}_{k+1|k}$ by replacing the term $C_{k|linearized}\hat{w}_{k+1|k}$with $\hat{x}_{k+1|k}$ as follows:

$$\hat{x}_{k+1|k} = C_{k|linearized}\hat{w}_{k+1|k} \tag{4.48}$$

Note that the system and output equations in (4.40) and (4.47) after the previous step becomes

$$x_{k+1|k} = x_{k|k} \tag{4.49}$$

$$z_{k+1|k} = Ix_{k+1|k} \tag{4.50}$$

Step 5: Network's a-priori output (measurements) $\hat{z}_{k+1|k}$ calculation

$$\hat{z}_{k+1|k} = I\hat{x}_{k+1|k} \tag{4.51}$$

Step 6: Measurement error $e_{z_{k+1|k}}$ calculation

Using the output $\hat{z}_{k+1|k}$ and the corresponding target (from the neural network training data set) z, measurement errors $e_{z_{k+1|k}}$ can be calculated as follows:

$$e_{z_{k+1|k}} = z - \hat{z}_{k+1|k} \tag{4.52}$$

Step 7: SVSF gain calculation

The SVSF gain is a function of the a priori and the a posteriori measurement errors $e_{z_{k+1|k}}$ and $e_{z_{k|k}}$, the smoothing boundary layer widths ψ, the SVSF memory or convergence rate γ, as well as the linear measurement matrix $C_{k|linearized}$. The SVSF gain is defined as a diagonal matrix such that:

$$K_{k+1} = diag\left[\left(\left|e_{z_{k+1|k}}\right| + \gamma\left|e_{z_{k|k}}\right|\right) \circ sat\left(\frac{e_{z_{k+1|k}}}{\psi}\right)\right] diag\left(e_{z_{k+1|k}}\right)^{-1} \tag{4.53}$$

Step 8: Weight update

The a-posteriori weights $\hat{x}_{k+1|k+1}$ are obtained as:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}e_{z_{k+1|k}} \tag{4.54}$$

Step 9: The a-posteriori output estimate and error calculation

$\hat{z}_{k+1|k+1}$ and measurement errors $e_{z_{k+1|k+1}}$ are calculated to be used for the next iteration:

$$\hat{z}_{k+1|k+1} = I\hat{x}_{k+1|k} \tag{4.55}$$

$$e_{z_{k+1|k+1}} = z - \hat{z}_{k+1|k+1} \tag{4.56}$$

Step 10: The actual weights (states) calculation

Get the actual weight estimates (states) $\hat{w}_{k+1|k}$ from the states $\hat{x}_{k+1|k}$ as follows:

$$\widehat{w}_{k+1|k+1} = C^{+}_{k|linearized}\widehat{x}_{k+1|k+1} \tag{4.57}$$

Steps 3 to 10 are iteratively repeated while shuffling (randomly shifting) the training data set each epoch. Shuffling process involves training the network by presenting training data sets in a random order each epoch. Shuffling aims at enhancing performance by allowing training process to be unbiased. For back propagation algorithm discussed earlier, shuffling helps in removing undesirable effects such as convergence to local minimum or oscillations that might occur during training.

Training proceeds until one of the stopping conditions is reached as previously discussed in chapter 3.

4.4 Pseudo-inverse instability

Numerous authors have experienced abrupt and unexpected instabilities with the pseudoinverse [185,186]. A sudden growth of the Jacobian matrix elements when calculating the pseudoinverse during the SVSF gain calculation occurs at each epoch, as in (4.57). Consequently, the network's outputs and thus the mean squared error between the targets and outputs increase significantly. A stabilizing adjustment is performed to avoid this problem.

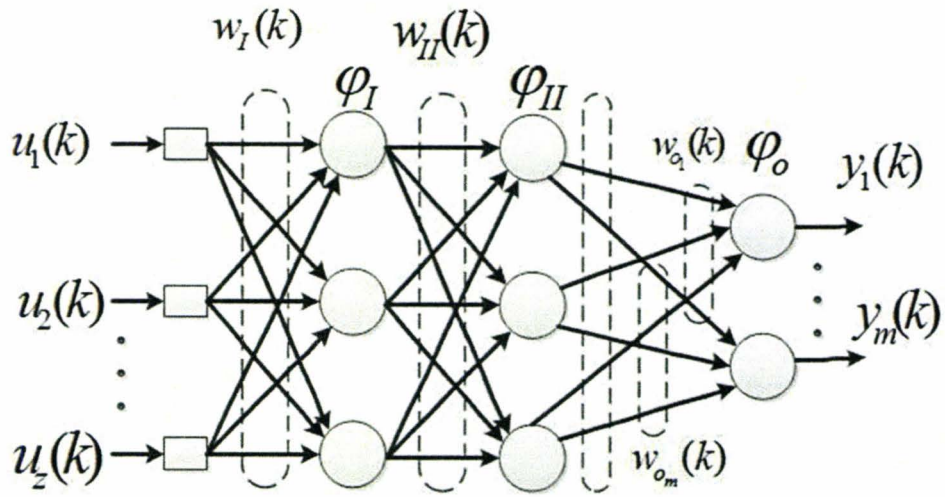

Figure 4.5. Feed-Forward Multilayer Perceptron with 'z' Inputs, 2 Hidden Layers and 'm' Outputs

The problem has been extensively analyzed in [185], and occurs due to the presence of singularities. Singularities occur when the Jacobian matrix loses rank. Small singular values of H might arise in the vicinity of these singularities. Consequently, larger values ensue when obtaining the pseudoinverse of the Jacobian H^+ thus creating larger error values which leads to instability. According to [187], it is rather difficult to detect these singularities. A traditional method of solving this instability problem is to replace the pseudoinverse H^+ with the following equation:

$$H_d^+ = H^T(HH^T + \rho^2 I)^{-1} \tag{4.58}$$

Where, ρ is called the damping parameter. The effect of the added damping is that it mitigates the effect of small singular values when computing the inverse. Its disadvantage is that a small error is introduced when calculating the inverse.

Chapter 5 Application of the SVSF Training Method to Benchmark Problems

Introduction

In this chapter, the application of the SVSF-based training method on a benchmark problem is discussed. The benchmark problem is known as PROBEN1 and it is widely applied in the literature to assess the performance of various training techniques. The SVSF-based training method is compared to classical training techniques such as the first order back propagation (BP) algorithm, the Levenberg-Marquardt (LM), Quasi-Newton (QN), and the extended Kalman Filter (EKF) using these benchmark problems. This chapter is divided to three sections. Section 5.1 involves the description of the three benchmark problems under consideration. Section 5.2 provides benchmarking rules followed during training. Section 5.3 includes simulation results of the SVSF-based training algorithm in addition to other previously mentioned classical training techniques.

5.1 Benchmark classification problem (PROBEN1)

PROBEN1 is a standard set of classification oriented problems, conventions, and guidelines that are used in assessing algorithms [165]. PROBEN1 is a collection of 15 data sets form 12 diverse fields that represent real world data in both continuous and binary

values. In addition, PROBEN1 includes a set of rules on how to conduct a benchmark neural network training test to allow easy comparisons of training algorithms. PROBEN1 has overcome various deficiencies related to previously published scientific projects in the field of artificial neural networks as stated in [165] as follows:

(1) Most of the problems used were artificial problems like XOR and n-parity. They have solid a-priori consistencies among training samples which make it unfair to test training algorithm generalization competences.

(2) There was barely enough statistical assessment of the simulation results.

(3) Information about various parameters and network topologies were not adequately communicated to enable other researchers to reproduce the results.

PROBEN1 consists of training data sets from the University of California, Irvine (UCI) learning database archive [165][1]. PROBEN1 provides three different versions of the same data set according to the order of the training samples within the dataset. For example, three versions of data set for Cancer diagnosis are available namely, *Cancer1, Cancer2,* and *Cancer3* that differs in the ordering of dataset. In this paper, the datasets are used for neural network training by splitting them in parts into 50%, 25%, 25% segments representing training, validation, and testing sets, respectively. In this research, a fully connected feed forward multilayer perceptron with two hidden layers is used. The network topology varies depending on the classification problem. Networks were

[1] Data is available for download through: (ics.ci.edu, directory /pub/machine-learning-database)

trained using the SVSF, EKF, the batch mode first-order BP, and the second order BP algorithms namely, Levenberg-Marquardt (LM) and Quasi-Newton (QN).

PROBEN1 includes 15 classification problems. Of these, three classification problems with different training difficulty levels are selected. They include Cancer, diabetes, and glass problems.

5.1.1 Cancer Problem

Data sets were originally obtained by Dr. W. H. Wolberg, University of Wisconsin Hospitals, Madison [188]. This problem represents classification of breast tumour to either malignant or benign. 699 training data sets, with nine inputs and two outputs each, are collected using microscopic investigations. Inputs represent attributes used for cancer classification includes: 1-Clump Thickness, 2-Uniformity of Cell Size, 3-Uniformity of Cell Shape, 4-Marginal Adhesion, 5-Single Epithelial Cell Size, 6-Bare Nuclei, 7-Bland Chromatic, 8-Normal Nucleoli, and 9-Mitoses. Outputs represent 0 or 1 depending on the classification of cancer to either malignant or benign.

Three forms of the same problem namely cancer1, cancer2, cancer3 are presented according to the order of data set. Cancer1 only has been implemented in this research. Datasets are split into three segments for training (50%), validation (25%), and testing (25%). Table 5.1 shows different class distributions and percentages. There are 16

missing values for attribute (input) 6 that are replaced by a constant value of 0.3 instead for network training.

Table 5.1 Cancer Problem Class Distribution

	Output Class		
Distribution	**Benign**	**Malignant**	**Total**
Total #	458	241	699
Total %	66	34	100

5.1.2 Diabetes Problem

This data set is from the "Pima Indians diabetes" folder from the UCI database archive. The problem tackles classification of "Pima Indian" individuals to either diabetes positive or not. The Datasets consist of 768 examples with eight inputs and two outputs. Inputs represent eight personal and experimental data as follows: 1-Number of times pregnant, 2-Plasma glucose concentration in 2 hours in an oral glucose tolerance test, 3-Diastolic blood pressure, 4-Triceps skin fold thickness (mm), 5- 2-Hour serum insulin (mu U/ml), 6-Body mass index (weight in kg/(height in m)^2), 7-Diabetes pedigree function, 8-Age (years). Two binary outputs provide classification of Pima Indian individuals to either diabetes negative (0) or diabetes positive (1). 65.1% of the examples are diabetes negative. Table 5.2 shows diabetes dataset distribution and percentages of samples.

Table 5.2. Diabetes Problem Class Distribution

	Output Class		
Distribution	**Diabetes**	**No diabetes**	**Total**
Total #	268	500	768
Total %	34	66	100

5.1.3 Glass Problem

The third dataset is fetched from the "glass" problem in UCI database archive. This problem tackles classification of glass to one of six categories. Glass data set consists of nine inputs: one representing glass refractive index (1) and the remaining eight inputs represent percentage content of eight glass splinter chemical elements namely, Na-sodium (2), Mg-magnesium (3), Al-aluminum (4), Si-silicon (5), K-potassium (6), Ca-calcium (7), Ba-barium (8), and Fe-iron (9).

Glass Samples are classified to one of the following six outputs namely, float processed (1), non-float processed building windows (2), vehicle windows (3), containers (4), tableware (5), and head lamps (6). This benchmark problem is used for forensic criminal investigations.

214 datasets are used in this test. All inputs are continuous. This benchmark problem is quite challenging as fewer number of datasets are available compared to cancer and diabetes problems. Hence, it tests the sensitivity of training algorithm to discard (waste) information. Table 5.3 shows glass distribution of samples.

Table 5.3. Glass Problem Class Distribution

	Output Class						
Distribution	**1**	**2**	**3**	**4**	**5**	**6**	**Total**
Total #	70	76	17	13	9	29	214
Total %	32.7	35.5	7.9	6.1	4.2	13.6	100

The three benchmark problems previously stated characterize different degrees of classification difficulties. Cancer provides adequate number of training samples. Classes are partially overlapped with complex boundaries amongst them. Hence, it is a fairly easy classification task [167]. Diabetes represents higher level of difficulty by overlapping classes as well as complex boundaries. Glass increases the level of difficulty by providing few training data sets in addition to complex boundaries and overlapping classes. Table 5.4 summarizes the training data attributes. For each benchmark problem,

it shows number of inputs, input attributes, output classes, number of training data sets, and the entropy.

Input attributes are all continuous. Output attributes are all binary for all problems under investigation. The entropy 'E' is a measure of random variable uncertainty or unpredictability. As entropy increases, the level of unpredictability increases too. Entropy for class probabilities $P(c)$ is defined as follows:

$$E = \sum_{Classes\ c} P(c)\, log_2(P(c)) \tag{5.1}$$

As shown in Table 5.4, cancer and diabetes are relatively predictable. Glass is highly unpredictable thus it is much harder problem.

Table 5.4 Attribute Structures of Classification Problems

Problem	Input values	Output Classes	Training set #	*E*
Cancer	9	2	699	0.93
Diabetes	8	2	768	0.93
Glass	9	6	214	2.18

5.2 Benchmarking Rules

In this research, benchmarking rules stated in [165] are followed. The training algorithms used are briefly discussed in Chapter 3. These are the first order, gradient-based batch back propagation algorithm [189], the Levenberg Marquardt algorithm [190,191], the Quasi-Newton technique [192], and the extended Kalman filter [178,47]. The measurement noise covariance matrix R elements are set to 0.1 such that $R = 0.1 * I_{mxm}$. The process noise covariance matrix Q elements are set to 0.01 such that $Q = 0.01 * I_{nxn}$. The initial error covariance matrix $P_{0|0}$ elements are set to $0.1 \times I_{n \times n}$.

The SVSF algorithm is as stated in chapter 4. The boundary layer thickness ψ is set to 0.01 and γ is set to 0.2. The state error covariance matrix P, measurement noise covariance matrix R, and the process noise covariance matrix Q are not required as in the case of the EKF, thus fewer parameters are required to be tuned. Using the SVSF, only one parameter requires tuning that is the boundary layer thickness ψ reducing the complexity of its implementation compared to the EKF. For the SVSF and EKF, weights and biases are updated once at the end of each epoch using the multi-streaming approach as discussed earlier in chapter 4.

For input and output data processing, training data sets are processed before they are used for training. First, input values that are identical for all input vectors are excluded from the training data set. These inputs do not add any information and may lead to numerical problems throughout network training [193]. Secondly, to achieve

faster training response, input vectors are normalized to the range [-1, 1]. A threshold function with value of 0.5 is applied to all neurons in the output layer thus binary outputs are attained.

Concerning training data division and early stopping, the training data set is divided into 50%, 25%, 25% segments for training, validation, and testing, respectively. Training portion is used to calculate the gradient and for weight and bias update. Validation set is used for cross-validation which is performed to assess training quality as training proceeds. Cross-validation is implemented to overcome over-fitting (over-training) [165]. Over-fitting may happen when the training algorithm focuses on training set peculiarities at the cost of losing generalization ability [166]. In other words, the trained network MSE might be minor during training but throughout testing the network may exhibit poor generalization performance.

In this research, cross-validation is performed by using early stopping techniques previously discussed in Chapter 3, namely, generalization loss and training progress stopping criteria. In the following section, the SVSF performance is compared to various training techniques as applied to the three benchmark problems.

5.3 SVSF Performance Compared to Classical Training Techniques

A fully connected feed-forward multilayer perceptron with two hidden layers is used. Table 5.5 summarizes the networks' architecture used for the three proposed

benchmark problems. All software is developed using MATLAB (2009a, MathWorks). The optimized Matlab Neural Networks toolbox is used for implementation of QN, BP, and LM algorithms while a non-optimized algorithms for the SVSF and the EKF are developed from scratch. A nonlinear sigmoidal activation functions $\varphi(.)$ as shown below is used in the hidden layers while linear neurons are employed in the output layer.

$$\varphi(w) = \frac{1}{1 + e^{-aw}} \qquad a > 0 \text{ and } -\infty < w < \infty \tag{5.2}$$

Where, a is set to one in this application.

In this research, multiple training and testing runs have been carried out followed by statistical analysis. This is due to the fact that random initialization arises in training a neural network. Consequently, different results may occur even by applying a similar training technique using the same training data set. Accordingly, 30 runs have been carried out followed by calculating the mean, standard deviation, and the 'best' run using test, training, and validation set error.

Table 5.5. Networks Architecture for the Three Benchmark Problems

Problem	Networks Architecture			
	#Inputs	Hidden#1	Hidden#2	#Outputs
Cancer	9	12	12	2
Diabetes	8	11	10	2
Glass	9	12	11	6

Assessment of training techniques, as stated in the literature (chapter 2), is performed according to MSE variations during training, trained networks generalization capability, ability to avoid premature convergence to local minimum, and number of epochs till convergence (or according to training time). Figures 5.1, 5.2, and 5.3 show the changing MSE over time for the three different cases for the training algorithms using the same network initialization and architecture. For the three benchmark problems, the SVSF has provided convergence in minimum number of epochs compared to Levenberg-Marquardt, Quasi Newton, and batch first order back propagation algorithm. The SVSF achieves comparable performance to the EKF.

For the Cancer problem, at the 2nd epoch, the SVSF reaches MSE of 0.06007 while the MSE for the EKF is 0.095. At the 3rd epoch, the MSE for the SVSF is 0.024 compared to 0.0319 for the EKF. For the Glass problem, at the 2nd epoch, the SVSF MSE is 0.125 while the MSE for the EKF is 0.147. At the 3rd epoch, the MSE for the SVSF is 0.095 compared to 0.1072 for the EKF.

For the Diabetes problem, the SVSF reaches MSE of 0.175 at the 3rd epoch compared to 0.201 for the EKF, while Quasi-Newton, Levenberg Marquardt, and first order BP achieve 0.263, 0.279, and 0.67 respectively. In conclusion, the SVSF provides stability and reaches the minimum MSE in the least number for epochs compared to other standard training techniques.

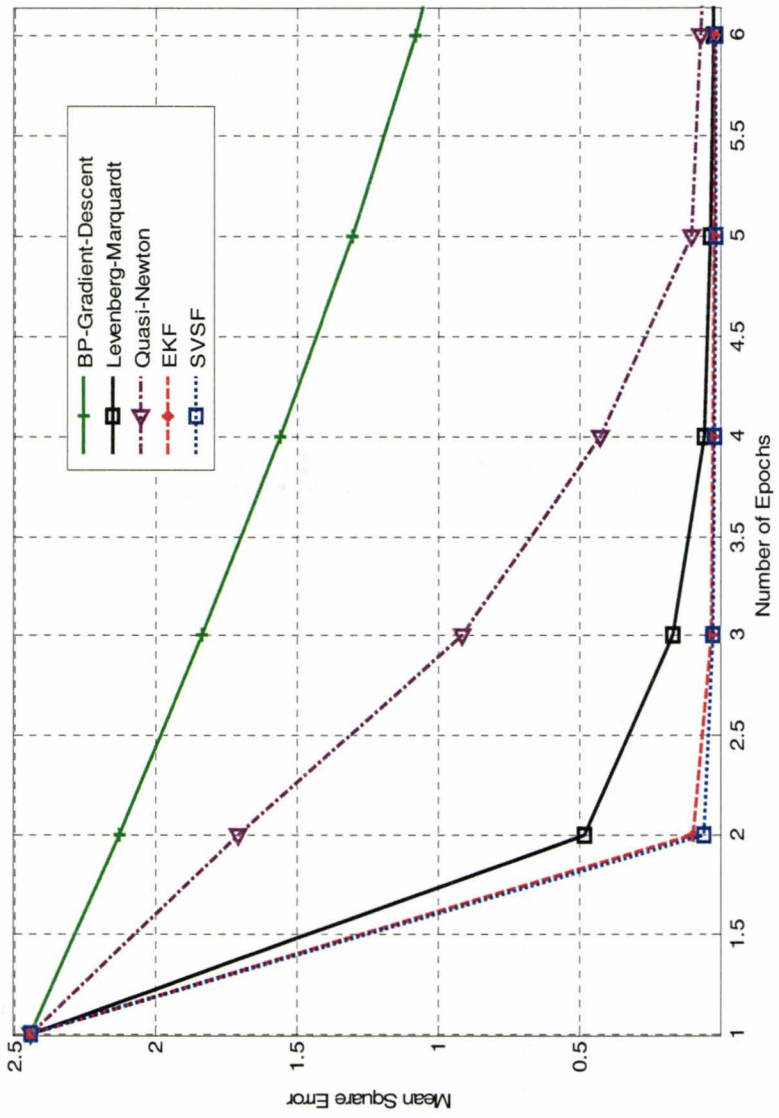

Figure 5.1. Cancer Mean Square Error Variation vs. Number of Epochs

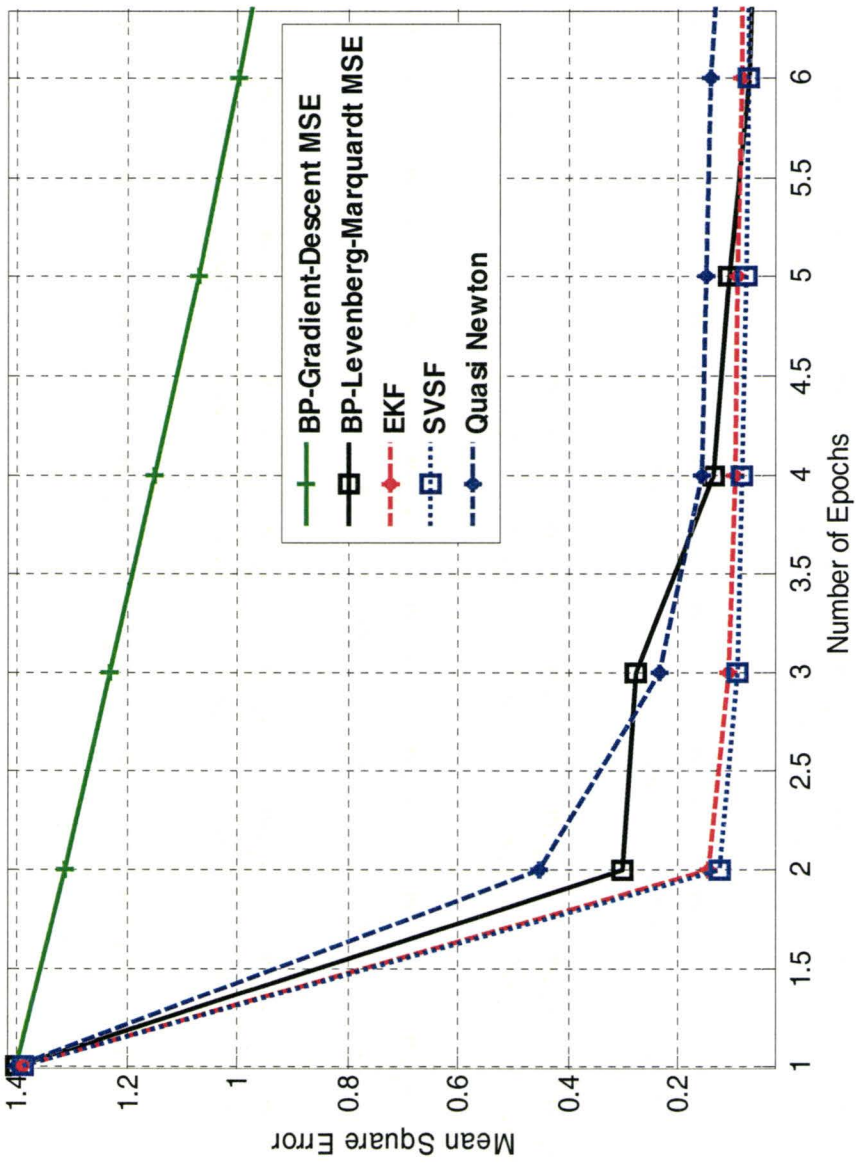

Figure 5.2. Glass Mean Square Error Variation vs. Number of Epochs

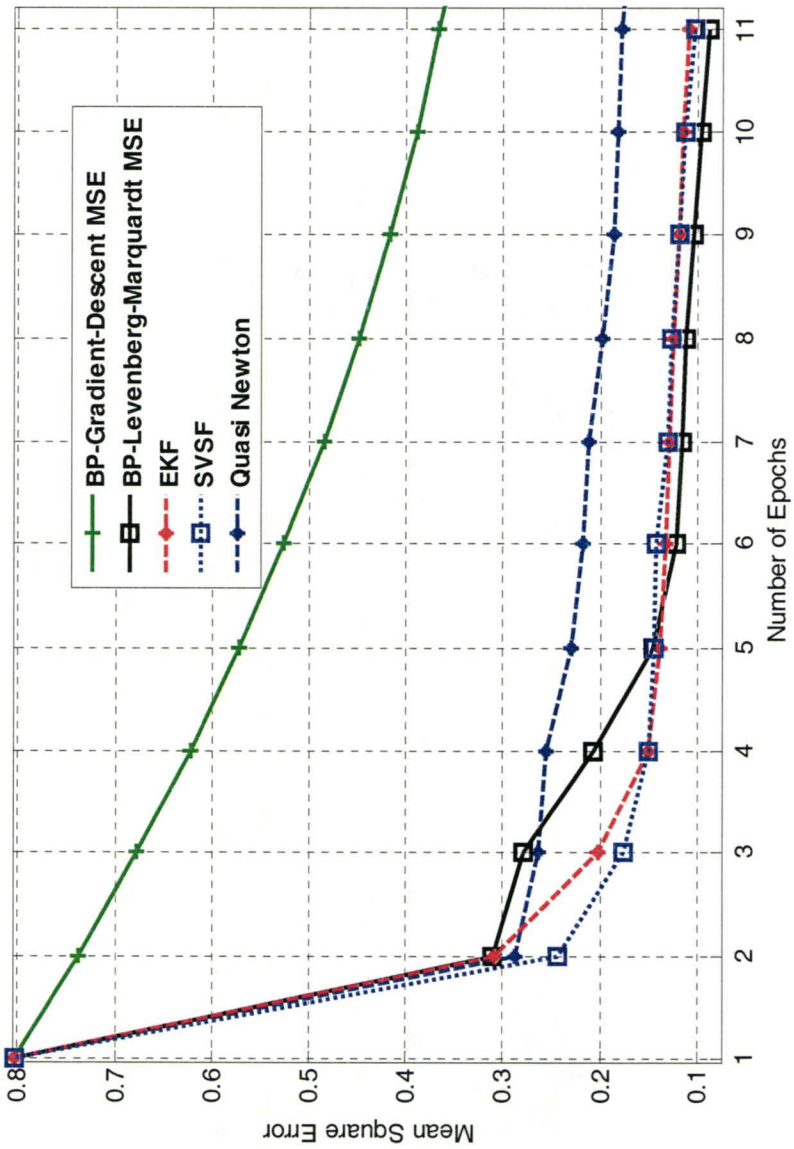

Figure 5.2. Diabetes Mean Square Error Variation vs. Number of Epochs

BP, LM, and QN algorithms are implemented using the Matlab Neural Networks toolbox which is optimized for speed and memory requirements while the SVSF and EKF functions are implemented using non-optimized, personally developed, m-file codes. Accordingly, comparison between the EKF and the SVSF only is provided. Tables 5.6, 5.7, and 5.8 show the average epoch computational time for the EKF and the SVSF.

Table 5.6. Average Epoch Computational Time for the Cancer Problem

Training Technique	EKF	SVSF
Time (Sec)	16.50	17.15

For the Cancer problem, the EKF requires less epoch time compared to the SVSF. For the diabetes problem, the EKF and the SVSF computational times are comparable while for the Glass problem, the SVSF requires less epoch time compared to the EKF.

Table 5.7. Average Epoch Computational Time for the Diabetes Problem

Training Technique	EKF	SVSF
Time (Sec)	12.2	12.32

Table 5.8. Average Epoch Computational Time for the Glass Problem

Training Technique	EKF	SVSF
Time (Sec)	13.14	12.33

Tables 5.9, 5.10, and 5.11 show simulation results using the proposed SVSF training algorithm along with other standard training algorithms. The tables show training and generalization (testing) results on the three selected benchmark problems. Mean squared error percentage is shown as the performance measure. Network weights are initialized 30 times and statistical analysis is performed on the data. Mean squared error, mean standard deviation, best, and worst runs have been recorded in the table in addition to mean number of epochs until one of the stopping criteria stated before is achieved.

For the Cancer problem, the SVSF has shown better results compared to the EKF, LM, and BP. During testing, which is the most important performance measure as it tests trained networks generalization capability, the SVSF provides the best mean generalization after QN. The SVSF's mean testing error is close to Quasi Newton. However, QN algorithm requires more epochs to train which gives the SVSF an advantage over the QN. In addition, the SVSF achieves the least level of standard

deviation during testing compared to EKF, LM, and BP. The SVSF's standard deviation is comparable to the QN. Regarding best and worst runs, SVSF, EKF, LM and QN achieve the same best run followed by BP. Regarding worst run, QN achieves the best performance followed by the SVSF and the EKF with the same performance, then LM, while the first order batch BP is far behind. Regarding number of epochs, the SVSF, EKF, and LM achieve the same number of epochs followed by the QN and the first-order BP.

For the Glass problem, during testing, the SVSF achieves better training and generalization capability (least mean squared error) and mean standard deviation compared to the LM, QN, and BP and provides comparable ones to the EKF. The SVSF and EKF provide the best runs compared to other training techniques and the EKF provides the least of the worst run error. Regarding number of epochs, the EKF, SVSF, LM are the same and less than the QN and BP. For diabetes problem, the QN achieves slightly less mean squared error percentage and standard deviation compared to the SVSF. However, the QN requires more than double the number of epochs to train compared to the SVSF. The LM and the EKF mean squared error percentages along with number of epochs are comparable with the SVSF's.

In conclusion, the SVSF has shown excellent generalization capability and minimum number of epochs to converge, especially for the cancer problem, which makes it a good candidate for feedforward neural networks training. Even though the

EKF and SVSF in their global mode are computationally expensive, they provide an advantage over BP, QN, and LM that they are able to avoid local minima problems by incorporating second order information in the form of the state error covariance matrix P (in case of using the revised SVSF form previously stated in chapter 4). In addition the SVSF can avoid local minima problems due to the switching corrective action that force the state estimates to converge to the actual state estimates.

The original SVSF provides an advantage over the EKF is that it has only one tuning parameter is required that is the boundary layer thickness ψ but three tuning parameters are required in case of the EKF representing the error covariance matrix P, system noise covariance matrix Q, and measurement noise covariance matrix R.

The SVSF and EKF computational requirements can be further enhanced using a decoupled form. In decupled SVSF and EKF forms, instead of grouping all network weights into one single block (as in the global mode), a decoupled SVSF and EKF can be implemented by forming groups of weights where all weights that are targeted to one neuron are considered as a group. Consequently, instead of processing the entire weights simultaneously, small matrices of weights are processed thus helps in decreasing computational requirements.

Table 5.9. Error Rates Using Cancer Data Set for Various Training Techniques

Technique	Training				Generalization				#
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	Epochs
First-Order BP	6.05	4.41	3.14	16.28	7.06	3.74	4.60	16.66	28
Levenberg-Marquardt	1.40	0.71	0.57	2.57	5.34	1.67	3.44	8.62	13
Quasi-Newton	2.37	0.55	1.71	3.71	4.31	0.72	3.44	5.17	26
EKF	1.54	0.62	0.57	2.28	4.77	0.94	3.44	6.32	13
SVSF	1.85	0.52	0.85	2.85	4.65	0.73	3.44	6.32	13

Table 5.10. Error Rates Using Glass Data Set for Various Training Techniques

Technique	Training				Generalization				#
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	Epochs
First-Order BP	69.03	6.86	57.76	88.20	71.32	8.88	52.83	90.57	30
Levenberg-Marquardt	29.48	6.83	17.39	52.17	40.50	7.18	32.08	64.15	12
Quasi-Newton	46.4	6.13	32.30	57.76	53.4	8.23	35.85	64.15	27
EKF	18.36	6.63	8.07	32.30	34.28	3.51	28.30	39.62	12
SVSF	25.61	8.53	14.91	62.11	35.66	4.01	28.30	49.057	12

Table 5.11. Error Rates Using Diabetes Data Set for Various Training Techniques

Technique	Training				Generalization				# Epochs
	Mean	StD	Best	Worst	Mean	StD	Best	Worst	
First-Order BP	36.15	13.44	21.36	66.14	38.60	12.1	23.96	64.06	41
Levenberg-Marquardt	18.86	2.64	14.32	24.74	25.40	3.01	18.75	33.33	10
Quasi-Newton	21.95	1.56	19.01	26.56	24.15	1.48	20.31	26.56	25
EKF	16.14	2.22	11.46	21.35	26.32	2.00	21.87	30.21	10
SVSF	18.40	6.03	14.32	47.66	26.44	4.40	22.92	47.92	10

Chapter 6 Fault detection using Neural Networks trained by the Smooth Variable Structure Filter

Introduction

In this chapter, the relatively new smooth variable structure filter (SVSF) is used for the training of nonlinear multilayered feed forward networks for fault detection applications. The SVSF is applied in a global (GSVSF), multi-streaming mode. The SVSF-based neural networks have been implemented in an industrial application, namely, Engine fault detection and classification using vibration data in the crank angle domain in a four-stroke, eight-cylinder engine. Furthermore, a comparative study between the popular back propagation (BP) method, the extended Kalman filter (EKF), and the SVSF is presented for these applications. Experimental results indicate that the SVSF is comparable with the EKF, and both methods outperform BP.

6.1 Engine Fault Detection Application Using Neural Networks

Throughout the literature, various fault detection and isolation (FDI) techniques have been implemented to detect and classify internal combustion engine faults. As stated in chapter two, they are divided accordingly to signal-based and model-based FDI techniques. In this section, a signal-based FDI technique, as discussed in Chapter four,

using neural networks trained by the SVSF is implemented as the fault detection strategy. Two faults have been induced in the engine; they involve missing bearing fault (MB) and piston chirp (PC) fault. Vibration signals recorded pertaining to these two fault cases as well as measurements data from the baseline fault-free engine are used for neural networks training.

6.1.1 Engine Experimental Setup and Operating Conditions

This section provides a description of the engine's experimental setup and operating conditions. The experimental setup is as shown in Figure 6.1, and involves a four stroke, 5.0 L, eight cylinder, Coyote-VP engine. The test and data collection are performed in a semi-anechoic chamber at FORD's Powertrain Research and Development Centre (PRDC). The semi-anechoic chamber is used to assure that the recorded vibration signal from the system is free from any noise contamination. The system's experimental setup is built in an arrangement that closely imitates the actual vehicle conditions. The setup involves the engine under consideration, clutch and transmission assembly including: alternator, compressor, fan belt, power-steering pump, and engine mounts. The engine under consideration is operated at around 600 rpm with idle conditions [194]. The engine's temperature can impact vibration signals as high temperatures might affect vibration due to changes in clearances due to expansions taking place between engine's

rotating parts. Consequently, the engines' oil and cooling water temperatures are held constant at around 180-190 °F [194].

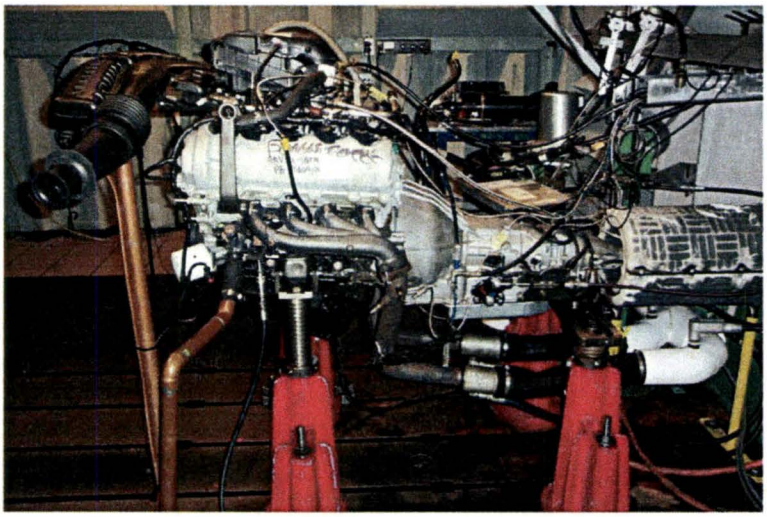

Figure 6.1. Engine Experimental Setup and Transmission unit [194]

6.1.2 Data Acquisition System, Transducers Types, and Positioning

Vibration data is recorded using a charge-type piezoelectric accelerometer. Data is then acquired by a PROSIG 5600 data acquisition system with a built-in 16 bit Analogue to digital (ADC) Converter. Digital vibration data is stored in a PC for offline analysis in order to extract fault condition information.

6.1.2.1 Data Acquisition system

The PROSIG ADC card is set at a sampling frequency of 32,768 Hz and with channel resolution of $0.3\ mV$. The data acquisition system, as shown in Figure 6.2, has eight

analogue input channels. Three channels are used in this research. The first channel is used for accelerometer vibration signal acquisition, the second channel is used for Cam Identification (CID) sensor signal discussed in more details in subsection 6.1.2.4, and the third channel used for measuring rotational speed (RPM).

Figure 6.2. PROSIG 5600 Data Acquisition System [194]

The accelerometer's analogue output voltage, that represents a measure of the engine's acceleration, is being sampled and conditioned using an anti-aliasing filter before being converted to 16-bit digital form using the built-in ADC card.

For our experimental setup, the continuously changing accelerometer's analogue signal is fed to an anti-aliasing filter with a cut off frequency of $13{,}107\ Hz$. The

frequency range of interest is $0\ to\ 10{,}000\ Hz$. In order to satisfy Nyquist-Shannon, since the signal's maximum frequency content is $10{,}000\ Hz$, the minimum sampling frequency that can be used is $20{,}000\ Hz$. By adding a safety factor, the sampling frequency is set to 32,768 *Hz* throughout the entire experiment. Thus, 32,768 samples are acquired every second. For neural networks training, the engine runs for four seconds, thus the total number of samples per engine is as follows:

$$Total\ number\ of\ Samples = 32{,}768 * 4 = 131{,}072\ samples$$

6.1.2.2 ADC Resolution Calculation

The resolution is generally defined as the smallest change in the input that causes change in the output. The voltage resolution of an ADC can be calculated by dividing the overall voltage measurement range by the number of discrete voltage levels

$$Resolution = \frac{Analogue\ Voltage\ Range}{Number\ of\ discrete\ levels} = \frac{V_{max} - V_{min}}{2^N}$$

The PROSIG resolution is calculated as follows, knowing that the analogue voltage range is $-10V\ to\ +10V$, the resolution is:

$$Resolution = \frac{V_{max} - V_{min}}{2^N} = \frac{10 - (-10)}{2^{16}} = \frac{20}{65{,}536} = 0.3\ mV$$

The PROSIG resolution is enhanced by involving a variable gain ranging from 1 to 1000, consequently, by setting the gain to 1000, the maximum resolution of $0.3\ \mu V$ can be attained.

6.1.2.3 Accelerometer Transducers

Piezoelectric accelerometers have been used to measure engine vibration in order to extract fault signatures. The piezoelectric accelerometer uses piezoelectric effect piezoelectric effect is a phenomenon where electric charges are accumulated on the surface of piezoelectric materials when subjected to mechanical strain. By applying a tension or compression force on the piezoelectric material, electric charges are produced on both sides of the material with polarity depending on the force direction. The piezoelectric effect is relatively important as it represents the link between mechanical and electrical states. The piezoelectric effect is reversible; by applying electric field on the piezoelectric crystal, internal forces are generated.

The piezoelectric accelerometer structure, as shown in Figure 6.3, consists mainly of a piezoelectric crystal, a seismic mass, preload stud, and built-in electronics (amplifier). The seismic mass loads the piezoelectric material through the preload stud. The accelerometer can be modeled as a damped mass on a spring as shown in Figure 6.4.

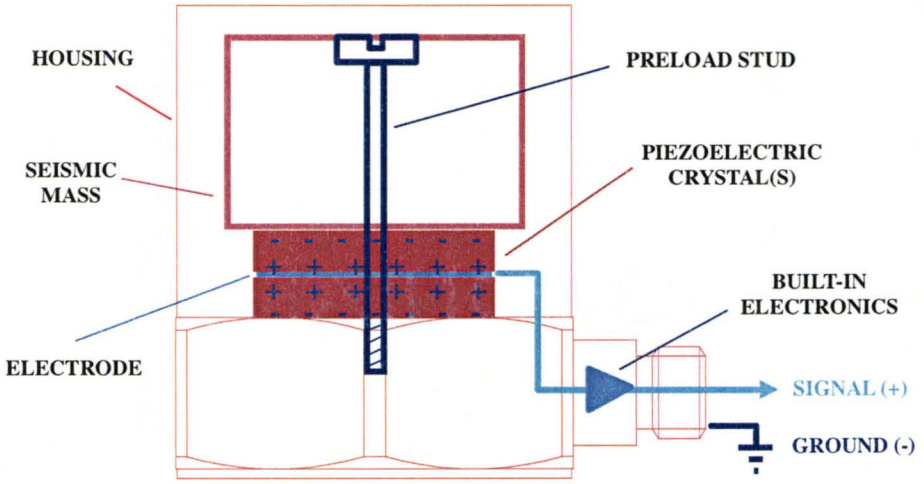

Figure 6.3. Piezoelectric Accelerometer Construction

In practice, the accelerometer is directly attached to the vibrating body, so when the accelerometer is subject to excitation, a force is generated according to Newton's second law (Force=Mass x Acceleration) on the piezoelectric material by the seismic mass which in turn produces electric charge that is proportional to the acceleration exerted. Then the output charge is magnified by an amplifier (Built-in electronics) and fed through a data acquisition system.

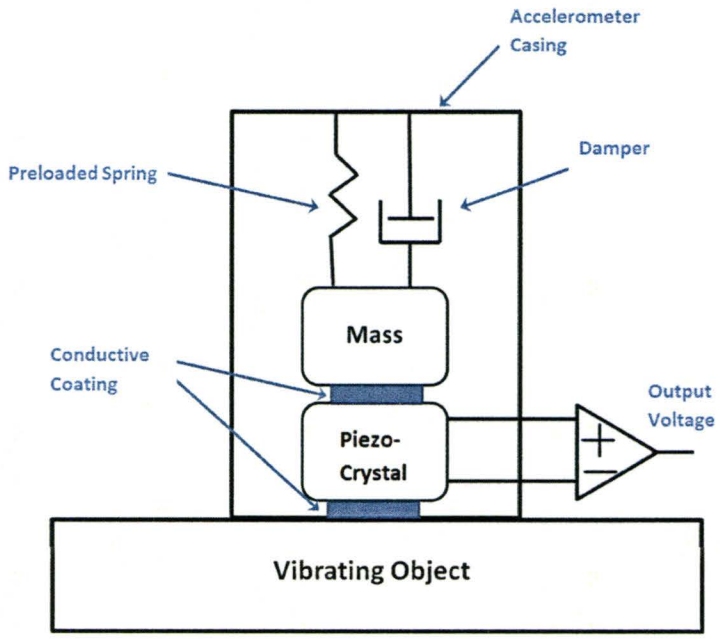

Figure 6.4. Piezoelectric modeling as a mass spring damper element

In our experimental setup, the piezoelectric accelerometer has been attached to the engine head in a predefined position (discussed later) in order to detect faults conditions of interest. The accelerometer model is of Bruel and Kjaer 4366 charge-type piezoelectric accelerometer. Charge-type piezoelectric accelerometers are characterized by the ability to withstand high temperatures, capability to produce accurate results in harsh environments which in turn makes it a good candidate to measure engine vibration.

6.1.2.4 The Transducer Locations

At Ford plant, the experimental setup used four charge-type piezoelectric accelerometers mounted on the engine to measure vibration signals. As shown in Figure 6.5, two accelerometers are placed on the cylinder head and two on the engine lugs. These accelerometers are used to measure vibration signal for detecting fault conditions by comparison with standard charts.

In this research project, two types of faults are considered; missing bearing and piston chirp that occur in one cylinder and can be detected using vibration signals from the upper right accelerometer shown in Figure 6.5.

In addition to this accelerometer, the engine's crank angle speed (in RPM) and position are measured. The engine's rpm is acquired using a tachometer and the signal is fed to the PROSIG data acquisition system. The crank angle position is detected using a cylinder identification (CID) sensor. The CID is an electromagnetic sensor that produces five sinusoidal beats to indicate the location of the camshaft.

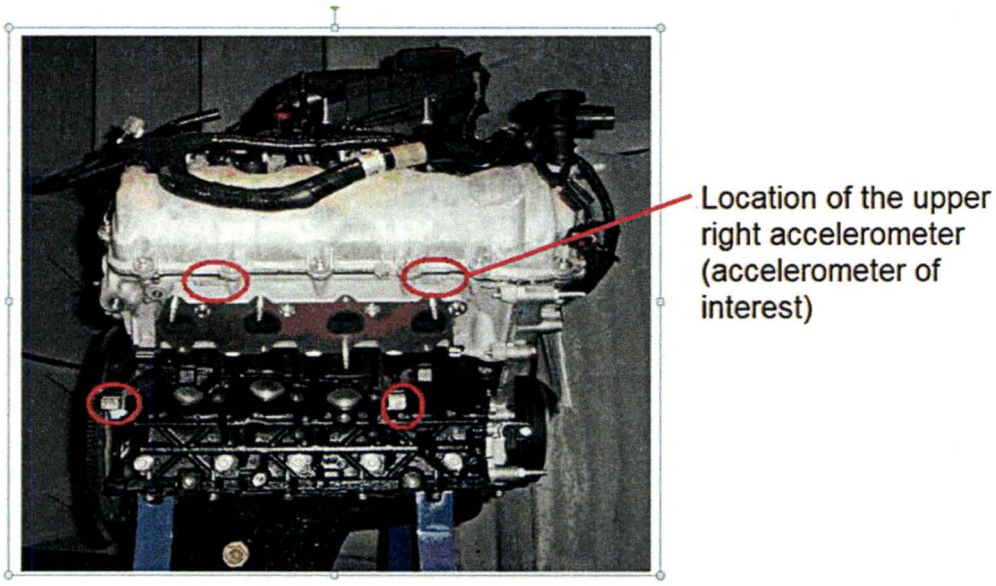

Figure 6.5. Accelerometers Location on the Engine's Lug and Cylinder Head [194]

The CID's signal is used for expressing vibration data in terms of the crank angle domain rather than the time domain which is later used for neural networks training.

After data acquisition, the time domain data is converted to the crank angle domain using the cam identification (CID) sensor signal. CID sensor is used to detect camshaft angle position.

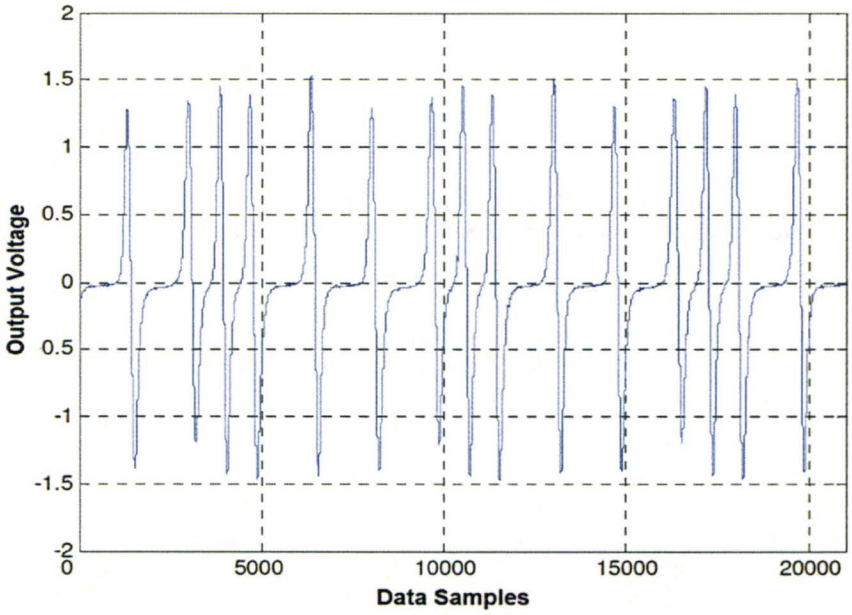

Figure 6.6. CID signal showing five sinusoidal pulses

It is a non-contact sensor mounted on the engine and generates sinusoidal pulses at the specific angles of 90°,120°,60°,120°,60°, 180°, and 90° of the engine cycle as shown in Figure 6.6.

The first sinusoidal pulse zero-crossing indicates that the first cylinder is $10°$ away from the top-dead-center (TDC). Accordingly, the positioning of the cylinder is determined using the CID signal.

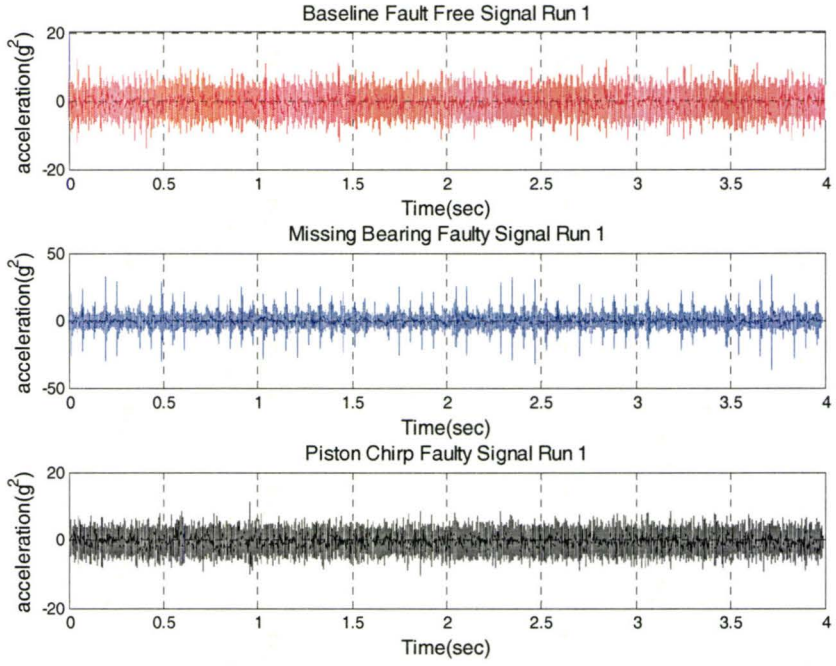

Figure 6.7. Vibration Signal in the Time Domain for the Baseline case, Missing Bearing, and Piston Chirp faults

The accelerometer's vibration signal in the time domain captured for the baseline case and the two faults of interest are as shown in Figure 6.7. Its conversion to the crank angle domain is shown in Figure 6.8.

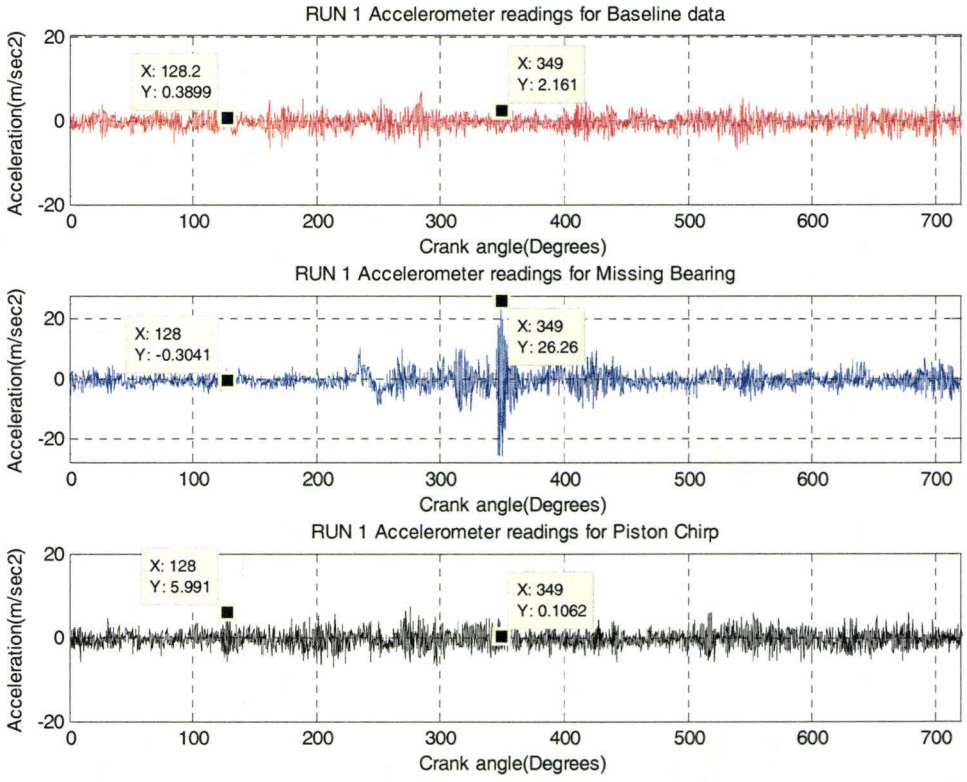

Figure 6.8. Baseline, Missing bearing, and Piston Chirp vibration data in the crank angle domain

By examining the transformed data in the crank angle domain in Figure 6.8, there is a clear spike at a crank angle of $349°$. The acceleration values for the three engine conditions considered (baseline, missing bearing, and piston chirp faults) are 2.61, 26.26, and 0.1062 $m/\sec^2$.

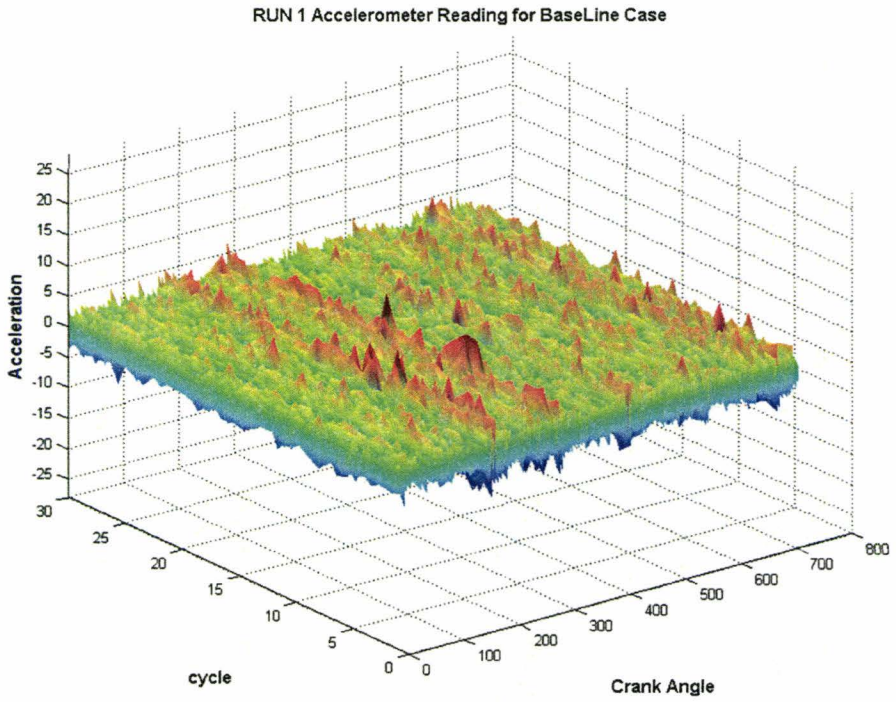

Figure 6.9. Baseline Vibration Data in the Crank Angle Domain for 30 Runs

Consequently, trained networks using various training techniques are able to identify the missing bearing fault with high accuracy, while some networks misclassify baseline and piston chirp faults. Another point of interest that differentiates between the three faults under consideration is at angle of 128°.

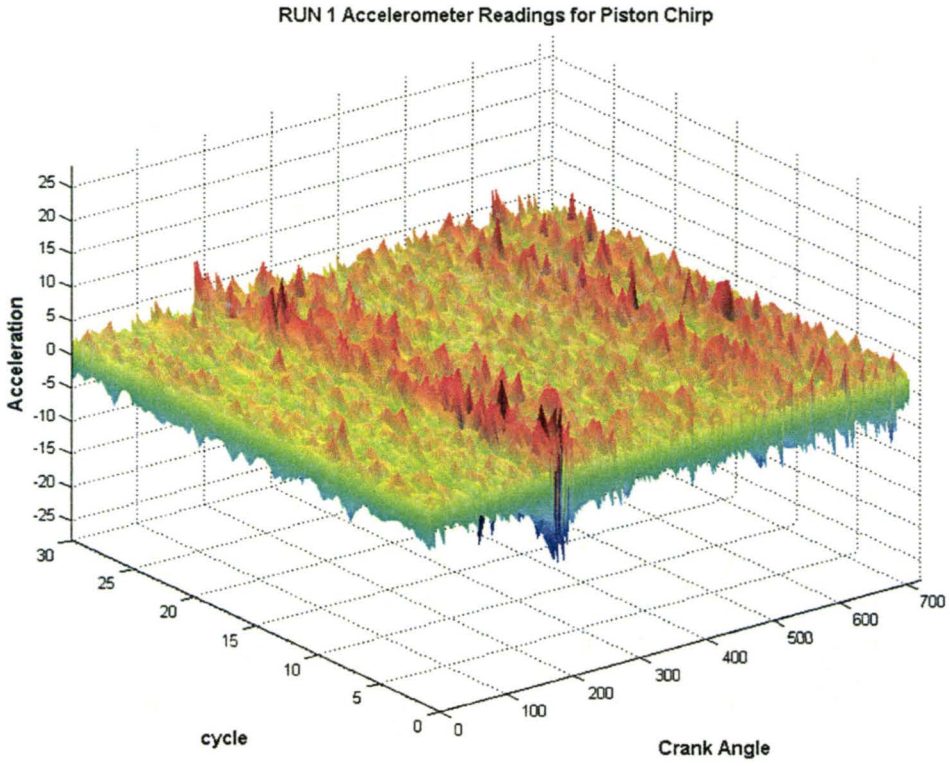

Figure 6.10. Piston Chirp Vibration Data in the Crank Angle Domain for 30 Runs

The acceleration values are 0.3899, -0.3041, and 5.991 $m/\sec^2$ for the baseline, missing bearing, and piston chirp faults, respectively. Figures 6.9, 6.10, and 6.11 show vibration data in the crank angle domain for 30 engine cycles for each fault under consideration.

6.1.3 Faults detection methodology

Vibration signals recorded for the two fault conditions (Missing bearing (MB) and piston chirp (PC)) as well as for the baseline fault-free engine are used for training the neural networks. Data sets include 40 runs with piston chirp fault, 40 with missing bearing, and 40 for fault-free engines. The time domain vibration data are transformed to the crank angle domain as previously discussed. Examples of recorded and transformed data sets for the two faults conditions and the fault-free engines are shown in Figure 6.8.

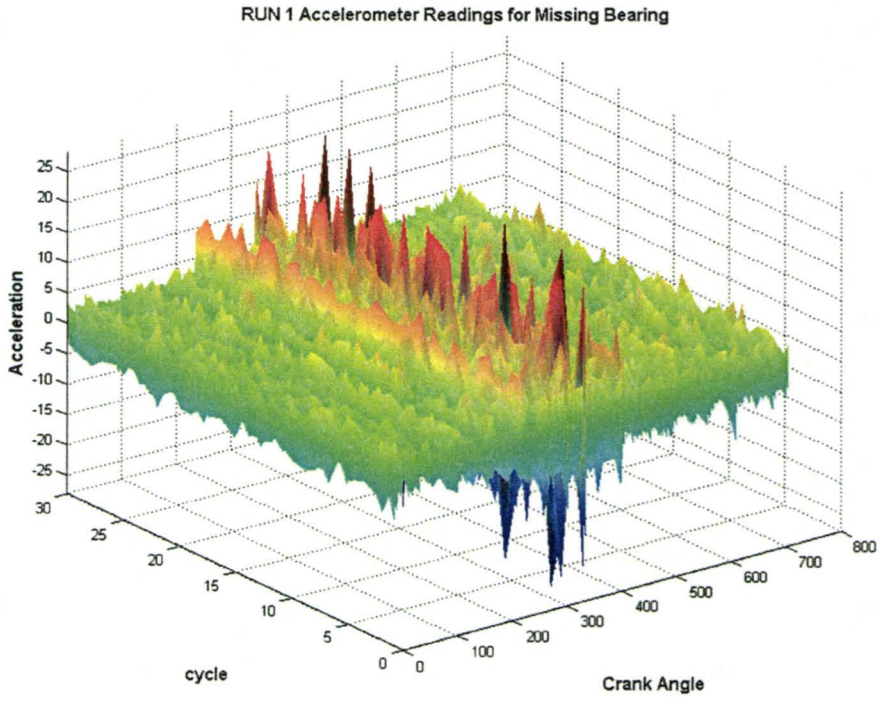

Figure 6.11. Missing Bearing Vibration Data in the Crank Angle Domain for 30 Runs

In this research, fully connected feedforward multilayer perceptron network with one input layer representing the vibration data in the crank angle domain, two hidden layers with four neurons each, and one output layer with three neurons is used. Trained network are required to classify engines to either one of the two faults or to a baseline (fault-free) case as follows: (1, 0, 0: Baseline engine), (0, 1, 0: Piston Chirp fault detected), (0, 0, 1: Missing Bearing fault detected).

The test data has been generated using 40 cycles from each case resulting in 120 data sets. Of these, 90 runs are used for networks training. The remaining 30 runs (10 test runs for each case) are used for validation of neural networks performance.

Networks were trained using the SVSF, EKF, batch first order, and second order BP algorithms (Levenberg-Marquardt and Quasi-Newton). Figure 6.12 shows root mean square error (RMSE) variation for the first 17 epochs. The SVSF convergence properties outperform the conventional first order BP algorithm and it is comparable to the EKF and second order BP. At the sixth epoch the SVSF reaches the least RMSE of 0.0221. Table 6.1 shows the trained networks RMSE using several training techniques after the sixth and the 15^{th} epochs.

Table 6.1. RMSE for Various Training Techniques after the 6th and 15th Epochs

Training Technique	6th Epoch RMSE	15th Epoch RMSE
Batch First-Order BP	0.4276	0.4052
Levenberg-Marquardt	0.1215	2.8e-8
Quasi-Newton	0.1261	0.01189
EKF	0.0522	0.01796
SVSF	0.0221	0.006504

Results are summarized in a matrix form widely known as the confusion matrix. The naming convention of the "confusion" is chosen in published literature as it signifies if the network is "confused" or correct in classifying the classes. As shown in Figure 6.15, each column represents the number and percentage of instances in the actual (Target) class. Each row represents number and percentage of instances in the prediction class. The training and testing results for the networks using 90 data sets and 30 data sets, respectively, are as shown in Figures 6.13 to 6.23.

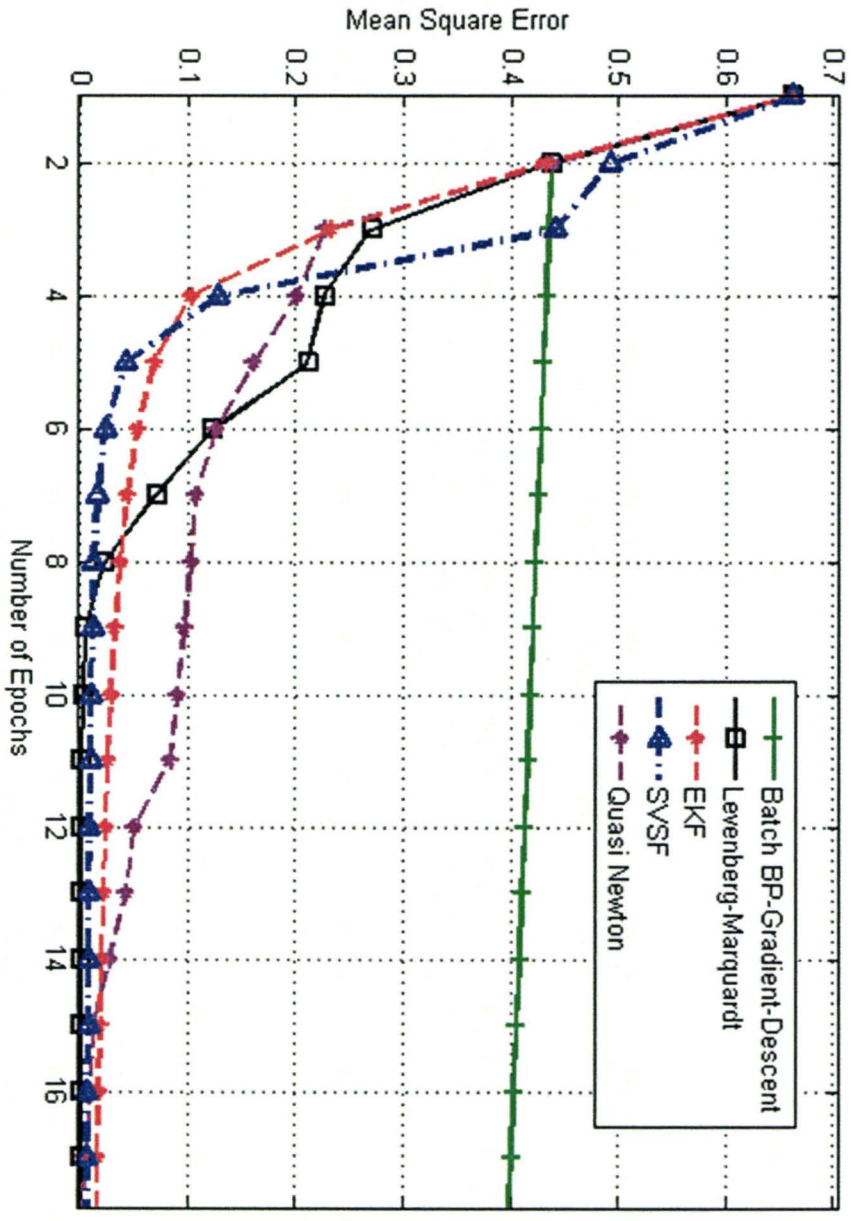

Figure 6.12. Performance (Mean Square Error) Variation Vs. Number of Epochs

Figure 6.13. Training confusion matrix for batch first order BP (after 500 epochs)

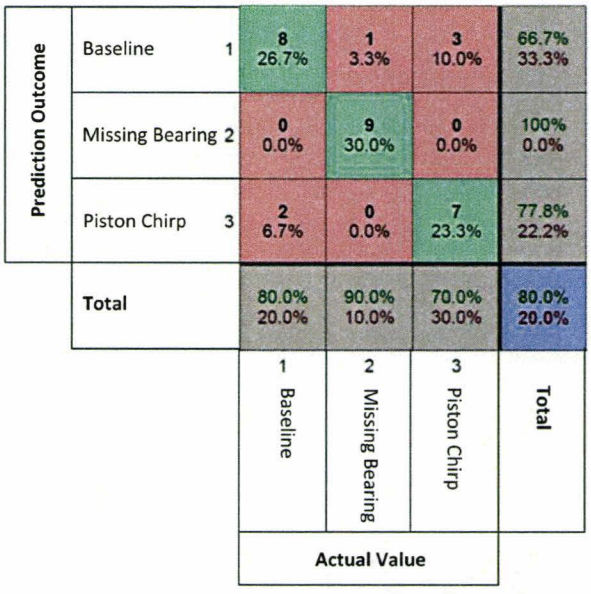

Figure 6.14. Testing confusion matrix for batch first order BP (after 500 epochs)

Training and testing results for the three cases shown in Figures 6.13 to 6.23 are summarized in Table 6.2. The SVSF achieved the highest testing percentage or correct classification in both training and testing along with Quasi-Newton followed by the EKF, the Levenberg-Marquardt, and finally the BP.

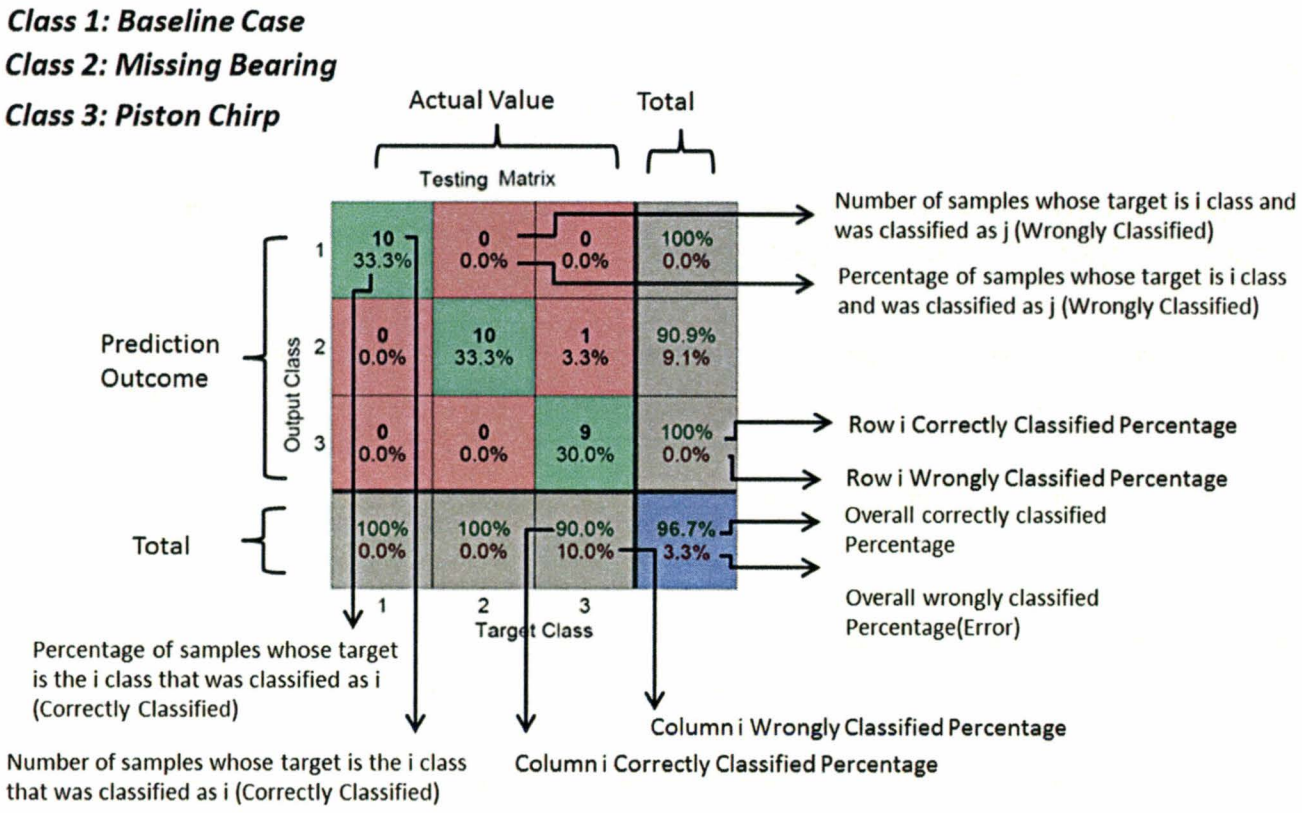

Figure 6.15. Confusion Matrix Illustration

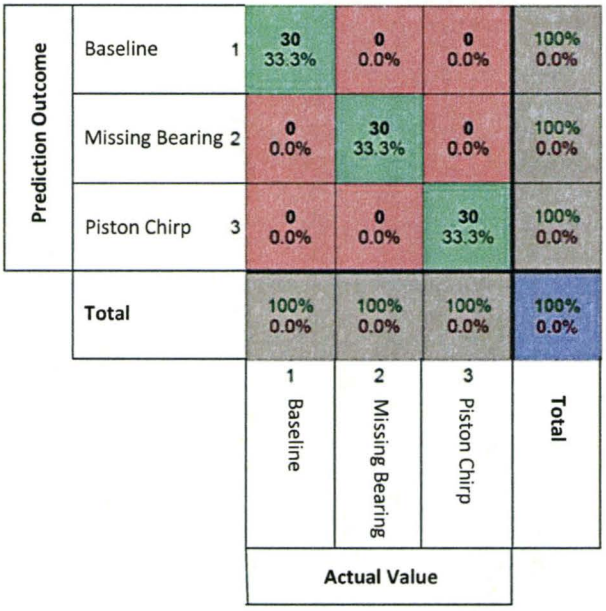

Figure 6.16. Training confusion matrix for Levenberg-Marquardt (after 30 Epochs)

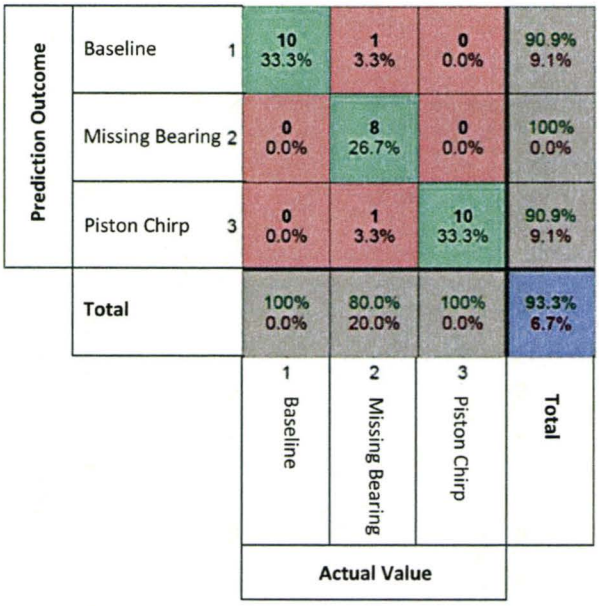

Figure 6.17. Testing confusion matrix for Levenberg-Marquardt (after 30 Epochs)

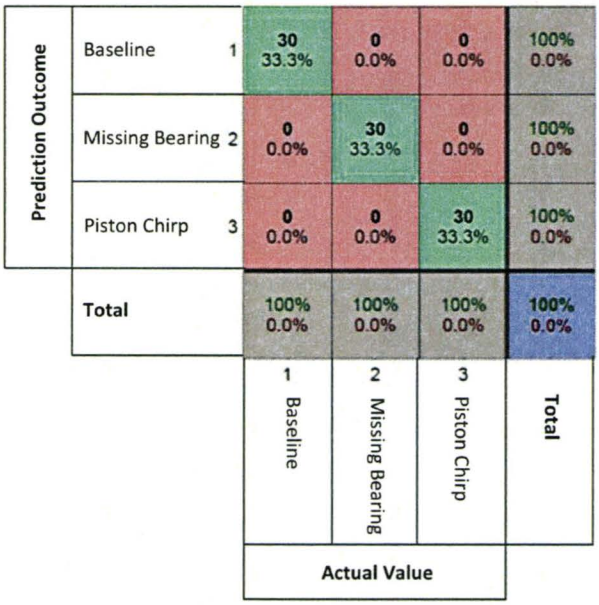

Figure 6.18. Training confusion matrix for Quasi-Newton (after 30 Epochs)

Figure 6.19. Testing confusion matrix for Quasi-Newton (after 30 Epochs)

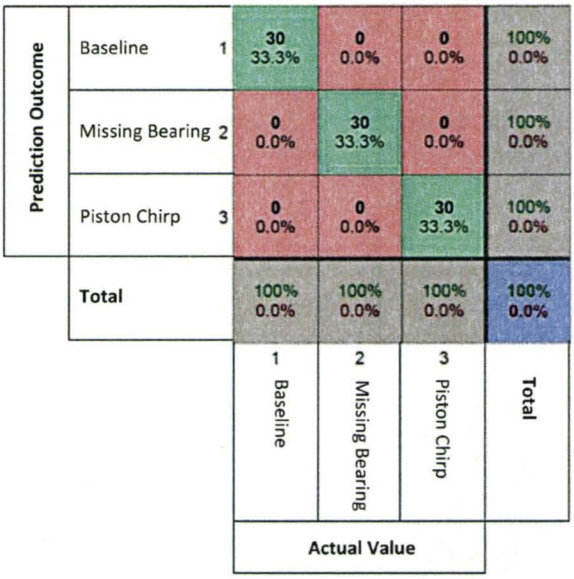

Figure 6.20. Training confusion matrix for EKF

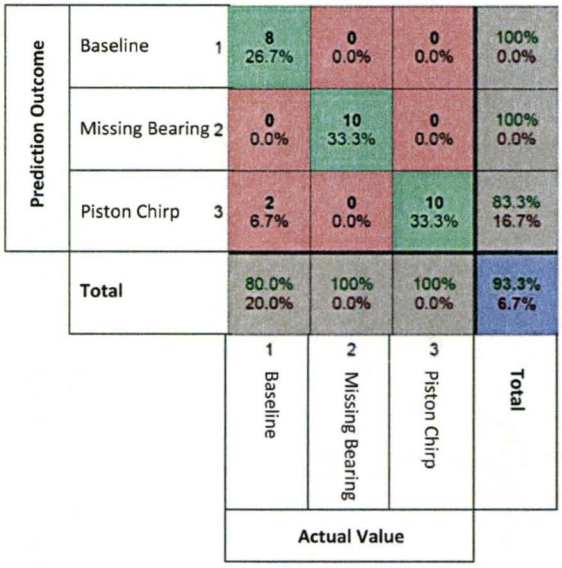

Figure 6.21. Testing confusion matrix for EKF

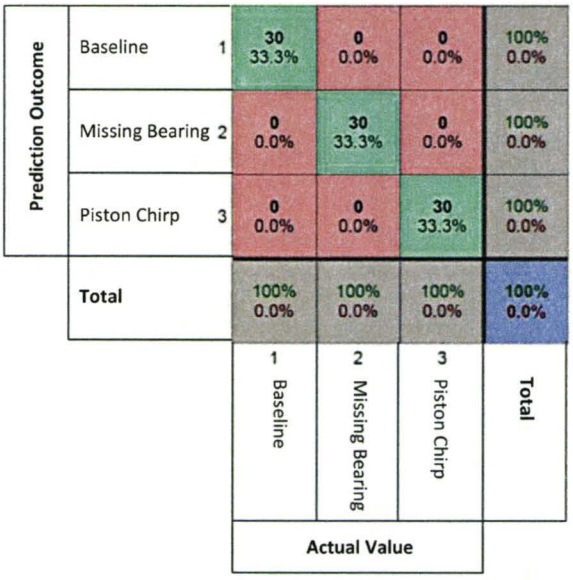

Figure 6.22. Training confusion matrix for SVSF

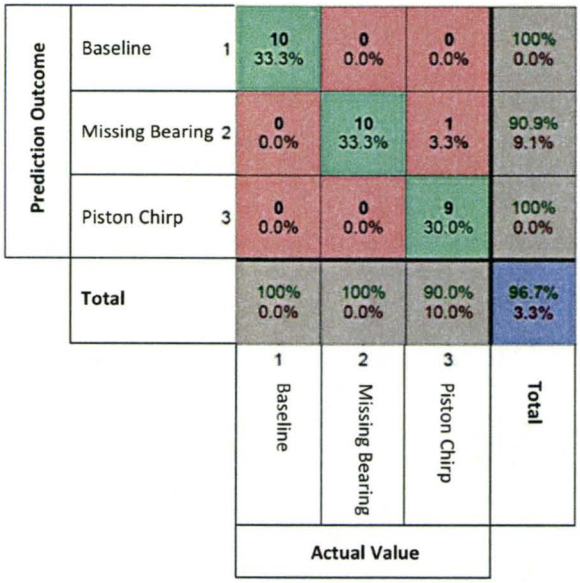

Figure 6.23. Testing confusion matrix for SVSF

Table 6.2. Overall Training and Testing Classification Results

Training technique	Training %	Testing %
First-order BP	91.1	80
Levenberg-Marquardt	100	93.3
Quasi-Newton	100	96.7
EKF	100	93.3
SVSF	100	96.7

Average epoch computational time for the EKF and SVSF are summarized in table 6.3. As previously stated, the EKF and SVSF are non-optimized codes while BP, LM, QN are optimized for speed and memory requirements. The SVSF requires less epoch computational time compared to the EKF.

Table 6.3. Average Epoch Computational Time for Engine Fault Detection Problem

Training Technique	EKF	SVSF
Time (Sec)	25.67	22.61

In Conclusion, in this fault detection application, the GSVSF has been successfully applied to train a multilayer feed-forward network and to detect and classify three engine faults. The GSVSF demonstrated stability, excellent generalization capability, and convergence in minimum number of epochs. The GSVSF training performance outperforms first order back propagation algorithms and is comparable with the GEKF, and second order optimization algorithms in terms of number of epochs. However, the GSVSF has shown better performance compared to EKF in terms of the final RMSE accuracy and generalization capability.

Chapter 7 Conclusions and Recommendations

7.1 Summary

The objective of this research is to propose and develop a new ANN training method based on a new filter known as the Smooth Variable Structure Filter (SVSF). This new method has been tailored to train Feedforward multilayer perceptron (MLP) networks. The SVSF training methodology is applied to real-world benchmark problems contained in the PROBEN1 database. PROBEN1 is internationally recognized and contains training data sets of various challenge levels. It also includes standard guidelines and benchmarking rules that should be followed to assure consistency and repeatability of results.

The new SVSF-based ANN training technique is also applied to a real industrial problem. It is used to detect and classify faults on a Ford diesel engine, namely piston chirp and missing bearing faults. SVSF-based trained network generalization capability and rate of convergence have been compared to other classical training techniques.

7.2 Conclusions and Outcomes

An accurate SVSF-based feedforward Multilayer perceptron (MLP) training technique is developed and tested on several real-world applications. The SVSF, in addition to be an excellent parameter and state estimation strategy, it can be tailored to train feedforward MLP. The SVSF performance is tested on three real-world benchmark problems and compared to other classical training techniques, namely, Back propagation (BP), Quasi-Newton (QN), Levenberg Marquardt (LM), and the Extended Kalman filter (EKF). In general, the proposed technique achieves guaranteed stability, excellent static input-output mapping, good generalization capability, and minimum number of epochs compared to other classical, commonly used, training methods.

For the three benchmark problems (cancer, Glass, and diabetes), simulation results show that the proposed SVSF training technique requires minimum number of epochs to reach convergence similar to the EKF and LM. In addition, results show that the SVSF-based ANN training technique provides comparable performance to other training techniques in terms of generalization capability which is the most important aspect especially for offline training as it tests the ability of trained networks to classify new data not previously seen during training phase.

For the Cancer problem, QN achieved best generalization followed by the SVSF, then the EKF, LM, and the BP achieves worst performance. However, QN require more epochs to train. For the Glass problem, the EKF provides best generalization followed by the SVSF, then the LM, QN, and finally the BP. For diabetes problem, QN achieves the best generalization followed by the LM, EKF, SVSF, and BP. However, the QN require more than double the number of epochs to train compared to EKF, SVSF, and LM.

The SVSF and EKF in their global form are computationally expensive compared to first and second order, optimization-based training techniques. However, the EKF and SVSF can avoid premature convergence to local minima problems by incorporating second-order information in the state error covariance matrix P. In addition to the state error covariance matrix, the SVSF can avoid local minima problems by using a switching chattering action in updating network's weights.

By comparing the original SVSF to the EKF, the SVSF requires only one tuning parameter (boundary layer thickness ψ) while three tuning parameters are required in case of the EKF namely, the error covariance matrix P, system noise covariance matrix Q, and measurement noise covariance matrix R.

A reliable, signal-based, fault detection and isolation strategy applied to industrial application is developed. The proposed SVSF-based ANN is applied to detect and classify faults in a Ford diesel engine. The proposed signal-based ANN fault detection methodology was successful in detecting and classifying two known faults that are commonly occurring and were of interest to Ford, namely, piston chirp and missing bearing faults. The fault detection methodology was also able to identify fault-free (baseline) engines with high accuracy.

It is concluded that accelerometers used for vibration recording are able to provide reliable measurements provided that the test on internal combustion engines is performed in a semi-anechoic chamber in order to provide data free from any noise contamination. In addition, it is concluded that data transformation to the crank angle domain is an essential transformation to allow the networks to detect spikes and irregularities that might occur at specific crank angle values.

The proposed engine fault detection methodology is customizable, it can be implemented in a production environment or at dealerships and assembly plants provided that the faults are known beforehand thus can significantly decrease the

warranty expenses for a company. In addition, the proposed ANN-based FDI can be used to detect more new faults by providing training data (vibration measurements) and expanding the confusion matrix.

Experimental results show that the SVSF-based training technique is able to achieve excellent training performance in terms of generalization capability. The SVSF and QN achieve the same generalization capability with classification accuracy of 96.7% followed by the EKF and LM with 93.3% .The first order BP algorithm achieves poor generalization capability of 80%.

7.3 Research Contributions

(1) A novel, accurate FF MLP training technique based on the SVSF.

(2) A comparative assessment of the proposed technique to others.

- Generalization enhancement while avoiding premature convergence to local minima problems.
- Minimum number of training epochs.
- Fewer tuning parameters (compared to EKF).

(3) A reliable FDI technique to detect and classify engine faults.

- Easy implemented FDI technique at any production environment.
- Robust FDI system with minimum number of sensors.
- Warranty cost reductions at dealerships and assembly plants.
- Customizable FDI to detect new engine faults.

7.4 Recommendations for Future Research

There is potential for further research as follows.

- Application of the proposed SVSF-based ANN training technique should be considered for the training of recurrent multilayer perceptron (RMLP) networks. RMLP networks comprise feedback from one or more hidden layers that is then being fed as an input to the network after using delay elements. Consequently, RMLP introduce dynamics to the neural networks in comparison to feedforward MLP networks that only involve static input-output mappings.
- There is a potential for introducing a decoupled form of the SVSF (DSVSF) to train feedforward and recurrent MLP networks. The decoupled SVSF-based ANN will decrease computational complexity. This is achieved by instead of grouping all network weights into one single block (as in the case of the GSVSF), the DSVSF

would form groups of weights. All weights that are targeted to one neuron are considered as a group.

- The range of faults considered for the Ford engine can be increased.
- The combination of the SVSF with other filtration methods such as the EKF can be considered for ANN training.

APPENDIX

Experimental data are collected at Ford's Powertrain Research and Development Center (PRDC) in Windsor, Ontario, Canada. Data processing and code development are performed at Center of Mechatronics and hybrid technologies (CMHT) at McMaster University.

Nomenclature

-1, +	Notation denoting an inverse and a pseudo inverse, respectively.	
$a^{(b)}$	The b^{th} Derivative of a.	
$\|.\|_{ABS}$	Absolute value	
^	Estimation Value	
T	Matrix Transpose	
$A, \hat{A}$	The time-invariant system matrix and its estimate, respectively.	nxn
$A_k, \hat{A}_k$	The time-variant system matrix at time k and its estimate, respectively	nxn
$B, \hat{B}$	The time-invariant input matrix and its estimate, respectively.	$nx1$
$B_k, \hat{B}_k$	The time-variant input matrix at time k and its estimate, respectively.	$nx1$
$diag(a)$	Create a diagonal matrix with a elements on its diagonal.	
Δa	Difference between $a's$ actual and estimated values.	
$e_{z_{k\|k}}, e_{z_{k\|k-1}}$	The a posteriori and a priori output's estimation error vectors at time k, respectively	$mx1$
$e_{x_{k\|k}}, e_{x_{k\|k-1}}$	The a posteriori and a priori state's estimation error vectors	$nx1$

	at time k, respectively.	
$E(a)$	The expectation operator of the element a	
Ψ	The smoothing boundary layer vector at time	$nx1$
$f(.), \varphi(.)$	Network's activation function	
$i.j$	Subscripts used to identify elements of matrices and vectors	$1x1$
n	System's number of states	$1x1$
m	Number of measurements	$1x1$
K_k	The correction gain of the Smooth Variable Structure Filter at time k	$nx1$
k	Time step value.	$1x1$
I_{nxn}	The identity matrix with dimensions of nxn	nxn
$\lim_{b\to c} a$	The value of a when b approaches c.	$1x1$
h_{ij}	Hessian matrix elements	$1x1$
$P_{k\|k-1}, P_{k\|k}$	The a priori and a posteriori error covariance matrices at time k, respectively	nxn
P_o	The error covariance matrix initial condition	nxn
Q_k	The process noise covariance matrix at time k	nxn
R_k	The measurements noise covariance matrix at time k	mxm
$RMSE$	The root mean square error.	$1x1$
$\sum_{i=b}^{c} a_i$	The summation of vector a from time b to time c	
φ'	First order derivative of the activation function φ	
$sgn(a)$	The sign function of the element a	
T_s	The sampling time	
$\widehat{w}_{k\|k}, \widehat{w}_{k\|k-1}$	The a-posteriori and a-priori estimates at time k, respectively.	$nx1$
$\widehat{w}'_{k\|k}, \widehat{w}'_{k\|k-1}$	The transformed -posteriori and a-priori estimates at time k	$nx1$
z_k, z_o	The output vector at time k and its initial value, respectively.	$mx1$
v_k, V_{max}	The measurement noise at time k and its upper bound, respectively	$mx1$
w_k, W_{max}	The system noise at time k and its upper bound, respectively	$nx1$

P_j	Network layer with j number of inputs	
$\hat{z}_{k\|k}, \hat{z}_{k\|k-1}$	The a posteriori and a priori output's estimation vectors at time k, respectively.	$mx1$
w_k, w_o	The state vector (weight vector) at time k, and its initial and boundary conditions, respectively.	$nx1$
r	Error Residuals vector	
q^{-p}, z^{-p}	Delay operator with p delay samples	
$N_n - 1$	Total number of nodes in the n^{th} layer	$1x1$
$w_{i,j}^n$	Link weight from $node(n,j)$ to the $node(n+1,i)$	$1x1$
α	Learning rate	$1x1$
d_j	Output neuron j target signal	$1x1$
β_{max}	Hessian matrix's largest Eigen value	$1x1$
$s(n)$	Quasi-Newton's direction vector	
m	Number of output neurons	1x1
μ	Back propagation's momentum constant	1x1
v_k	Lyapunov function	
φ_I, φ_{II} and φ_o	activation functions of the first, second and output layers, respectively	
N	Number of training samples	1x1
$C_{k\|linearized}$	The linearized Jacobian matrix	mxn
$E_{valid}(t)$	Validation set squared error at epoch t	$mx1$
K_{kalman}	Kalman gain matrix	$n_i xm$
N_T	Total number of synaptic weights (including bias)	1x1
Γ	Global scaling matrix (or global conversion factor)	mxm
G	Gradient vector	
b_i^n	Node offset (bias) for $node(n,i)$.	1x1
$\{x(n), y(n)\}$	Neural network's n input-output training data set	$nx1$
Node(n, i)	The i^{th} node in the n^{th} layer	
E	The error function	

References

[1] Zhang Runxuan, "Efficient Sequential and Batch Learning Artificial Neural Network Methods for Classification Problems," Singapore, 2005.

[2] E., Hu, M.Y., and Hung, M.S. Patuwo, "Two-Group Classification Using Neural Netoworks ," *Decision Sciences* , vol. 24, pp. 825-845., 1993.

[3] B., and Misra, M. Warner, "Understanding Neural Networks as Statistical Tools," *The American Statistician*, vol. 50, pp. 284-293, 1996.

[4] Roselito A. Teixeira, Antonio De Baraga, and Benjamim De Menezes, "Control of a Robotic Manipulator Using Artificial Neural Networks with On-line Adaptation," *Journal of Neural Processing*, vol. 12, no. 1, Aug. 2000.

[5] G. V. Cybenko, "Approximation by superpositions of a sigmoidal function ," *Mathematics of Control, Signals, and Systems*, vol. 2, 1989.

[6] Christopher M. Fraser, "Neural Networks: Literature Review From a Statistical Perspective," *California State University, Hayward* , 2000.

[7] T.J. Sejnowski, B.P. Yuhas, M.H. Goldstein, and R.E. Jenkins, "Combining visual and acoustic speech signals with a neural network improves intelligibility," *Advances in Neural Information Processing Systems*, vol. 2, pp. 232-239, 1990.

[8] D.A. Pomerleau, *Neural Network Perception for Mobile Robot Guidance.*: Boston: Kluwer, 1993.

[9] W.G. Baxt, "Use of an Artificial Neural Network for Data Analysis in Clinical Decision-Making: The Diagnosis of Acute Coronary Occlusion," *Neural*

Computation, vol. 2, pp. 480-489, 1990.

[10] W.G. Baxt, "Use of an Artificial Neural Network for the Diagnosis of Myocardial Infarction," *Annals of Internal Medicine*, vol. 115, pp. 843-848, 1991.

[11] H. Fujita, T. Katafuchi, T. Uehara, and T. Nishimura, "Application of Artificial Neural Network to Computer-Aided Diagnosis of Coronary Artery Disease in Myocardial Spect Bull's-Eye Images," *Journal of Nuclear Medicine*, pp. 272-276, 1992.

[12] M.T., Engeler, W.E., Frank, P. Leung, "Fingerprint processing using backpropagation neural networks," *Proceedings of the International Joint Conference on Neural Networks*, vol. I, pp. 15-20, 1990.

[13] P.M. Shea and F. Liu, "Operational experience with a neural network in the detection of explosives in checked airline baggage," *Proceedings of the International Joint Conference on Neural Networks*, vol. II, pp. 175-178, 1990.

[14] R.P., Sejnowski, T.J. Gorman, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, pp. 75-89, 1988.

[15] Y. Le Cun, B. Boser, J.S. Denker, and D. Howard Henderson, "Handwritten digit recognition with a back-propagation network," *Advances in Neural Information Processing Systems*, vol. 2, pp. 248-257, 1990.

[16] J., and Gasteiger, J. Zupan, "Neural Networks: A New Method for Solving Chemical Problems or Just a Passing Phase?," *Analytica Chemica Acta*, vol. 248, pp. 1-30., 1991.

[17] J.R. Long, V.G. Gregoriou, and P.J. Gemperline, "Spectroscopic Calibration and Quantitation Using Artificial Neural Networks," *Analytical Chemistry*, vol. 62, pp.

1791-1797, 1990.

[18] P.J. Gemperline, J.R. Long, and V.G. Gregoriou, "Nonlinear Multivariate Calibration Using Principal Components Regression and Artificial Neural Networks," *Analytical Chemistry*, vol. 63, pp. 2313-2323, 1991.

[19] S.P. Chitra, "Using Neural Networks for Problem Solving," *Chemical Engineering Progress*, pp. 44-52, April 1993.

[20] J.M. Hutchinson, "A Radial Basis Function Approach to Financial Time Series Analysis," *Ph.D. dissertation, Massachusetts Institute of Technology*, 1994.

[21] K.Y Tam and M.Y. Kiang, "Managerial Applications of Neural Networks: The Case of Bank Failure Predictions," *Management Science*, vol. 38, pp. 926-947, 1992.

[22] R.M., Sengupta, S.K., Goroch, A.K., Rabindra, P., Rangaraj, N., and Navar, M.S. Welch, "Polar Cloud and Surface Classification Using AVHRR Imagery: An Intercomparison of Methods," *Journal of Applied meteorology*, vol. 31, pp. 405-420, 1992.

[23] B.D. Ripley, "Neural Networks and Related Methods for Classification," *Journal of the Royal Statistical Society, Series B, Methodological*, vol. 56, pp. 409-456, 1994.

[24] A. Lapedes and R. Farber, "Nonlinear Signal Processing Using Neural Network: Prediction and System Modeling," *Los Alamos National Lab Technical Report*, vol. 87, p. 2662, 1987.

[25] R. Sharda, "Neural Networks for the MS/OR Analyst: an Application Bibliography," *Interfaces*, vol. 24, pp. 116-130, 1994.

[26] Z. Tang, C. de Almeida, and P.A. Fishwick, "Time series forecasting using neural networks vs. Box-Jenkins methodology," *Simulation*, no. 57, pp. 303-310, 1991.

[27] Y. Yoon, G. Jr. Swales, and T.M. Margavio, "A Comparison of Discriminant Analysis Versus Artificial Neural Networks," *Journal of the Operational Research Society*, vol. 44, pp. 51-60, 1993.

[28] V. Subramanian, M.S. Hung, and M.Y. Hu, "An Experimental Evaluation of Neural Networks for Classification," *Computers and Operations Research*, vol. 20, pp. 769-782, 1993.

[29] E. Kolman and M. Margaliot, "Knowledge Extraction From Neural Networks Using the All-Permutations Fuzzy Rule Base: The LED Display Recognition Problem," *IEEE journal of Neural Networks*, vol. 18, no. 3, pp. 925-931, 2007.

[30] http://factoidz.com/neurons-impulses-and-neurotransmitters/.

[31] W.S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.

[32] H.S. Stern, "Neural Networks in Applied Statistics," *Technometrics*, vol. 38, pp. 05-214, 1996.

[33] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," no. Psychological Review 65, pp. 386-408, 1959.

[34] F. Rosenblatt, *Principles of Neurodynamics, New York: Spartan Books.*, 1962.

[35] B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," *IRE WESCON Convention Record*, vol. 4, pp. 96-104, 1960.

[36] M., and Papert, S. Minsky, "Perceptrons," *MA: MIT Press, Cambridge*, 1969.

[37] R.J. Schalkoff, "Artificial Neural Networks," *New York: McGraw-Hill*, 1997.

[38] P. Werbos, "Backpropagation through time; what it does and how to do it," *Proceedings of the IEEE*, vol. 78, pp. 1550–1560, 1990.

[39] Simon Haykin, *Neural Networks - A Comprehensive Foundation*, 3rd ed.: Prentice hall, 2007.

[40] Simon Haykin, *Kalman filtering and neural networks*, 3rd ed.: Prentice Hall, 2001.

[41] S. Saarinen, R.B. Bramley, and G. Cybenko, "The numerical solution of Neural-Network training problems," *Center For Supercomputing Res., Univ. Illinois, Urbana* , 1991.

[42] Zhou Gulan and Jennie Si, "Advanced Neural-Network Training Algorithm with Reduced Complexity Based on Jacobian Deficiency," *IEEE transactions on Neural Networks*, 1998.

[43] M. T. Hagen and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Networks* , vol. 5, pp. 989-993, 1994.

[44] R. L. Watrous, "Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization," *IEEE first international Conference Neural Networks* , vol. 2, pp. 619-627, 1987.

[45] Rudy Setiono and Lucas Chi Kwong Hui, "Use of a Quasi-Newton Method in a Feedforward Neural Network Construction Algorithm," *IEEE Transactions on Neural Networks*, vol. 6, January 1995.

[46] Felix Heimes, "Extended Kalman Filter Neural Network Training: Experimental Results and Algorithm Improvements," *IEEE*, vol. 0-7803-4778-1/98, 1998.

[47] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman algorithm," *Advances in Neural Information Processing Systems 1*, pp. 133–140, 1989.

[48] G.V. Puskorius and L.A. Feldkamp, "Decoupled extended Kalman filter training of feedforward layered networks," *Proc. IJCNNÕ91 I, Seattle*, pp. 771-777, 1991.

[49] R.J. Williams, "Training recurrent networks using the extended Kalman filter," *Proc. IJCNN'92 IV*, pp. 241-246, 1992.

[50] Pu Sun and Marko Kenneth, "Training recurrent neural networks for very high performance with the extended Kalman algorithm," *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 8, pp. 121-126, 1998.

[51] Xiaolong Deng, Xie Jianying, Weizhong Guo, and Jun Liu, "A new learning algorithm for diagonal recurrent neural network," in *First International Conference on Natural Computation*, 2005, pp. 44-50.

[52] L.A. Feldkamp, T.M. Feldkamp, and D.V. Prokhorov, "Neural network training with the nprKF ," in *Proceedings of the International Joint Conference on Neural Networks*, 2001 , pp. 109-114.

[53] E. A. Wan and R. van der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium.*, 2000, pp. 153-158.

[54] S. J. Julier, J.K. Uhlmann, and H. F. Durrant-Whyte, "A New Approach for Filtering Nonlinear Systems," in *Proceedings of the American Control Conference*, 1995, pp.

1628-1632.

[55] S.J. Zhang, Jing, Z. L., and J.X. Li, "Fast learning high-order neural networks for pattern recognition," *Electronics Letters*, vol. 40, no. 19, pp. 1207-1208, 2004.

[56] H., Taguchi, A., Sone, M. Kawata, "Pattern error equality learning of neural network with the genetic elgorithm," *Transactions of the institute of Electronics, Information and Communication Engineers*, vol. J81D-II, pp. 2841-9, Dec 1998.

[57] Carl Looney, "Stabilization and speedup of convergence in training feedforward neural networks," *Journal of Neurocomputing*, vol. 10, pp. p 7-31, 1996.

[58] Valeriy Dubrovin, Yuriy N. Vunukov, and Sergey A. Subbotin, "The algorithm of synthesis of logically transparent multilayer neural networks," *Proceedings of Artificial Neural Networks in engineering Conference*, vol. 13, pp. 21-26, 2003.

[59] S.M., Lee, G.S. Yang, "Neural network design by using Taguchi method," *Transactions of the ASME. Journal of Dynamic Systems*, vol. 121, no. 3, pp. 560-3, 1999.

[60] C.C., Chang, T.Y.P., Xu, Y.G., To, W.M. Chang, "Selection of training samples for model updating using neural networks," *Journal of Sound and Vibration*, vol. 249, no. 5, pp. 867-883, 2003, 2002.

[61] Ehsan Sobhani-Tehrani, *Fault Diagnosis of Nonlinear Systems Using a Hybrid Approach.*: ISSN 0170-8643, 2009.

[62] M.L. Leuschen, I.D. Walker, and J.R. Cavallaro., "Fault Residual Generation via Nonlinear Analytical Redundancy," *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, vol. 13, MAY 2005.

[63] E. Y. Chow and A. S. Willsky, "Analytical redundancy and the design of a robust failure detection systems," *IEEE Trans. Autom. Control*, vol. AC-29, pp. 603-614, 1984.

[64] Chung-Chi Hsieh, "Optimal task allocation and hardware redundancy policies in distributed computing systems," *European Journal of Operational Research*, vol. 147, pp. 430-447, June, 2003.

[65] W.V Subbarao, "Software redundancy or hardware redundancy," in *Southcon/90 Conference Record, 20-22 March 1990 , Orlando, FL, USA* , 1990.

[66] H. Jonuscheit, "Acoustic Tests on Combustion Engines in Production," 2000.

[67] Mano Ram Maurya, Paritosh Praveen K., Venkatasubramanian Venkat , and Raghunathan Rengaswamy, "A framework for on-line trend extraction and fault diagnosis," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 6, pp. 950-960, 2010.

[68] B.R. Bakshi and G. Stephanopoulos, "Representation of process trends—III. Multi-scale extraction of trends from process data," *Computers and Chemical Engineering*, vol. 18, no. 4, pp. 267-302, 1994.

[69] S. Postalcioglu, Erkan K., and E.D. Bolat, "Discrete wavelet analysis based fault detection," *WSEAS Transactions on Systems*, vol. 5, no. 10, pp. 2391-7, October 2006.

[70] W. Sun, A. Palazoglu, and J. A. Romagnoli, "Detecting abnormal process trends by wavelet-domain hidden Markov models," *Aiche Journal* , vol. 49, no. 1, pp. 140-150, 2003.

[71] Fang Liu, "Data-based Fault Detection and Isolation (FDI) for a nonlinear ship propulsion system," Simon Fraser University , Maste's thesis 2004.

[72] Venkat Venkatasubramanian, Kewen Yin Raghunathan Rengaswamy, and Surya N. Kavuri, "A review of process fault detection and diagnosis Part III," *Process history based methods, Computers and Chemical Engineering*, vol. 27, pp. 327–346, 2003.

[73] Srinivas Katipamula and Michael R. Brambley, "Methods for fault detection, diagnostics, and prognostics for building systems - a review," *part I, Int. J. HVAC & R Research*, vol. 11, pp. 3–26, 2005.

[74] A. Ramirez-Trevino, E. Ruiz-Beltran, I. Rivera-Rangel, and E. Lopez-Mellado, "Online fault diagnosis of discrete event systems. A Petri net-based approach," *IEEE Transactions on Automation Science and Engineering*, 2007.

[75] R. Debouk, "Diagnosis of discrete event systems- a modular approach," *in Proc. IEEE Conf. SMC.*, 2003.

[76] P.M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - A survey," *Automatics*, pp. 459-474, 1990.

[77] B. Jiang and F. Chowdhury, "Observer-based fault diagnosis for a class of non-linear systems," *in Proc. Of American control conference*, July 2004.

[78] S.A. Ashton and D.N Shields, "Fault detection observer for a class of nonlinear systems," *New Directions in Nonlinear Observer Design (Lecture Notes in Control and Information Sciences)*, vol. 244, pp. 353-373, 1999.

[79] D.N. Shields, S. Ashton, and S. Daley, "Fault detection for pipelines: A nonlinear observer approach," *IEE Colloquium*, vol. 174, pp. 10/1-10/9, April 1997.

[80] H. Hammouri, M. Kinnaert, and E. H. El Yaagoubi, "Observer-based approach to fault detection and isolation for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 10, pp. 1879-1884, 1999.

[81] C. Edwards, S.K. Spurgeon, and R. J. Patton, "Sliding mode observers for fault detection".

[82] C.M. Hajiyev and J. Caliskan, "Fault detection in flight control systems based on the generalized variance of the Kalman filter innovation sequence," *American Control Conference*, vol. 1, 1999.

[83] L. An and Sepehri N., "Hydraulic actuator circuit fault detection using extended Kalman filter," in *American Control Conference* , 2003.

[84] N. Tudoroiu and M. Zaheeruddin, "Fault Detection and Diagnosis of Valve Actuators in Discharge Air Temperature (DAT) Systems, using Interactive Unscented Kalman Filter Estimation," *Industrial Electronics, IEEE International Syposuim*, 2006.

[85] S. Wang, Burton R., and S. R. Habibi, "A smooth variable structure filter for state estimation," *Control and Intelligent Systems*, vol. v 35, n 4, pp. 386-393, 2007.

[86] Moskwa Kao and J. John, "Nonlinear diesel engine control and cylinder pressure observation," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 7, no. 117, n 2, pp. p 183-192, Jun 1995.

[87] Kao Minghui and John J. Moskwa, "Model-Based Engine Fault Detection Using Cylinder Pressure Estimates from Nonlinear Observers," *Transactions of the ASME*, 1994.

[88] Rama K. Yedavalli, "Robust Estimation and Fault Diagnostics for Aircraft Engines with Uncertain Model Data," 2007.

[89] X. Ding and P.M. Frank, "Fault Diagnosis Using Adaptive Observers," in *Singapore International Conference on Intelligent Control and Instrumentation*, 1992.

[90] J. Chen, R.J. Patton, and G.P. Liu, "Detecting incipient sensor faults in flight control systems," in *Proceedings of teh IEEE Conference on Control Applications*, 1994, pp. 871-876.

[91] H.Schneider and P.M. Frank, "Observer-based supervision and fault detection in robots using nonlinear and fuzzy logic residual evaluation," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 3, pp. 274-82, May 1996.

[92] Wenfei Li, Yedavalli, and K. Rama, "Dynamic threshold method based aircraft engine sensor fault diagnosis," in *Proceedings of the ASME Dynamic Systems and Control Conference*, 2008.

[93] Ashton Shields, "A robust fault detection observer for a class of hydraulic systems," *Systems Science*, vol. 25, pp. 25-45, 1999.

[94] J.A.F Vinsonneau, Shields King, P.J., and K.J Burnham, "Modelling and observer-based fault detection for an automotive drive train," in *European Control Conference ECC*, 2003, pp. 240-245.

[95] Escobet Teresa and Travé-Massuyès Louise, "Parameter estimation methods for fault detection and isolation," *In Proceeding of the 12th International Workshop on Principles of diagnosis*, pp. Via Lattea, Italy, 2001..

[96] Martin Oscar Giacomeeli, "Fault detection on an aircraft and development of a contingency control strategy," Lulea University of Technology, Master's Thesis

2008.

[97] A. Pouliezos, G. Stavrakakis, and C. Lefas, "Fault detection using parameter estimation," *Quality and Reliability Engineering International Conference*, vol. 5, no. 4, pp. 283-290, Oct 1989.

[98] T. Jiang, K. Khorasani, and S. Tafazoli, "Parameter estimation-based fault detection, isolation and recovery for nonlinear satellite models," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 799-808, July 2008.

[99] F.Bagheri, H.Khaloozaded, and K. Abbaszadeh, "Stator Fault Detection in Induction Machines by Parameter Estimation, Using Adaptive Kalman Filter," in *Mediterranean conference on control and automation*, 2007.

[100] Mattias Nyberg, "Parity Functions as Universal Residual Generators and Tool for Fault Detectability," *IEEE Decision and Control conference*, vol. 5, pp. 4483-4489, 1997.

[101] Potter J.E. and M.C. Suman, "Threshold redundacy management with arrays of skewed instruments," *Integrity electron, flight control Systems* , pp. 11-25, 1977.

[102] E. Chow and A. Willsky , "Analytical redundancy and the design of robust failure detection," *IEEE Transactions on Automatic Control*, vol. 29, no. 7, pp. 603-614, 1984.

[103] S. K. Neguang, P. Zhang, and S. Ding, "Parity based fault estimation for nonlinear systems:An LMI approach," *In Proceedings of American Conrol Conference*, pp. 5141-5146, 2006.

[104] T Hofling, "Detection of parameter variations by continious time parity equations,"

IFAC World Congress, pp. 513-518, 1993.

[105] J.J. Gertler and R. Monajemy , "Generating directional residuals with dynamic parity equations," *Proceedings of the 12th Triennial World Congress of the International Federation of Automatic Control. Vol.5. Associated Technologies and Recent Development*, pp. 507-512, 1994.

[106] Hong Jin and Zhang Hongyue , "Failure detection using optimal parity vector sensitive to special sensor failure," , *International Conference on Multi sensor Fusion and Integration for Intelligent Systems*, pp. 55-61, 1996.

[107] R.J. Patton and J. Chen, "Review of parity space approaches to fault diagnosis for aerospace systems," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 2, pp. 278-285, 1994.

[108] Paul Van, Bos, André Gelder, "Fault detection and isolation of a cryogenic rocket engine combustion chamber using a parity space approach," in , *3rd IEEE International Conference on Space Mission Challenges for Information Technology*, 2009, pp. 341-345.

[109] N. Pennacchi, P., Tanzi, E., Vania, A. Bachschamid, "Identification of Multiple Faults in Rotor Systems," *Journal of Sound and Vibration* , vol. 254(2), pp. 327-366, 2002.

[110] P. Pennacchi, E. Tanzi and A. Vania N. Bachschmid, "Identification of Transverse Crack Position and Depth in Rotor Systems," *Mechanica*, pp. 563-582, 2000.

[111] R. J. Patton, C. Lopez-Toribio, and F. J. Uppal, "Artificial intelligence approaches to fault diagnosis," *IEEE Colloquium on Condition Monitoring: Machinery, External Structures and Health*, vol. 5, pp. 1–18, 1999.

[112] E. Sobhani-Tehrani, K. Khorasani, and S. Tafazoli, "Dynamic Neural Network-based Estimator for Fault Diagnosis in Reaction Wheel Actuator of Satellite Attitude Control System," *In Proceedings of the International Joint Conference on Neural Networks*, 2005.

[113] Xun Wang, Neil McDowell, Kruger, Uwe, and Geoffre McCullough, "Semi-physical neural network model in detecting engine transient faults using the local approach," *17th World Congress International Federation of Automatic Control, IFAC*, July 2008.

[114] A. Yazdizadeh and K. Khorasani, "Nonlinear system identification using embedded dynamic neural networks," *IEEE on Computational Intelligence*, 1998.

[115] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, 1993.

[116] C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Transactions on Neural Networks*, vol. 6, 1995.

[117] F. Abdollahi, H.A. Talebi, and R. V. Patel, "Stable identification of nonlinear systems using neural networks: theory and experiments," *IEEE/ASME Transactions on Mechatronics*, 2006.

[118] H. A. Talebi, K. Khorasani, and S. Tafazoli, "A Recurrent Neural-Network-Based Sensor and Actuator Fault Detection and Isolation for Nonlinear Systems with Application to the Satellite's Attitude Control Subsystem," *IEEE transactions*, 2009.

[119] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transaction on Neural Networks*, 1990.

[120] Hansen, Larsen Svarer, "Design and evaluation of Tapped-Delay Neural Network architectures," *IEEE international conference on Neural networks*, 1993.

[121] M. Ayoubi, "Nonlinear dynamic systems identification with dynamic neural networks for fault diagnosis in technical processes," *IEEE international conference*, 1994.

[122] Duyar, Guo, Merrill Sarvanan, "Modeling of the Space Shuttle Main Engine Using Feed-forward Neural Networks," *Proceeding of the American Control Conference*, 1993.

[123] Mark A. Motter, Principe, and C. Jose , "Gamma memory neural network for system identification," *IEEE International Conference on Neural Networks - Conference Proceedings*, vol. 5, pp. 3232-3237, 1994.

[124] J.C. Principe and M. Motter, "System identification with dynamic neural networks," *In World Congress on Neural Networks*, 1994.

[125] Leonard J A and M A Gamer, "Diagnosing Dynamic Faults using modular neural-nets," *IEEE Expert Systems Magazine*, 1993.

[126] Naidu S., E. Zafirou , and T. J. McAvoy , "Use of neural-networks for failure detection in a control system," *IEEE Control Sys. Magazine*, 1990.

[127] M. H. Terra and Tin R., "Fault detection and isolation in robotic manipulators via neural networks - a comparison among three architectures for residual analysis," *Journal of Robotic Systems*, vol. 18, no. 7, pp. 357-374.

[128] A. Jabbari, R. Jedermann, and W. Lang, "Application of Computational Intelligence for Sensor Fault Detection and Isolation ," *International Journal of Computer and*

Information Engineering , 2007.

[129] B. Koppen-Seliger, P.M. Frank, and A. Wolff, "Residual evaluation for fault detection and isolation with RCE neural networks," *Proceedings of the American Control Conference*, vol. 5, 1995.

[130] M.H. Terra and R. Tinos, "Fault detection and isolation in robotic systems via artificial neural networks," *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 2, 1998.

[131] Richard O Duda, Peter Hart, and David Stork, *Pattern Classification.*: Wiley, 2000.

[132] Sergios Theodoridis and Konstantinos Koutroumbas, *Pattern Recognition.*, 2003.

[133] Wesley E. Snyder and Hairong Qi., *Machine Vision.*: Cambridge University, 2004.

[134] N. Morgan and H. Bourald, "Continuous speech recognition using multi-layer perceptrons with hidden Markov models," in *In International Conference on Accoustics, speech, and signal processing*, 1990, pp. 413-416.

[135] Rangachar Kasturi and Mohan Trivedi, *Image Analysis Applications.*: Marcel Dekker, 1990.

[136] Patrice Degoulet, *Introduction to Clinical Informatics.*: Springer, 1996.

[137] Jafar Zarei, Poshtan Javad, and Poshtan Majjd, "Bearing fault detection in induction motor using pattern recognition techniques," in *IEEE 2nd international Power and Energy Conference* , 2008, pp. 749-753.

[138] Enrique Alexandre-Cortizo, Manuel Rosa-Zurera, and Lopez Ferreras, "Application of fisher linear discriminant analysis to speech/music classification," in *EUROCON* -

The International Conference on Computer as a Tool, 2005, pp. 1666-1669.

[139] S. Serhan, M. Abdullah, and M. Naura, "Face Recognition system: New classifier scheme based on the cluster-value and the K-nearest neighbourhood," *Advances in Modeling and Analysis*, pp. 1-20, 2006.

[140] Hossam Osman and Fahmy, Moustafa M., "On pattern classification using linear-output neural network classifiers," in *Proceedings of teh 36th Midwest Symposuim on Circuits and Systems*, 1992, pp. 1292-1295.

[141] Gopinath 0. Chandroth, Noel E. Sharkey Amanda J.C. Sharkey, "Acoustic Emission, Cylinder Pressure and Vibration: A Multisensor Approach t o Robust Fault Diagnosis," *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 6, no. 2000, 2002.

[142] Keming Wang, "Neural Network Approach to Vibration Feature Selection and Multiple Fault Detection for Mechanical Systems," *Proceedings of teh First International Conference on Innovative Coputing, Information and Control*, 2006.

[143] A.J.C. Sharkey and N.E. Sharkey, "Combining diverse neural nets ," *The Knowledge Engineering Review* , pp. 231-247 , 1997.

[144] A.J.C. Sharkey, N.E. Sharkey, and O.C Gopinath, "Diversity, Neural Nets and Safety Critical Applications ," *In L.F. Niklasson and M.B. Boden (Eds) Current Trends in Connectionism, Lawrence Erlbaum Associates, Hillsdale, NJ.*, pp. 165-178, 1995.

[145] Bo Li, G. Goddu, and Mo-Yuen Chow, "Detection of common motor bearing faults using frequency-domain vibration signals and a neural network based approach," *Proceedings of the American Control Conference*, vol. 4, 1998.

[146] R. Rengaswamy, D. Mylaraswamy, K.E. Arzen, and Venkatasubramani, "A comparison of model-based and neural network-based diagnostic methods," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 6, pp. 805-818, Dec 2001.

[147] R. Rengaswamy and V. Venkatasubramanian, "A syntactic pattern recognition approach for process monitoring and fault diagnosis," *Engineering Applications of Artificial Intelligence*, vol. 8, no. 1, pp. 35–51, 1995.

[148] Rolf Isermann, "Model-based fault detection and diagnosis- status and applications," *Annual reviews in control*, 2005.

[149] Suzuki, "A Boolean complementary fuzzy logic system," *Transactions of the institute of Electronics, Information and communication*, 1996.

[150] Zadeh L.A., "Fuzzy Sets," *Information and control*, vol. 8, pp. 338-353, 1965.

[151] R.SaravanaKumar, K. Vinoth Kumar, and Dr. K.K. Ray, "Fuzzy Logic based fault detection in induction machines using Lab view," *IJCSNS International Journal of Computer Science and Network Security*, vol. 9, no. 9, 2009.

[152] Dennice Gayme, Sunil Menon, Charles Ball, Dale Mukavetz, and Emmanuel Nwadiogbu, "Fault Diagnosis in a Gas Turbine Engines using Fuzzy Logic," *IEEE for Control Systems*, pp. 3756-3762, 2003.

[153] Seraphin C. Abou, Marian Stachowicz Manali Kulkarni, "Fault Detection in Hydraulic System Using Fuzzy Logic," *Proceedings of the World Congress on Engineering and Computer Science* , vol. II, 2009.

[154] Meijun Yang and Qiang Shen, "Reinforcing fuzzy rule-based diagnosis of turbo machines with case-based reasoning," *International Journal of Knowledge-Based*

and Intelligent Engineering Systems, 2008.

[155] S.K. Sin and R. J. Defigueiredo, "Fuzzy system design through fuzzy clustering and optimal predefuzzification," *2nd IEEE Fuzzy Syst., San Francisco, CA*, 1993.

[156] E. G. Laukonen, k.M. Passino, V. Krishnaswam, G.C. Luh, and G. Rizzoni, "Fault Detection and Isolation for an Experimental Internal Combustion Engine via Fuzzy Identification," *IEEE transactions*, 1995.

[157] Isermann Fussel, "Hierarchical Motor Diagnosis Utilizing Structural Knowledge and a Self-Learning Neuro-Fuzzy Scheme," *IEEE transactions*, 1998.

[158] S.R. Habibi and Burton R., "Parameter identification for a high-performance hydrostatic actuation system using the variable structure filter concept," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. v 129, n 2, pp. 229-235.

[159] S. A. Gadsden and S. R. Habibi, "Target tracking using the smooth variable structure filter," in *ASME Dynamic Systems and Control Conference*, 2009, DSCC2009.

[160] V. Jouppila, S.A. Gadsden, S.R. Habibi, G.M. Bone, and A. Ellman, "Sliding mode controller and filter applied to a pneumatic McKibben muscle actuator," *Proceedings of the ASME international Mechanical Engineering Congress and Exposition*, pp. 539-547, 2009.

[161] Wang Shu, Habibi Saeid, and Burton Richard, "A comparative study of a smooth variable structure filter and the extended kalman filter," *Transactions of the Canadian Society for Mechanical Engineering*, vol. v 32, n 3-4, pp. p 353-369, 2008.

[162] S. R. Habibi, "A Combined Variable Structure and Kalman Filtering Approach," *American Control Conference*, 2008.

[163] S.A. Gadsden, D. Dunne, S.R. Habibi, and T. Kirubarajan, "Comparison of extended and unscented Kalman, particle, and smooth variable structure filters on a bearing-only target tracking problem," *Signal and data processing of small targets* , August 2009.

[164] Mohammad A. Al-SHabi, "THE GENERAL TOEPLITZ/OBSERVABILITY SMOOTH VARIABLE STRUCTURE FILTER," McMaster University, Department of Mechanical Engineering , Hamiton, Canada, Ph.d Thesis 2011.

[165] Lutz Prechelt, "PROBEN1-A set of neural network benchmark problems and benchmarking rules," 1994.

[166] N. Morgan and H. Bourlard, "Generalization and Parameter Estimation in feedforward nets: Some Expirements," pp. 630-637, 1990.

[167] Dietmar Heinke and Fred Hamker, "Comparing Neural Networks: A Benchmark on Growing Neural Gas, Growing Cell Structures, and Fuzzy ARTMAP," *IEEE transactions on neual networks*, p. vol 9, November, 1998.

[168] k.L. Jones, I.G Lustig, and A.L. Kornhasuer, "Optimization techniques applied to neural networks: Line search implementation for back propagation ," *IJCNN International Joint Conference on Neural Networks.*, 1990.

[169] G.V. Reklaitis, A. Ravindran, and K.M. Ragsdell, *Engineering Optimization, Methods and Applications.*: Wiley, 1983.

[170] P.E. Gill, W. Murray, and M.H. Wright, "Practical Optimization," *Academic Press,*

New York, 1981.

[171] Russell D. Reed and J. Marks II Robert , *Neural Smithing: Supervised learning in feedforward artificial neural networks.*: Cambridge: The MIT Press, 1999.

[172] www.mathworks.com.

[173] David F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comput.*, vol. 24, pp. 647–656, July 1970.

[174] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs: NJ: Prentice-Hall, 1979.

[175] Trebaticky Peter and Jiri Pospichal, "Neural Network Training with Extended Kalman Filter Using Graphics Processing Unit," in *ICANN*, 2008, pp. 198-207.

[176] Youji Iiguni, Hideaki Sakai, and Hidekatsu Tokumaru, "A Real-Time Learning Algorithm for a Multilayered Neural Network Based on the Extended Kalman Filter," *IEEE transaction on signal processing*, vol. Vol. 40, No. 4., April 1992.

[177] L.A. Feldkamp and G.V. Puskorius, "'A signal processing framework based on dynamic networks with application to problems in adaptation, filtering and classification," in *IEEE*, 1998, pp. 2259-2277.

[178] Feldkamp L.A. and G.V. Puskorius, "Training controllers for robustness: multi-stream DEKF," in *IEEE international Conference on Neural Networks*, Orlando, 1994, pp. 2377-2382.

[179] Feldkamp L.A. and G.V. Puskorius, "Training of robust neurocontrollers," in *IEEE International Conference on Decision and Control*, Orlando, 1994, pp. 2754-2760.

[180] V. Utkin, *Sliding Modes in Control Optimization.*: New York: Springer-Verlag, 1992.

[181] S. R. Habibi, "The Smooth Variable Structure Filter," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 1026-1059, 2007.

[182] S. A. Gadsden and S. R. Habibi, "A New Form of the Smooth Variable Structure Filter with a Covariance Derivation," in *IEEE Conference on Decision and Control*, Atlanta, Georgia, 2010.

[183] S. A. Gadsden and S.R. Habibi, "An Optimal Smoothing Boundary Layer for the Smooth VAriable Structure Filter," *ASME Journal of dynamic systems, measuring, and control*, 2011.

[184] S. A. Gadsden and S. R. Habibi, "A New Form of the Smooth Variable Structure Filter with a Covariance Derivation," *IEEE Conference on Decision and Control*, 2010.

[185] K. O'Neil and Chen Y.-C, "Instability of Pseudinverse Acceleration Control of Redundant Mechanisms," *Proceedings of the IEEE International Conference on Robotocs and Automation* , April 2000.

[186] C. Melchiorri, C. Bonivento K. Doty, "A theory oof generalized inverses applied to robotics," *International Robotics Journal* , vol. 12, pp. 1-19, 1993.

[187] H. Lipkin and Pohl E. , "Enumeration of singular configuration for robotic manipulators," *journal of Mech. Design* , vol. 113, no. 3, pp. 272-279, 1991.

[188] W. H. Wolberg, ""Cancer dataset obtained from Wiliams H. Wolberg from University of Wisconsin Hospitsls, Madison"".

[189] G.E. Hinton, and R.J. Williams, D.E. Rumelhart and J. McClelland D.E Rumelhart, "Learning internal representations by error propagation," *Parallel Data Processing, Chapter 8*, vol. 1, pp. 318-362 , The M.I.T. Press, Cambridge, MA, 1986.

[190] K. Levenberg, "A Method for the Solution of Certain Problems in Least Squares," *Quart. Appl. Math.*, vol. 2, pp. 164-168, 1944.

[191] D. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM J. Appl. Math.*, vol. 11, pp. 431-441, 1963.

[192] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*.: NJ: Prentice-Hall, 1983.

[193] Mark Hudson Beale, Martin T. Hagan, and Howard B. Demuth, *Neural Network Toolbox™ 7 User Guide*.

[194] Bhaskar Ray, "Engine Fault Detection Using Wavelet Analysis," University of Windsor, Master's of Applied Science Thesis 2007.

[195] J. Wong, K. McDonald, and A. Palazoglu, "Classification of process trends based on fuzzified symbolic representation and hidden Markov models," *Journal of Process Control*, vol. 8, no. 5, pp. 395-408, 1998.

[196] V. I. Utkin, "Variable Structure Systems With Sliding Mode: A Survey," *IEEE Transactions on Automatic Control*, vol. 22, pp. 212-222, 1977.

[197] V. I. Utkin, *Sliding Mode and Their Application in Variable Structure Systems*, English Translation ed.: Mir Publication, 1978.

[198] W. Sun, A. Palazoglu, and J. A. Romagnoli, "Detecting abnormal process trends by

wavelet-domain Markov Models," *Aiche Journal*, vol. 49, pp. 140-150, 2003.

[199] J. J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[200] A.J.C. Sharkey, G.O Chandroth, and N.E. Sharkey, "A Multi-Net System for the Fault Diagnosis of a Diesel Engine," *Neural Computing and Applications*, pp. 152-160, 1999.

[201] Habibi S. R., "Comparison of Hydrostatic and Servovalve controlled Hydraulic Actuation Systems in robotics," *SAE Transactions, Journal of commercial Vehicles, Section 2*, pp. pp231-243, 2000.

[202] S. Postalcioglu, K. Erkan, and E.D. Bolat, "Discrete wavelet analysis based fault detection," *WSEAS Transactions on Systems*, vol. 5, pp. 2391-2397, 2006.

[203] H.B. Demuth, and M.H. Beale M.T. Hagan, *Neural Network Design*. ISBN 0-9717321-0-8 (available from John Stovall, john.stovall@colorado.edu, 303.492.3648).

[204] E.R. Leitzinger, "Development of In-process engine defect detection methods using NVH indicators," University of Windsor, Masters Thesis 2002.

[205] S. R. Habibi and R. Burton, "The Variable Structure Filter," *Journal of Dynamic Systems, Measurement, and Control (ASME)*, vol. 125, pp. 287-293, September 2003.

[206] S. R. Habibi, "The Smooth Variable Structure Filter," in *IEEE*, May 2007, pp. Vol. 95, No. 5.

[207] Stuart Geman, Elie Bienenstock, and Rene Doursat, "Neural Networks and the bias/variance dilemma," *Neural Computations*, 1992.

[208] Ronald A. Fisher, "The use of multiple measurements in taxonomic problems," *Annuals of Eugenics*, pp. 179-188, 1936.

[209] P. A. Cook, *Nonlinear Dynamical Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1986.

[210] Yuvin Adnarain Chinniah, "Fault Detection in the Electrohydrostatic Actuator Using the Extended Kalman filter," University of Saskatshwan, Saskatoon, Canada , 2004.

[211] G.O. Chandroth, A.J.C. Sharkey, and N.E. Sharkey, "Cylinder pressures and vibration in internal combustion engine condition monitoring," *Proceedings 'Comadem 99', Sunderland, UK,* July 1999.

[212] G. Chandroth, A.J.C. Sharkey, and N.E Sharkey, "Artificial Neural Nets and Cylinder Pressures in Diesel Engine Fault Diagnosis," *In Proceedings of INMARCO98 (Institute of Marine Engineers (India)), Mumbai, India.*, 1998.

[213] http://bellespics.eu/image/67b51c9b/.

[214] R.P., and Sejnowski, T.J. Gorman, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, pp. 75-89, 1988.

[215] B. Warner and M. Misra, "Understanding Neural Networks as Statistical Tools," *The American Statistician*, vol. 50, pp. 284-293, 1996.

[216] H. Altun, A. Bilgil, and B.C. Fidan, "Treatment of multi-dimensional data to enhance neural network estimators in regression problems," *Expert Systems with*

Applications, vol. 32, pp. 599–605, 2007.

[217] P.M. Frank Shneider H., "Observer-Based Supervision and Fault Detection in Robots," *IEEE*, vol. 4, no. 3, pp. 274-282, May 1996.

[218] Simon Haykin, *Kalman filtering and neural networks*, 0471369850471221546th ed., 2001.

[219] S. Wang, S. Habibi, and Burton R., "A smooth variable structure filter for state estimation," *Control and Intelligent Systems*, vol. v 35, n 4, pp. 386-393, 2007.

[220] Andrew Gadsden and Habibi Saeid, "Target tracking using the smooth variable structure filter," in *ASME Dynamic Systems and Control Conference*, 2009, DSCC2009.

[221] Saeid Habibi, "The Smooth Variable Structure Filter," in *IEEE*, May 2007, pp. Vol. 95, No. 5.

[222] Saeid Habibi, "A Combined Variable Structure and Kalman Filtering Approach ," *American Control Conference* , 2008.

[223] S. R. Habibi and R. Burton, "Parameter Identification for a High Performance Hydrostatic Actuation System using the Variable Structure Filter Concept," *ASME Journal of Dynamic Systems, Measurement, and Control*, 2007.