Machine Learning-based Strategies for Robust Fault Detection and Identification of Mobile Robots

Farzad Baghernezhad

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

December 2012

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis proposal prepared

By: **Farzad Baghernezhad**

Entitled: **Machine Learning-based Strategies for Robust Fault Detection and Identification of Mobile Robots**

and submitted in partial fulfilment of the requirements for the degree of

Master of Applied Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. Rabin Raut, Chair

_____ Dr. Youmin Zhang, External Examiner

_____ Dr. Wei-Ping Zhu, Examiner

_____ Dr. Khashayar Khorasani, Supervisor

Approved by _____

Chair of the Electrical and Computer Eng. Dept.

_____

Dean of Engineering

ABSTRACT

Machine Learning-based Strategies for Robust Fault Detection and Identification of Mobile Robots

Farzad Baghernezhad

Nowadays, process monitoring and fault diagnosis techniques are becoming a critical component of modern automatic control systems. One of the most crucial issues for the design of automatic control systems is reliability and dependability. Traditional ways to achieve these goals are through designing adaptive and robust controllers to eliminate any influence of faults on an output. Using these approaches, faults are managed but they could ultimately lead to failures; after which no controller could repair such effects. In order to minimize such damages, it is necessary to diagnose and rectify faults as soon as possible. In a fault detection system, residual generation is the first step in detecting faults, but residuals are not the only element of a dependable fault detection system. A fault detection system is reliable when an appropriate residual evaluation method is used along with a suitable residual generation technique.

The problem of fault detection and identification in a nonlinear system with applications to mobile robots is addressed in this thesis. For this purpose, first a new simulator for mobile robots containing kinematic and dynamic equations of a mobile robot and its actuators is designed. To detect faults in the system, a linear velocity of the mobile robot is chosen and modeled with computationally intelligent techniques. Locally linear models (LLM) as a neuro-fuzzy technique and radial basis function (RBF) as a powerful neural

network are used to estimate the linear velocity of a mobile robot and generate residuals by comparing these with the system measurements. Subsequently, residuals are evaluated by using fixed and adaptive threshold bands. Adaptive threshold bands are generated using locally model thresholds (LMT) and model error modeling (MEM) technique in order to reduce the fault detection delay and false alarms. Finally, fault identification of a mobile robot by using multiple model technique is presented with the two proposed methods of modeling and threshold generation. The fault identification task consists of determining the occurrence of the fault as well as its location and magnitude. This is accomplished on four different types of faults with different magnitudes that are divided in ten different classes. For each scenario, simulation results are presented to demonstrate and illustrate the advantages and disadvantages of each methodology.

The main contributions of this thesis can be stated as follows: (a) the development and design of a new fault diagnosis method and a residual evaluation scheme by using an adaptive threshold band that is accomplished by using locally linear models of the system, (b) development of a fault detection approach based on computational intelligent algorithms for mobile robots using the MEM algorithm to generate adaptive threshold bands. (c) development of two fault identification approaches based on the concept of multiple models for a mobile robot, and (d) the resulting improvements of the proposed adaptive threshold bands are shown through extensive simulation results.

TO MY PARENTS

for their love and supports

ACKNOWLEDGEMENTS

First and foremost, I would like to express my most sincere gratitude to my supervisor, Dr. Khashayar Khorasani for his continuous patience, support and guidance in the development of this thesis.

Also, I would like to thank my committee members, Dr. Youmin Zhang, Dr. Wei-Ping Zhu and Dr. Rabin Raut for devoting their valuable time in guiding and evaluating my work.

Finally, I would like to give my special thanks to my parents whose love enabled me to complete this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IFAC	International federation of automatic control
FD	Fault detection
FI	Fault Isolation
FDI	Fault detection and Isolation
FAI	Fault analysis or identification
FR	Fault recovery
AC	Accuracy
TP	True positive rate
FP	False positive rate
TN	True negative rate
FN	False negative rate
P	Precision
PCA	Principal components analysis
MIMO	multi-input multi-output
LLM	Locally linear models
RBF	Radial basis function
LoLiMoT	Local linear model tree
LS	Least squares
WLS	Weighted least squares
ANN	Artificial neural network
NRBF	Normalized radial basis functions
PD	Proportional derivative
LMT	Local model thresholds
MEM	Model error modeling

Chapter 1

Introduction

1.1 Motivation

A fault is an abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function [1]. Every system in the world is vulnerable to faults. If faults are not determined and resolved in proper time, they can be amplified and lead to failures; after which no controller could repair such effects. In order to minimize such damages, it is necessary to diagnose and rectify faults on time.

Faults typically have a low probability of occurrence, but the costs associated with them are high. For example, a small crack in the wheel of a train can break the entire wheel and cause derailing of the train, leading to a potentially big disaster. It could be avoided if the crack in the wheel could be found on time. This is a critical issue for all systems such as chemical processes, nuclear plants, power distribution and unmanned systems. Due to such importance, process monitoring and fault diagnosis are becoming critical components of modern automatic control systems [2]. For the last two decades, researchers from the industrial and academic fields have investigated and developed a number of fault diagnosis methods for different systems such as networking systems [3], building systems [4, 5] and mechanical systems [6].

In this thesis, the problem of fault detection of mobile robots is addressed. Mobile robots, in contrast to robot manipulators, are robots with the ability to move. They are important members of unmanned vehicles. Such vehicles can be smaller and have lower weight due to the fact that no human is involved directly. Moreover, there is no cost related to the human involvement, such as teaching, insurance, man-made mistakes, etc. Beyond these merits, the main advantage that is noted for unmanned vehicles is their performance in dirty and dangerous missions where humans cannot access easily. The ability of these vehicles to surpass in hazardous missions has caused them to be adopted in many fields.

One of the major characteristics of autonomous systems, in general, and mobile robots in particular, is that they should possess the capability of fault diagnosis. In other words, complex systems need to be equipped with intelligent mechanisms designed for fast detection and isolation of faults and early detection of performance degradations for cost effectiveness and time maintenance. As a matter of fact, fault diagnosis procedures not only contribute to the autonomy of these systems but also change the maintenance philosophy that has been used for several years.

For instance, when mobile robots are used for planetary explorations in space missions, they should be able to operate in a long period of time without intervention from the central command and control station that are based on earth. In other words, communication with a central station is limited to a short period of time even during fault free and healthy conditions. Moreover, due to the long roundtrip communication delays, the capability of a central station to respond to emergencies and stimuli is very limited and real-time monitoring is almost impossible. Consequently, the major subsystems of a mobile robot, such as power subsystems, the driving subsystems, the steering subsystems, communication and sensors should be as autonomous as possible. In fact, autonomy plays a vital role in the accomplishment of the mobile robot missions in such applications. In recent years there has been intensive research work on fault diagnosis of a variety of components and subcomponents of wheeled mobile robots, such as motors [7], gears [8], tires

[9], suspension [10], sensors [11], etc.

1.2 Literature Review on Fault Diagnosis

On-time fault detection and isolation (FDI) of industrial systems reduces productivity loss and assists prevention of unusual event progress. In an FDI process, the objective is to have a system which detects faults as quickly as possible with minimum false alarms. Early detection and isolation of faults allows the application of recovery actions leading to maintaining the process operation and avoiding system shutdown, breakdowns and even catastrophes involving human fatalities and material damage. Traditional approaches to fault diagnosis are based on the hardware redundancy methods which use multiple components such as sensors and actuators to measure or control a particular value, which does impose extra equipment, space and maintenance cost [12]. In past years different alternative methods have been proposed for this purpose. A complete literature review on various FDI methods is presented in [13]. As seen in Figure 1.1, which shows a general on-line FDI system, first phase in detecting faults is residual generation which can be defined as the difference between the expected variable and the measured one. Generating rich residual plays an important role in the FDI process and this has attracted a large body of literature ranging from analytical methods to artificial intelligence approaches [14]. The residual should be sufficiently large in the presence of faults and small or negligible in the absence of faults. In fact, noise, disturbances and modeling errors of the system should not effect the residual [16]. A control system that is able to tackle such a challenging problem is shown in Figure 1.2.

The main objective of this chapter is to present an overview to the fault diagnosis and its most popular approaches.

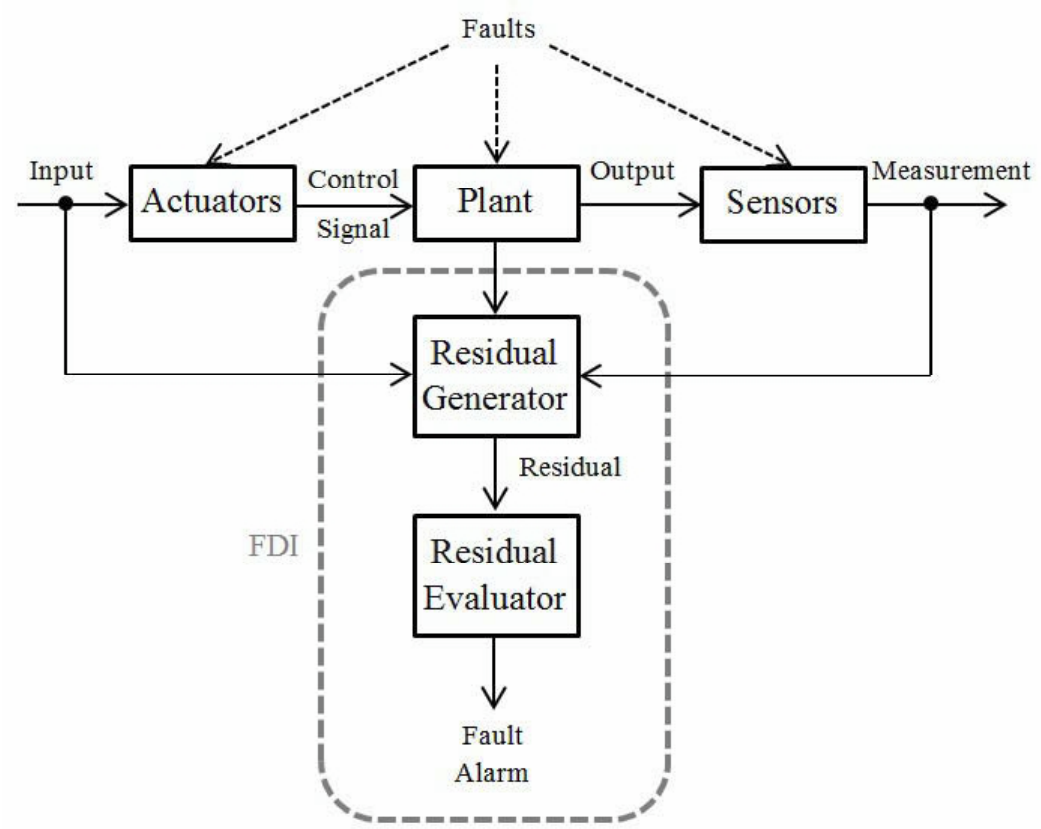

Figure 1.1: An FDI system [15].

1.2.1 Basic Concepts of Fault Diagnosis Techniques

In order to understand the concept of fault diagnosis, first the most important and general definitions and nomenclature in this field are introduced. The presented definitions might not be used identically the same in all references, but it is widely accepted by the experts in this field. The following definitions are based on the proposed terminology by the International Federation of Automatic Control (IFAC) published in variety of references [1, 2, 13, 17, 18], etc.

States and Signals

Fault: Unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable, usual or standard condition.

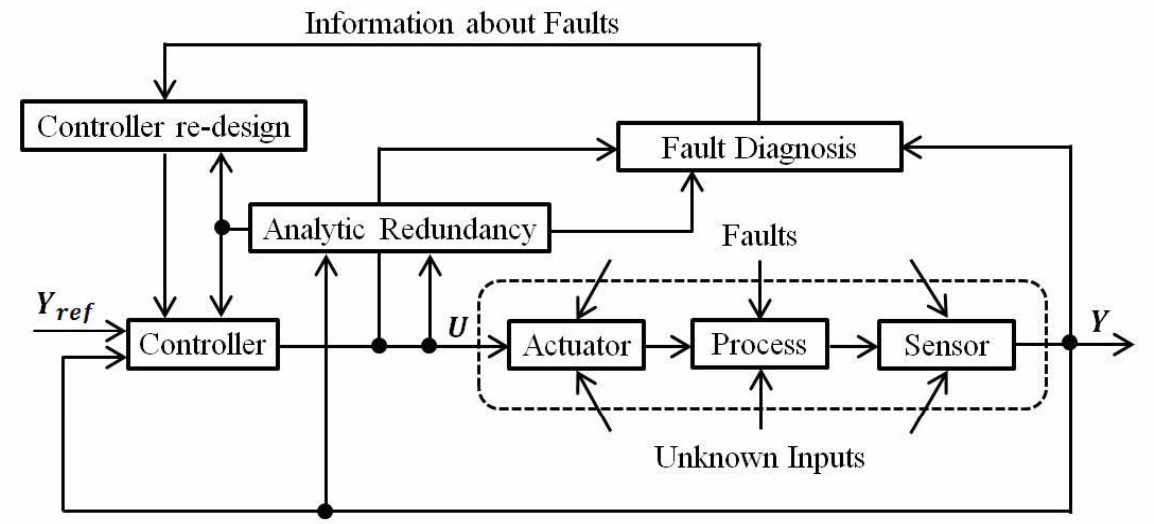

Figure 1.2: A closed-loop control system [2].

Failure: A permanent interruption of a system's ability to perform a required function under specified operating conditions.

Error: A deviation between a measured or computed value of an output variable and its true or estimated one.

Disturbance: An unknown and uncontrolled input.

Residual: A fault indicator, based on the deviation between the measurements and the model equation based computations.

Functions

Fault detection (FD): Determination of the faults that are present in a system and the time of the detection.

Fault isolation (FI): localization (classification) of faults.

Fault analysis or identification (FAI): determination of the type and magnitude of the fault.

Fault recovery (FR): refers to the reconfiguration of the system using healthy or the available components and actuators/sensors.

Monitoring: A continuous real-time task of determining the conditions of a physical system and recognizing and indicating anomalies of its behavior.

Supervision: Monitoring a physical system and taking appropriate actions to maintain the operation in case of faults.

Prognosis: Prognosis focuses on predicting the condition of an engineered system or equipment at times in the future. As with FAI, prognosis is used along with evaluations of impacts to make operation and maintenance decisions. Use of prognosis enables transition from maintenance based on current conditions of engineered systems and equipment (condition-based maintenance) to predictive maintenance. Predictive maintenance is based on anticipated future conditions of the equipment, its remaining useful time before failure (or time before reaching an unacceptable level of performance), the rate of degradation, and the nature of the failure if it were to occur.

Time Dependency of Faults (Figure 1.3)

Permanent or Abrupt Fault: Faults modeled by a stepwise function. It represents a bias in the monitored signal.

Incipient Fault: Faults modeled by ramp signals. It represents drift of the monitored signal.

Intermittent Fault: Faults modeled by a combination of impulses with different amplitudes.

Fault Detection Parameters

Usually fault detection performance is reported using detection delay and confusion matrix. Detection delay, also called detection time, is the time from the occurrence of the fault to the fault alarm time and the confusion matrix reports the features of the classification [31]. When only fault detection is considered, generally the following parameters are calculated, namely

a denotes the number of healthy data correctly identified.

b denotes the number of healthy data incorrectly detected as fault.

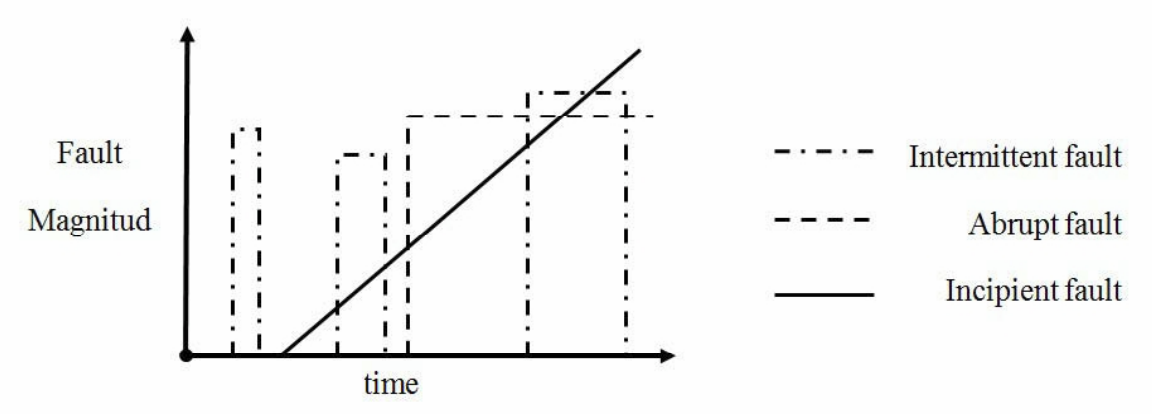

Figure 1.3: Time dependency of a fault [2].

c denotes the number of faulty data incorrectly identified as healthy.

d denotes the number of faulty data correctly detected.

Accuracy (AC) is the proportion of the total number of predictions that were correct. It is determined by using the equation

$$AC = \frac{a+d}{a+b+c+d} \tag{1.1}$$

Recall or true positive rate (TP) is the proportion of positive cases that were correctly identified, as calculated by using the equation

$$TP = \frac{d}{c+d} \tag{1.2}$$

False positive rate (FP) is the proportion of negatives cases that were incorrectly classified as positive, as calculated by using the equation

$$FP = \frac{b}{a+b} \tag{1.3}$$

True negative rate (TN) is defined as the proportion of negative cases that were classified correctly, as calculated by using the equation

$$TN = \frac{a}{a+b} \tag{1.4}$$

False negative rate (FN) is the proportion of positives cases that were incorrectly classified as negative, as calculated by using the equation

$$FN = \frac{c}{c+d} \tag{1.5}$$

Precision (P) is the proportion of the predicted positive cases that were correct, as calculated by using the equation

$$P = \frac{d}{b+d} \tag{1.6}$$

Another way of reporting the classification performance is by using a table to show that each data has classified in which class. In fact, confusion matrix shows the predicted and actual classifications. This matrix is usually used when the performance of fault isolation and identification is needed to be reported. For example, if N classes of healthy and faulty data have to be classified (identified or isolated), a confusion matrix is a matrix of N rows which are real classes and N columns which are predicted classes. Each element of the matrix shows how many data of the i^{th} class has been detected in the j^{th} class $(i, j = 1, 2, ..., N)$.

Fault Detection with Limit Checking

Before taking a precise look at the different methods of diagnosing faults, limit checking technique is introduced. The most frequent and simple technique for fault detection is the limit checking of a measured variable $Y(t)$ [1]. In this case, the measured variables of a process are monitored and checked if their absolute values exceed certain limit values. Generally, two limit values, called thresholds, are presented: a maximal value Y_{max} and a minimal value Y_{min}. A normal state is when

$$Y_{min} < Y(t) < Y_{max} \tag{1.7}$$

The above implies that there is no fault in the process if the monitored variable stays within a certain tolerance zone. When $Y(t)$ exceeds from one of threshold bands, fault is detected as shown in Figure 1.4. Threshold bands come from the priori information about

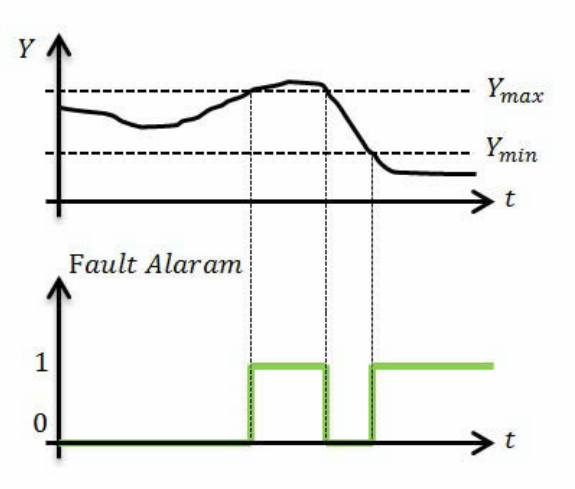

Figure 1.4: Fault detection by using limit checking [1]

the system and measured variables. They should be as small to detect faults as quickly as possible and large enough to minimize false alarms. As a result, a trade-off between a tight band and a wide band of thresholds exists.

1.2.2 Fault Detection Methods

As described before, the main idea of detecting faults comes from comparing the information gathered from measured components of the system with the prior knowledge about it. Fault detection methods are categorized typically into two major classes based on how the priori knowledge is achieved:

Model-based methods which use mathematical model of the system and *model free based (data or signal [13]/learning/process history[19]-based)* methods which do not use the mathematical model of the system. In fact, model-based approaches are based on an *a priori* knowledge developed from an understanding of the process, while model free-based approaches are designed based on an *a priori* knowledge extracted from past experiences of the process.

The two methods can be classified into two subsets with quantitative and qualitative methods as illustrated in Figure 1.5. The quantitative method uses a mathematical functional relation between inputs and outputs. On the other hand, it is considered as a qualitative method if the relation between faults and failures is expressed in terms of qualitative functions using different units of the process [19].

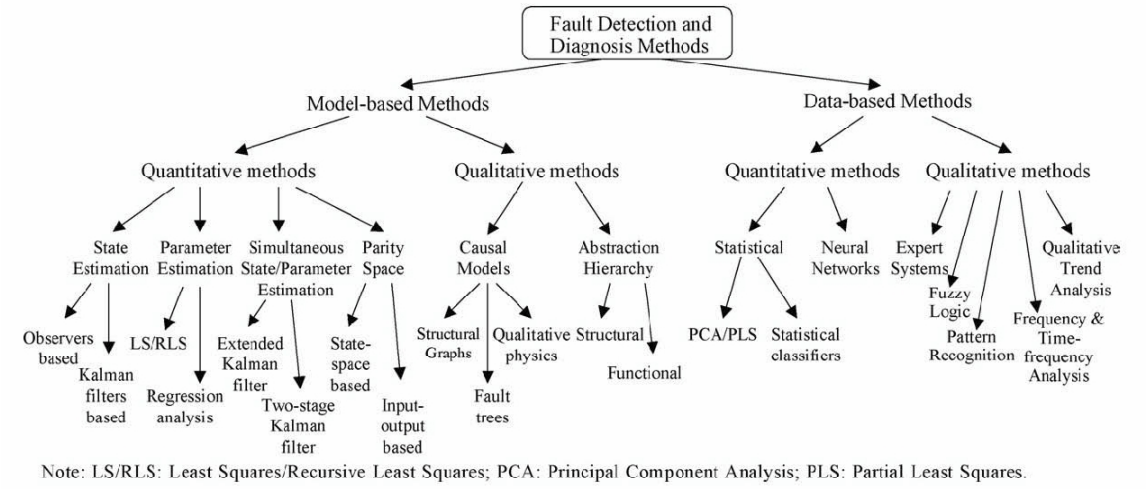

Figure 1.5: Classification of FD methods [13].

Model-Based Fault Diagnosis Techniques

As described before, if the priori information about the measured variable is generated using mathematical model of the system, the method is known as model-based fault diagnosis technique. It is pointed out in [17] that this method is a relatively young research field and its development is rapid due to currently receiving considerable attention. Different approaches for fault detection using mathematical models have been developed in past years [17, 20, 21], while the main themes are classified into three methods [12, 13, 17]:

Observer-based methods [17, 22, 23]: In observer or filter-based approaches, the outputs of the system are estimated from measurements (or subset of measurements) by

using either Kalman filters in a stochastic setting or Leunberger observers in the deterministic settings. A residual is then defined as the weighted output estimation error. The observer-based FDI approach has become as one of the most common approaches in this domain due to the increasing popularity of state-space models as well as the wide usage of observers in modern control theory and applications [12].

Parity space methods[17, 24, 25]: In this approach residual signals or parity vectors are generated based on consistency checks on system input and output within a time window. Using this method, the FDI system designer is not required to have rich knowledge of the advanced control theory. In recent years, a great number of FDI methods and ideas have been first presented in the parity space framework and later extended to the observer-based framework [17].

Parameter identification based methods [17, 26] : This method is based on system identification techniques. This approach uses the fact that faults are reflected in the physical system parameters such as friction, mass, resistance, etc. The basic idea is on-line parameter estimation of the actual system and comparing result with the parameters of the reference model that is obtained initially under the fault-free condition [12].

Model-Free Fault Diagnosis Techniques

Model-free fault diagnosis techniques are applied to the system when priori knowledge is extracted from past experiences of the process without using mathematical model of the system. Most commonly used techniques are based on statistical methods such as principal components analysis (PCA) and computational intelligence methods. The methods used in this thesis are categorized into the computational intelligence methods that are discussed in more detail below.

Computational Intelligence Methodologies in FDI

In this domain, faults of the system are detected by using a representation of the model of the system which is trained by computational intelligence methodologies and without requiring mathematical equations of the system. Typically, in the detection phase, normal healthy behavior of the system is modeled. A comparison between the output of the model and the output of the system is made, in which residual signals are generated. In the isolation phase, learned models can be utilized to achieve the classification of the residuals into appropriate categories of faults. Certain fault diagnosis systems utilize learned models for both detection and isolation, while there are also hybrid systems which utilize them for either detection or isolation phases only. Below, we take a brief look at three different kinds of learning-based modeling techniques.

Neural Network Applications: Neural networks have been used extensively in mobile robots as controllers [45] and also as modeling tools for fault diagnosis purposes [30, 46, 47]. They can be applied for both detection and isolation of faults in systems [46]. Neural networks model a system after the training process is completed which is done by learning from the existing data of the systems. At the simplest level, a single neuron produces an output for a given set of inputs. The neuron's output plays the role of input for the next layer's neurons until the output layer which produces the outputs of the system. By proper adjusting the network parameters; weights, biases, functions, number of neurons and layers, it has been shown that the neural networks are general (universal) function approximator [28, 29, 43].

Fuzzy Logic Applications: Zadeh first proposed fuzzy set theory in 1965 [48]. In 1973, he introduced the concept of linguistic variables and proposed the use of If-Then rules to represent human knowledge [49]. In 1976, Mamdani proposed a linguistic tool to build the fuzzy model of a system [50]. He proposed to model the system behavior by using if-then rules connecting linguistic terms that captured the intuitive understanding of the available signals by human subjects. In recent years, fuzzy logic has been successfully

applied to different fault detection and diagnosis technical processes [51]. Also, fuzzy logic is very often used to perform fault isolation tasks, where the relations between residuals and the faulty states of the monitored systems are expressed by a set of if-then rules [27, 52].

Neuro-Fuzzy Applications: Neuro-fuzzy networks are fuzzy models that are not solely designed by expert knowledge (fuzzy models characteristic) but are at least partly learned from data [43]. A neuro-fuzzy model is a powerful combination of neural networks and fuzzy logic techniques. This method is used for both fault detection via modeling, and fault isolation via classification of nonlinear systems [53].

Two categories of combinations between neural networks and fuzzy systems are proposed in [54]. There are systems where the neural networks represent the basic methodology and fuzzy logic the secondary one. In other methods, which is also used in this thesis, fuzzy logic represents the basic methodology and neural networks as the secondary one. In these systems a set of fuzzy rules are put in the form of a neural network in order to make use of the learning, adaptation and parallelism capabilities provided by neural networks [53].

1.2.3 Threshold Generation

As illustrated in Figure 1.1, apart from residual generation through modeling of the system in the fault-free condition, another important issue for FDI is residual evaluation or decision making. The decision-making task in diagnostic problems starts with observation of behavior recognized as a deviation from which is expected or desired and establishes some hypothesis about the cause of the fault [53]. In this step, optimal selection of the threshold band is the most important aspect [33].

Typical logical schemes associated with FDI algorithms depend on a set of constant or fixed thresholds [34]. If a fixed threshold band is chosen, its value should be sufficiently small so that faults can be detected at their early stages and also the rate of false alarms

is minimized. In this case, lower and upper bands are constructed which determine the minimum and the maximum values that each residual signal can reach under fault free operating condition [35, 36].

The other methodology that is used for threshold generation is by using adaptive threshold band. In this approach threshold values change in regards to the data obtained from the system and the residual generation aspect. Several methods illustrate that adaptive threshold bands have better performance than fixed ones. In [34] these two methods are compared on FDI of actuator failures of an aircraft. An adaptive threshold generation based on support vector machines for fault detection is presented in [37]. Another adaptive threshold generation technique based on eigen decomposition is discussed in [16]. By using fuzzy rules, development of adaptive fuzzy thresholds is also an attractive idea [38]-[40].

In this thesis by using locally linear models (LLM) of the system, which belongs to the neuro fuzzy modeling method, a new idea of adaptive threshold generation is proposed to improve the FDI process. An alternative adaptive threshold band is also presented by using the model error modeling technique [55] to improve the fault detection performance.

1.2.4 Fault Identification

Several methods exist to determine the type and the magnitude of a fault. One of the most common methods is by using multiple models. In this method, multiple models are designed or trained to estimate the output of the system under different faulty cases with different severities. By comparing the measured output with the estimated outputs different residuals are obtained which can determine the type of the fault and its magnitude. The study on the residuals leads one to a classifier which can be a simple logic or a neural network.

Different methods can be used to both model faults and classify them. In [41], a bank of parallel Kalman filters, each with a different internal fault model, is utilized to

model different faults in the system and then the faults are identified with a hypothesis testing computation. A bank of Kalman filters is used in [47] to model various faults in a mobile robot. A neural network is then used as a classifier to identify the faults. In [42], neural networks are used as both different fault models and classifiers.

1.3 Problem Statements and Thesis Contributions

In this thesis, a nonholonomic two wheeled mobile robot having nonlinear dynamics is controlled to follow a predefined trajectory in the absence of obstacles. In the trajectory plan, certain fault scenarios are implemented in different parts of the system. The faults are chosen as the most common mobile robot faults that are addressed in previous studies [30, 35, 86, 89, 90] and [97]. The purpose of this thesis is to design a reliable fault diagnosis system to detect and identify faults as soon as possible with minimum false alarms.

To propose a solution for the stated problem, first a mobile robot simulator is developed. The simulator contains the kinematic and dynamic equations of a two wheeled mobile robot controlled by a multi-input multi-output (MIMO) controller. By using the data that is gathered from the simulator, linear velocity of the system is modeled with two learning-based methods of system identification; namely the locally linear neuro fuzzy model trained by a locally linear model tree (LoLiMoT) algorithm and a RBF neural network. Subsequently, various faults that are implemented in the system are detected by limit checking of the modeled output. In this study, to decrease false alarms and detection time, adaptive threshold bands are proposed. For the first method, an adaptive threshold band is generated by using locally linear models of the system by attributing a threshold value to each model which are named locally model thresholds (LMT). Another adaptive threshold band is generated by using model error modeling (MEM) technique. The adaptive threshold performances on fault detection of mobile robots are compared with the fixed threshold. Also, for the RBF neural network, the fault detection performance

of the adaptive threshold bands generated by the MEM algorithm is compared with the fixed threshold band. In the last part, different faults of the system are identified by using multiple faulty models of the system and two of best methods for the fault detection.

1.4 Thesis Outline

Chapter 2 describes the developed fault detection schemes. To achieve this goal, two different computational intelligence methodologies are explained to model nonlinear systems in order to detect and identify system faults.

Chapter 3 presents a complete model for a two wheeled mobile robot. The kinematic and dynamic equations of a mobile robot and the permanent DC motors as actuators, are provided to develop a complete model of a mobile robot. This model is controlled by proportional and derivative control gains.

Chapter 4 first introduces possible faults in the system and reviews previous efforts on fault detection of mobile robots. The implemented faults in the system are listed. By using modeling methods that are presented in Chapter 2, three robust fault detection schemes are proposed. Finally, by using simulation results, the strengths and weaknesses of the proposed fault detection techniques are discussed.

Chapter 5 describes the fault identification of the system by using a multiple model algorithm. In this chapter, various fault scenarios are simulated and the performance of the proposed modeling methods on fault identification of the mobile robots are evaluated and compared.

Chapter 6 contains a brief summary of the main topics and the most important contributions of the thesis. Finally, some suggestions for future studies on the fault diagnosis of mobile robots are made.

Chapter 2

Background Information

2.1 Introduction

Modeling and identification of nonlinear systems is a very challenging problem. Proposing a structure able to describe every possible system efficiently is very hard. Therefore, different modeling approaches have been presented to respond to the needs in each domain. Using a suitable method is one of the modeling and identification issues. In this thesis, to model a mobile robot platform the external dynamics approach with a nonlinear static approximator is used as an appropriate structure for modeling the system.

This chapter introduces an overview of the background material related to the system modeling methods that are used in this study. At first, the proposed fault detection methodology is briefly introduced to provide the justification on modeling the system (the details about the detection logic will be explained in Chapter 4). After that, two static nonlinear system identification methods are explained to model the output of the system. The first method is a locally linear neuro fuzzy model which uses local linear models (LLM) and the second method uses radial basis function (RBF) neural networks. In this study, to estimate and represent a dynamic system using static tools, justifications on the external dynamics approach is used [43].

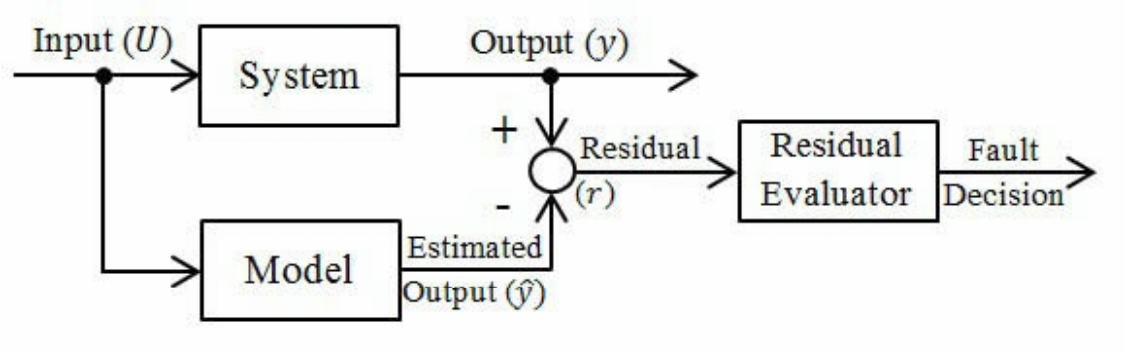

Figure 2.1: Simple structure of the proposed method for fault detection.

2.2 Proposed Fault Detection Methodology

The simple scheme that is used as a methodology for detecting faults in this thesis is illustrated in Figure 2.1. In this figure, a model is designed to estimate y, the output of a nonlinear dynamical system. Models should be accurate to maintain residuals in a bounded area under the fault free situations. After training an appropriate model, faults in the system are detected by using limit checking of the modeled output. By comparing the measured output of the system y with the estimated one $\hat{y}$, the residuals r are generated according to:

$$r = \hat{y} - y \tag{2.1}$$

When the system is affected by faults, the estimation error or the magnitude of the residual increases. Hence, it can be interpreted as an appropriate fault signature. Residuals are evaluated by comparing their values with either adaptive or fixed threshold bands.

2.3 Modeling Approach

As described before, the first step in detecting faults is modeling the system. For this purpose, two nonlinear static modeling methods are used to estimate linear velocity of the mobile robot. The training structure of the models is shown in Figure 2.2. The models use

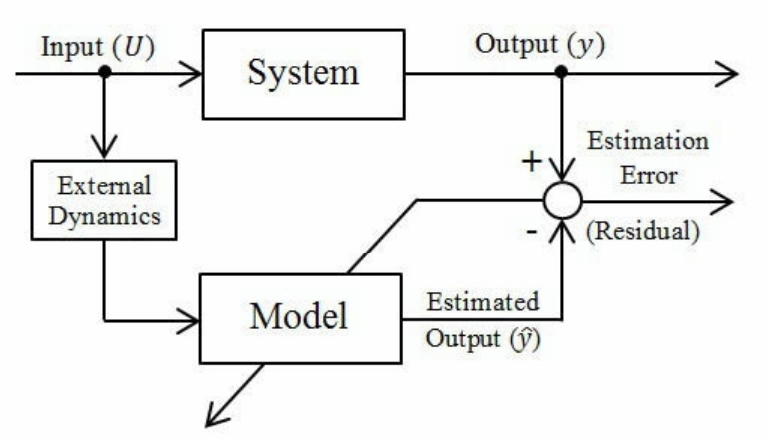

Figure 2.2: Training process for representing the model.

inputs of the system and their past samples to estimate the output. The inputs and output of the system at the time instant k are defined as $U(k)$ and $y(k)$, respectively. The objective of modeling the system leads to approximation of the following function

$$y(k) = f(x(k)), \;\; where\, x(k) = [U(k) \;\; U(k-1) \;\; ... \;\; U(k-m)]^T \tag{2.2}$$

In order to determine the appropriate structure of the model in addition to optimizing the parameters, the dynamic order m has to be estimated. To add dynamics to a static learning approximator, the external dynamics approach is used [43]. The external dynamics without output feedback is applied here which will be explained in next subsection.

2.3.1 External Dynamics

The modeling methods used in this study belong to static methods of system identification. In recent years, several approaches have been proposed [56]-[63] to present dynamics to a static learning approximator which can be mainly classified into two categories. The first method, known as the external dynamic, is to feed the network with current and delayed values of the system inputs and outputs [43]. The external dynamic strategy is the most commonly used method to model and identify nonlinear dynamical systems [56]-[58].

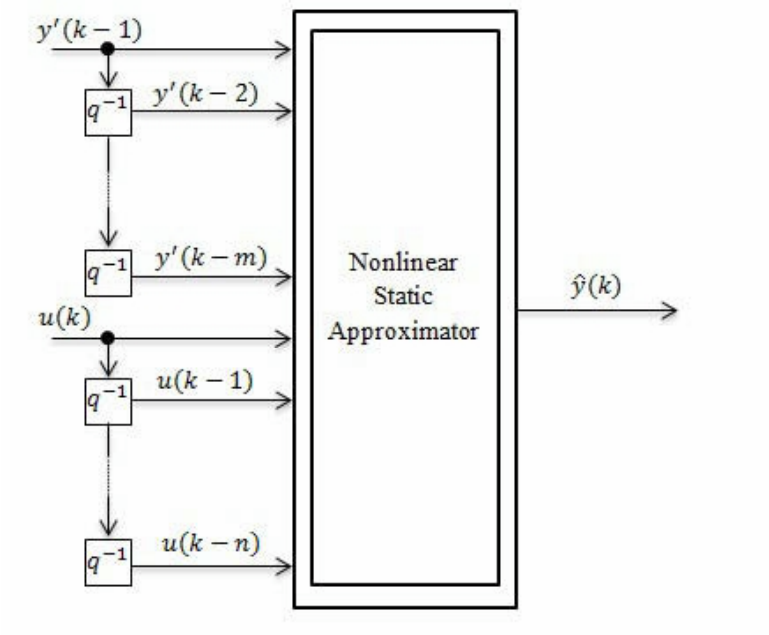

Figure 2.3: The external dynamics approach [43].

This method uses the fact that nonlinear dynamic model can be divided into two parts: a nonlinear static approximator and an external dynamic filter bank as shown in Figure 2.3. The filters are chosen as simple time delays, which are denoted as tapped-delay lines. Several authors have applied this method to different applications and good references for this category are [43] and [57].

In contrast to models with external dynamics, an alternative approach uses internal dynamics. In fact, models with internal dynamics are based on the extension of the static models with internal memory by utilizing dynamic neurons [43]. Therefore, past inputs and outputs are not used at the model input which reduces the input space dimensionality. In past years, several dynamic neuron structures have been reported in the literature [59]-[63]. In [63], dynamic neural network structures are presented to demonstrate that they are capable of representing the relevant class of dynamic nonlinear systems.

Due to the existence of several means for choosing the architecture and the training algorithm of the external and the internal dynamic methods, their comparison is not

straight forward. However, some major differences between them are obvious. The external dynamic approach is less robust and more sensitive with respect to the information about the system order m. In other words, the internal dynamics approach does not necessarily require an assumption on the order of the system. The order of feedback delays in the internal dynamics approach should be determined by using nonlinear parameter optimization methods. This fact imposes a great computational burden in the training phase. The missing interpretability is another drawback of the internal dynamics, however, interpretability of the external dynamics approach depends on the chosen network architecture [43]. When a high order and a multivariable system needs to be modeled, the internal dynamics approach has the advantage of lower curse of dimensionality due to the fact that the network is fed only with the actual input and outputs.

In conclusion, the external dynamics approach needs more storage while the dynamic neurons approach has more computational complexity for training and running the identifier. In this thesis, the external dynamics is chosen due to the fact that by using dynamic neurons one needs to estimate the related parameters and a wrong choice of parameters might cause internal instability. Below, two different methods for the external dynamics are explained.

Models with Output Feedback. One way is to use output feedback as well as input at the input of the external dynamics. Regarding the type of the model, either system output $y(k-1)$ or model output $\hat{y}(k-1)$ can be used as the input of the external dynamics. If $y'(k-1)$ in Figure 2.3 is the measured output of the system ($y(k-1)$), the configuration is known as prediction with a series-parallel model. Otherwise, if $y'(k-1)$ is estimated output of the system ($\hat{y}$), the configuration is known as simulation with a parallel model. Prediction is typically used when the current state of the system is measurable as in weather forecasting and stock market predictions. When the system output cannot be measured or when a sensor is replaced by a model, the simulation is required. Also, for FDI purposes, usually the system output is compared with the simulated model output [43].

One issue for models with output feedback is to find the dynamic order m which is crucial for an acceptable performance of the method. In fact, there is no efficient method for determining the dynamic order. Another disadvantage that is encountered in output feedback methods in [43] is the lack of general proof of stability.

Models without Output Feedback. In this case, the nonlinear static model uses only the previous or the filtered inputs. The price to be paid for the not having feedback is that the dynamic order n has to be chosen very large to describe the system dynamics properly which does limit the application of this method. The above drawback can be reduced largely by incorporating priori knowledge about the system dynamics into the linear filters [43].

In this thesis, the external dynamics approach without output feedback is used since the system is found to be a low order dynamical system. The estimators are able to model the system by using present and past inputs. This method needs less memory than the case which uses present and past outputs as well as the inputs.

2.3.2 Supervised Learning

Computational intelligence models are complex functions usually composed of the superposition of simpler functions. Finding the appropriate structure as well as good parameters for the model is known as training. The most usual method of training is the supervised learning which assumes that there is a set of (k) training samples $\{(U_n, y_n)\}_{n=1}^{k}$, each consisting of an input vector and a measured or desired output or output vector. Learning involves sweeping through the training set, gradually adjusting the parameters and structure of the model so that the output of the model gets sufficiently closer to the real output. In fact, data samples act as teacher or training sets. The detail on training each model is described separately in the next subsection, but before that the different sets of data that are used to train and validate models are explained.

Data Selection for the Supervised Learning

Generally, when there is no limitation on the data generation, three sets of data are used in the process of training and testing the model. Each set of data should be chosen wisely in order to contain all of the characteristics of the system. First set is the training data, which is used to adjust the parameters. Using input and output of the training data and proper optimization algorithm, weights, biases and function parameters are trained in order to minimize the sum square error or the cost function.

Second set of data is the test data, which is used to adjust the structure of the network. Besides training the model with training data, test data helps to find its structure. By comparing the system's output with the estimated output of the test data, model's performance is calculated for each parameter.

For example, consider a number of neurons in the hidden layer of an RBF neural network or in a LLM. An important characteristic of these models is the number of neurons. If an inadequate number of neurons are used, the network will be unable to model complex data, and the resulting fit will be poor. On the other hand, using too many neurons might decrease the estimation error in the training data, but it increases the training time excessively and worse, the network may over fit the data. Overfitted systems, model random noise of the data. The result is that the model fits the training data extremely well, but it generalizes poorly to new and unseen data. In order to avoid these problems, the number of neurons should be optimized. Therefore, by increasing the number of neurons one by one (the increasing rate depends on the sensitivity of the network to the parameter), the estimation sum square error for the test data will start to increase (refer to Figure 2.4). The last neuron value before increasing the sum square error of the test data is the optimized number of neurons. This method can be used for obtaining the numerical parameters such as the number of training, the layers and the neurons. In FDI algorithms, typically residual of the test data is used to generate the threshold bands.

The third set of data is the validation data which does not play a role in the training

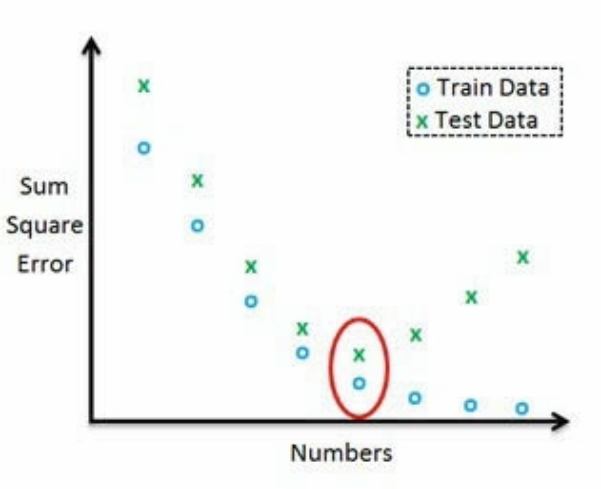

Figure 2.4: The optimized number of neurons is found when the sum square error of the test data is minimized.

of the model. This set of data is used to show the performance of the model when new data is applied. Below the models that are used for the fault detection problem are described in more detail.

2.4 Locally Linear Models (LLM)

A parallel superposition of partial (local) models by localizing weights is designated as a local model network [64]. This kind of modeling has been developed under different names and in different fields [64]. The main idea is based on approximation of the model of a nonlinear function with piece-wise local models (neurons). In most cases, partial models are chosen as linear models and a switching algorithm among models is applied by using fuzzy rules which produces locally linear neuro-fuzzy models or the LLM for short [43]. The most important factor for the success of local linear models is the division strategy of the original complex problem [56]. A complex modeling problem is divided into a number of smaller and therefore simpler subproblems, which are almost solved independently by identifying simple, e.g., linear models [43].

The schematic of the LLM structure with M neurons is shown in Figure 2.5. The objective is to adjust the model parameters and structures to generate $\hat{y}$ to represent the output estimation of the system having a single output y and multiple inputs u_m $(m = 1, ..., p)$. Mathematically, a parallel superposition of the partial models is just given by summing up the outputs of these models that are multiplied by the associated validity functions. The validity functions determine the corresponding neurons' role in constructing the initial model, which are typically chosen as normalized Gaussian functions with axis-orthogonal [43] functions that are given by

$$\Phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \tag{2.3}$$

where

$$\mu_i(\underline{u}) = exp(-\frac{1}{2}(\frac{(u_i - c_{i1})^2}{\sigma_{i1}^2} + ... + \frac{(u_p - c_{ip})^2}{\sigma_{ip}^2})) \tag{2.4}$$

The normalized Gaussian validity function $\Phi_i(.)$ depends on two parameters, namely the center coordinates c_{ij} and the standard deviation σ_{ij}. Since validity functions control the activity of the LLM, they are also called activation functions.

The estimated output as depicted in Figure 2.5 is given by

$$\hat{y} = \sum_{i=1}^{M} \Phi_i \hat{y}_i \tag{2.5}$$

where Φ_i is the validity function and $\hat{y}_i$ is the output of each locally linear model. The larger the Φ_i, the more is the contribution of $\hat{y}_i$ in the output estimation.

The output of the i^{th} LLM is given by

$$\hat{y}_i = w_{i0} + w_{i1}u_1 + w_{i2}u_2 + ... + w_{ip}u_p \tag{2.6}$$

where w_{ij} denotes the LLM parameters for the neuron i and the input j. The parameters are optimized by using local linear model tree (LoLiMoT) technique which is discussed in next subsection.

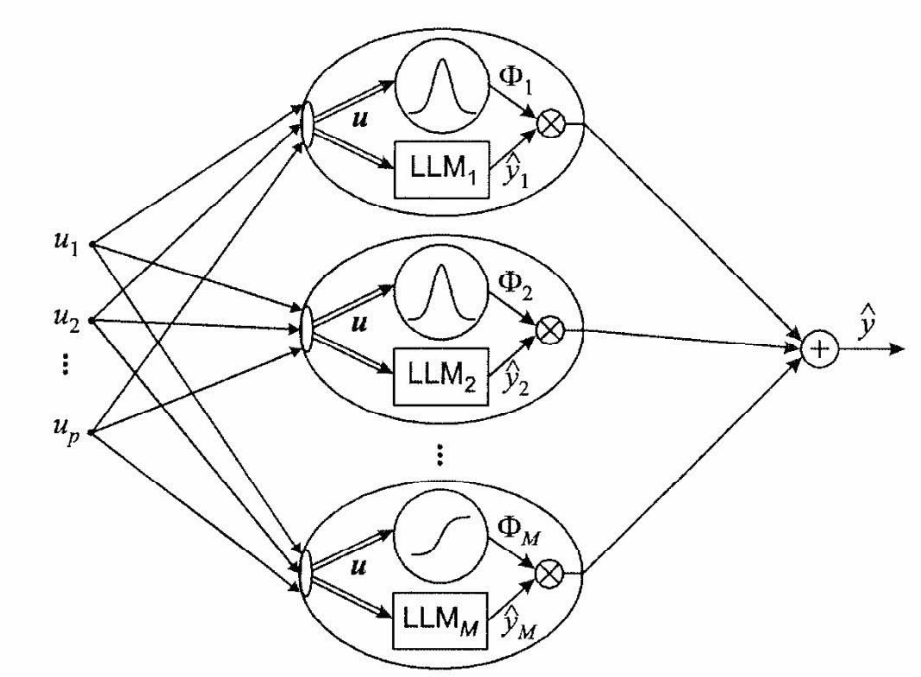

Figure 2.5: The network structure of an LLM model with M neurons corresponding to p inputs [43].

2.4.1 Local Linear Model Tree (LOLIMOT) Algorithm

As described before, the division strategy for the original complex problem is the most important factor for the success of a local linear modeling. Therefore, the properties of the LLM models crucially depend on the applied construction algorithm that implements a certain division strategy. In this study, the LoLiMoT strategy which is a powerful neuro-fuzzy algorithm is used to train the LLM model. LoLiMoT is an incremental tree-construction algorithm that partitions the input space by axis orthogonal splits [43]. In each iteration a new rule or LLM is added to the model. The term neuro is used for the LoLiMoT algorithm because it is designed in a network form and it learns from data. It is considered as a fuzzy network, because it uses local linear models. The fuzzy rules in this method are the local models which are added to the main model at each iteration [65]. The training algorithm for the LoLiMoT can be divided into internal and external loops which optimizes in parallel the identification process in each iteration. The external loop adjusts the structure

of the system and the internal loop adjusts its parameters.

LoLiMoT Algorithm [43, 58, 65]: The overall objective is to adjust the model parameters and its structure in Equation (2.5) and Figure 2.5 to estimate the output of the system y by using k samples $\{(U_n, y_n)\}_{n=1}^{k}$, where U is the input vector of u_m $(m = 1, 2, ..., p)$ having a dimension p. The algorithm is devised as follows.

1. Start with an initial model: At first, a neuron with linear gains $(w_{10}, w_{11}, ..., w_{1p})$ and a validity function ($\Phi_1 = 1$) is trained to yield

$$\hat{y} = \Phi_1 \hat{y_1} = \Phi_1 (w_{10} + w_{11} u_1 + w_{12} u_2 + ... + w_{1p} u_p) \tag{2.7}$$

To adjust the weights the weighted least squares (WLS) method is used which leads to a particularly simple solution as

$$\mathbf{w} = (\mathbf{U}^T Q_i \mathbf{U})^{-1} \mathbf{U}^T Q_i \mathbf{Y} \tag{2.8}$$

where $Q_i = diag(\phi_i(U(1)), \phi_i(U(2)), ..., \phi_i(U(N)))$,

$$\mathbf{U} = \begin{pmatrix} 1 & u_1(1) & u_2(1) & \dots & u_p(1) \\ 1 & u_1(2) & u_2(2) & \dots & u_p(2) \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & u_1(k) & u_2(k) & \dots & u_p(k) \end{pmatrix}$$

$\mathbf{Y} = [y_1\ y_2\ ...\ y_k]^T$ denotes the output vector of the training set data and $\mathbf{w} = [w_1\ w_2\ ...\ w_P]^T$ denotes the vector of weights.

2. Obtain the worst LLM: Calculate a local loss function J_i for each of the $i = 1, ..., M$ locally linear models using the:

$$J_i = \sum_{n=1}^{k} e^2(n) \Phi_i(U(n)) \tag{2.9}$$

where k is the number of the training data points, e is the error between the measured and the estimated output, and $\Phi_i(U)$ for the first iteration is set to 1 and there after its value is

set based on the previous iteration as calculated in the third step below. The worst LLM, which is now obtained corresponding to the below

$$l = maximum(J_i) \tag{2.10}$$

is obtained.

3. Evaluate all the units or divisions: The LLM l is considered for further refinement. The hyper-rectangle of this LLM should be split into two halves with an axis orthogonal split. Division in all the p dimensional spaces are tried. For each possible division, the validity functions and the LLM parameters are calculated by using Equations (2.3) and (2.8), respectively (choosing the value of the centers c_{ij}s and the standard deviation σ_{ij} for Gaussian validity functions will be discussed in subsequently.) The local loss function is computed by using Equation (2.9).

4. Obtain the best division: The best of the p alternates (the one with the least loss function) based on the ones constructed in step 3 is chosen and adapted for the LLM.

5. Test the stopping condition: If the stopping condition is satisfied stop the algorithm, otherwise go to Step 2.

Dividing and training the neurons is continued until a stopping condition is satisfied. This condition can be set based on a sufficiently small sum square estimation error or an upper bounded on the maximum reachable number of neurons. The flowchart of this algorithm is shown in Figure 2.6 (a). The dividing method for a two dimensional function with three iterations is shown in Figure 2.6 (b). This technique provides a powerful tool for non-parametric analysis and control of nonlinear systems [43, 56, 65, 66].

Choosing the centers c_{ij}s and the standard deviations σ_{ij} are not discussed in the above algorithm steps. Figure 2.7 shows these parameters for three neurons for a two dimensional function. c_{ij}s show the center of the Gaussian functions which are chosen as the center of the rectangles and Δ_{ij} denotes the extension of the hyper rectangle of the local model i in the dimension u_j. The only parameter that has to be specified by the user

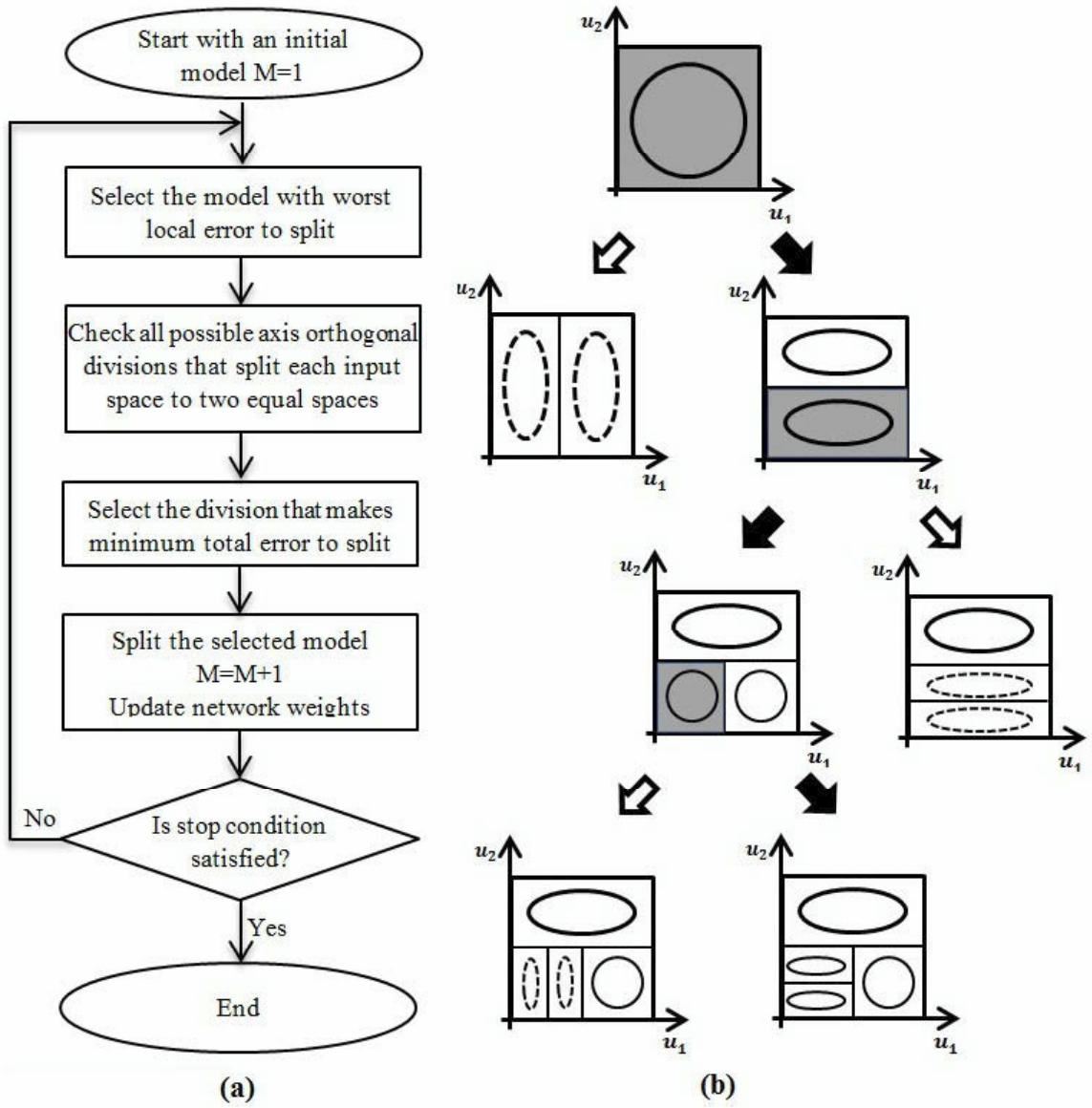

Figure 2.6: (a) The procedure for the LoLiMoT algorithm [67] (M is number of the neurons and the local linear models), (b) Three iterations of the LoLiMoT algorithm for a two dimensional function; neurons are divided orthogonally in order to result in the least estimation error.

is the standard deviation of the Gaussian functions, that is

$$\sigma_{ij} = \alpha \Delta_{ij} \tag{2.11}$$

where α is the proportionally factor between the rectangles' extensions and the standard deviations. Although each specific application has its optimal value, usually $\alpha = \frac{1}{3}$ is a good selection [43].

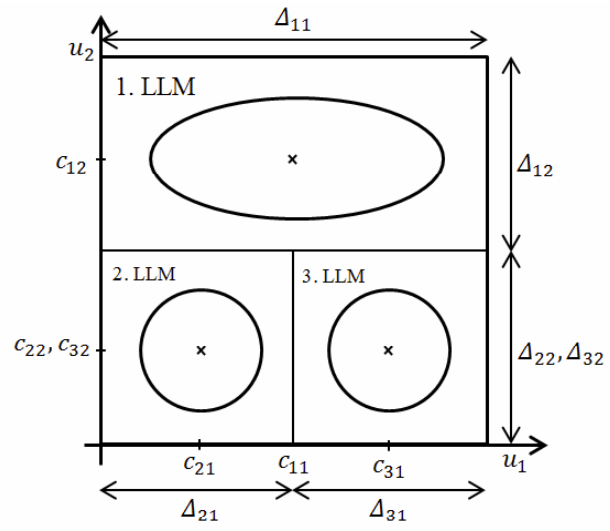

Figure 2.7: The unique relationship between the input partitioning and the validity functions Φ_is [43].

The above approach is quite fast and robust. It is especially important to note that due to the local estimation approach, the computational demand increases only linearly with the number of the local models [43].

2.5 Neural Networks

In this section, another computational intelligence modeling method, namely the radial basis function (RBF) neural network is presented. Neural networks are now widely used

in several fields in industry for automatic process control, medical image analysis, meteorology for weather forecast, etc. In fact, they are applicable in almost every situation in which a relation between the predictor variables (inputs) and predicted variables (outputs) exists, even when that relationship is complex [68].

Structure of a Neural Network

An artificial neural network (ANN), usually called neural network, is a mathematical model using combination of many neurons that are inspired by the biological nervous systems, such as the brain. Figure 2.8 shows a simplified multi-input single-output neuron which consists of weights $w_{1,1}$ to $w_{1,R}$, a bias b, an activation function f and a gain a. A feed forward neural network is made up of several neurons connected into one or more layers as illustrated in Figure 2.9. For most networks, a layer contains neurons that are not connected to one another. The input layer is not really a layer of neurons as these units simply serve to introduce the values of the input variables. The hidden and output layer neurons are connected to all units in the preceding layer. Again, it is possible to define networks that are partially-connected to only some units in the preceding layer; however, for most applications fully-connected networks are used [68].

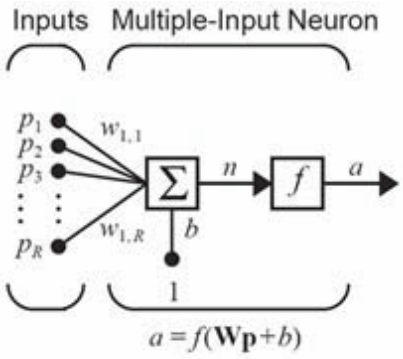

Figure 2.8: A simple neuron [44].

The result of a feed forward neural network as illustrated in Figure 2.9 is a static

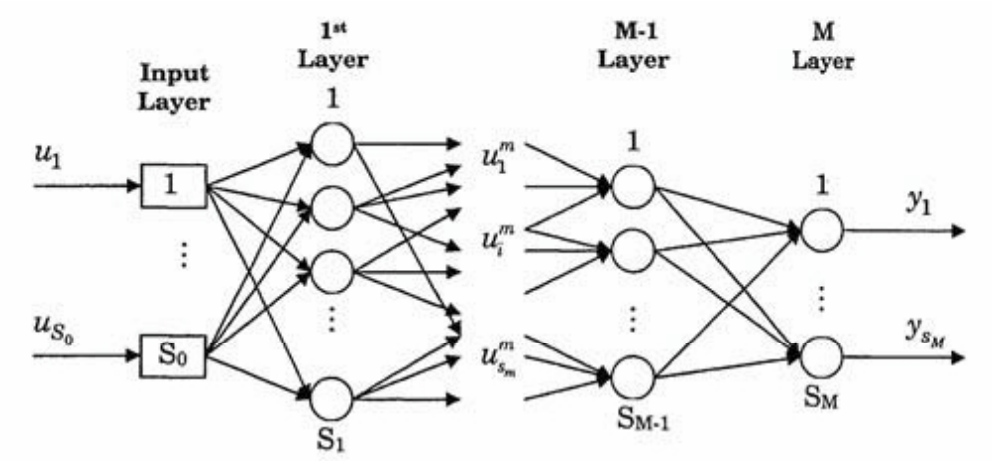

Figure 2.9: M-layer feed forward neural network [44].

function. In contrast with these networks, feedback networks have loops which allow signals to travel from output of a neuron to its input. Feedback networks are dynamic and can become extremely complicated. In this thesis, a feed forward RBF neural network is used as described next.

2.5.1 Radial Basis Function (RBF) Neural Networks

RBF neural network is a feed forward network with one hidden layer. As the name implies, a radial construction mechanism is used in its hidden layer. Radial functions are a special class of functions. Their response depends on the input distance from a central point. Gaussian-like radial basis functions are local and are more commonly used than multiquadric-type radial basis functions which have a global response. They are also more biologically plausible due to their finite response.

Hidden neurons of the RBF neural networks are based on the distance between an input vector and the prototype vector of the neuron. In an RBF neuron, first the distance of each input to the center of the neuron is calculated. The result then passes the activation function $g(x)$ which is usually chosen as a Gaussian function

$$g(x_j) = exp(-\frac{1}{2}x_j^2) \tag{2.12}$$

The distance x_j is calculated by using the center c_j and the norm matrix $\underline{\Sigma_j}$, which are the hidden layer parameters of the j^{th} RBF neuron, as follows

$$x_j = ||\underline{u} - \underline{c}_j||_{\underline{\Sigma_j}} = \sqrt{(\underline{u} - \underline{c}_j)^T \underline{\Sigma_j} (\underline{u} - \underline{c}_j)} \tag{2.13}$$

Often the $\underline{\Sigma}$ matrix is diagonal, so that it contains the inverse variances for each input dimension. Therefore, the distance calculation for the p inputs is simplified to

$$x_j = ||\underline{u} - \underline{c}_j||_{\underline{\Sigma_j}} = \sqrt{\sum_{i=1}^{P} (\frac{u_i - c_{ji}}{\sigma_{ji}})^2} \tag{2.14}$$

Superposition of several such neurons creates an RBF neural network. The functional representation of the RBF network which is shown in Figure 2.10 can be written as

$$\hat{y} = \sum_{j=0}^{m} w_j \Phi_j(x), \Phi_j(x) = exp(-\frac{||x - c_j||^2}{2\sigma_j^2}), \;\; \Phi_0(.) = 1 \tag{2.15}$$

where m is the number of neurons and Φ_j is the activation function of the j^{th} neuron.

The parameters that have to be determined or optimized during the training of an RBF neural network are as follows [43]:

-***Centers*** ***(c_j)*** are nonlinear parameters of the hidden layer neurons. They determine the positions of the basis functions.

Several methods exist [43] to find the proper centers for an RBF neural network such as the center selection using the subset of data points, clustering based algorithms, nonlinear optimization center placement, etc. Among these methods, the K-means clustering algorithm is the most common and simple clustering algorithm [43], and is used to train the RBF neural network in this thesis.

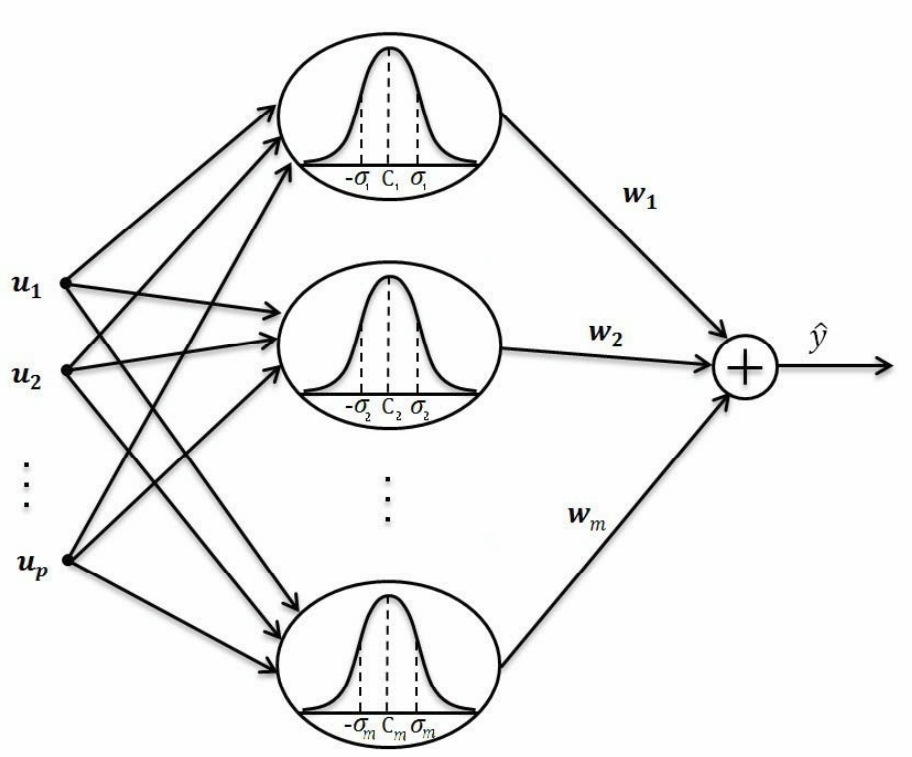

Figure 2.10: An RBF neural network [43].

K-means clustering algorithm. Clustering algorithms are used to obtain a set of centers which reflects the distribution of the data points. Each center c_j is supposed to be representative of a group or set of data points.

Suppose there are p data points $x_i, (i = 1, ..., p)$ in total and where finding m representative vectors c_j, $j = 1, ..., m$ is desired. The number of centers (m) is decided in advance. The algorithm seeks to partition the data points $x_i, i = 1, ...p$, into m disjoint subsets S_j containing P_j data points, in such a way as to minimize the sum of squares clustering function as given by

$$J = \sum_{j=1}^{m} \sum_{x_i \in S_j} ||x_i - c_i||^2 \tag{2.16}$$

Minimization of the above mentioned cost function is accomplished as follows [43]:

1. Choose initial values for the C cluster centers $\underline{c}_j, j = 1, ..., C$. This can be done

by picking randomly C different data samples.

2. Assign all data samples to their nearest cluster center.

3. Compute the centroid (mean) of each cluster. Set each center to the centroid of its cluster, that is

$$c_j = \frac{1}{P_j} \sum_{x_i \in S_j} x_i. \tag{2.17}$$

where i runs over those P_j data samples that belongs to the cluster j

4. If any cluster center has been moved in the previous step go to step 2; otherwise stop.

The above mentioned classical k-mean clustering algorithm has been extended to more complex methods such as the on-line K-means clustering algorithm, Fuzzy K-mean algorithm, and Gustafson-kessel algorithm, among others [43].

-***Standard deviations*** (σ_j) are nonlinear parameters of the hidden layer neurons. They determine the width (and possibly rotations) of the basis functions. In this thesis, standard deviations of the basis functions are adjusted by using test data and the trial and error approach which is described in section 2.3.2.

-***Output Layer weights*** (w_j) are linear parameters. They determine the heights of the basis functions and offset value. The relation between the output of the neural network and its weights as illustrated in Figure 2.10 is given by

$$\hat{y} = \sum_{j=1}^{m} w_j h_j(x) \tag{2.18}$$

where h_js are the radial basis functions' outputs and w_js are linear weights. In this case, applying a supervised learning method, the least squares (LS) principle leads to a particularly easy optimization. The LS by using the training sets ($\{(x_i, y_i)\}_{i=1}^{p}$), minimizes the sum squared error S as given by

$$S = \sum_{i=1}^{p} (y_i - \hat{y}_i)^2 \tag{2.19}$$

with respect to the weights of the model. If a weight penalty term is added to the above sum squared error with a term which penalizes large weights, the cost function S will become

$$C = \sum_{i=1}^{p} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} w_j^2 \tag{2.20}$$

The above is a ridge regression (weight decay) and the regularization parameter $\lambda > 0$ controls the balance between the fitting the data and avoiding the penalty. A small value for λ implies the data can be fit tightly without causing a large penalty; and a large value for λ implies a tight fit has to be sacrificed if it requires large weights. In [69] the minimization of the above cost function is found by using

$$\hat{\mathbf{w}} = (\mathbf{H}^T\mathbf{H} + \lambda \mathbf{I}_m)^{-1}\mathbf{H}^T\mathbf{Y} \tag{2.21}$$

where $\mathbf{H}$, the design matrix, is given by

$$\mathbf{H} = \begin{pmatrix} h_1(x_1) & h_2(x_1) & \dots & h_m(x_1) \\ h_1(x_2) & h_2(x_2) & \dots & h_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(x_p) & h_2(x_p) & \dots & h_m(x_p) \end{pmatrix}$$

and $\mathbf{Y} = [y_1\ y_2\ ...\ y_P]^T$ is the output vector of the training set and $\hat{\mathbf{w}} = [\hat{w}_1\ \hat{w}_2\ ...\ \hat{w}_P]^T$ is the vector of weights which minimizes the cost function C [69].

2.6 LLM vs. RBF, Comparative Study

Neuro-fuzzy models, such as neural networks, are general function approximators [43, 70]. Several studies have demonstrated their performance in nonlinear system identification and

control [71, 72]. An intuitive way of training and the use of minimum number of adjustable parameters, that lead to a high generalization performance are two characteristics of the LLM models.

Using the Gaussian validity functions in the LLM neurons make this approach quite similar to a modified version of RBF networks which use normalized radial basis functions (NRBF). Although these methods originate from different modeling families having different interpretations, their structure is almost the same. A comparison of these methods in [43] is shown in Figure 2.11. In fact, in the RBF network, normalized Gaussian basis functions are weighted with a constant coefficient (w_{i0}) while a local linear model uses different coefficients for each neuron. In other words, LLM degenerates to a NRBF network if $w_{i0} \neq 0$ and $w_{i1} = w_{i2} = ... = w_{ip} = 0$.

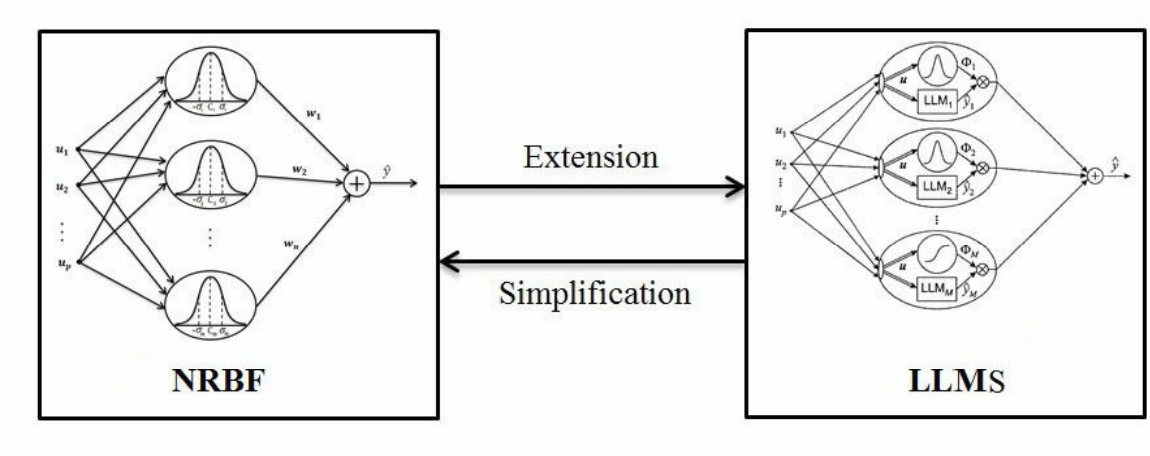

Figure 2.11: Relationship between the LLM model and an RBF network [43].

In [56] modeling and identification capability of these methods on catalytic reformer unit has been compared. Final gasoline products qualities such as research octane number (RON) and reid vapor pressure (RVP) in catalytic reformer unit are chosen to be identified. RVP and RON are nonlinear functions of qualities of blend components. As concluded in [56], modeling the studied system using LLM model leads to a estimation error as compared to the RBF neural network.

2.7 Conclusions

In modern applications, devices and systems are increasingly being used in regimes where they cannot be accurately described by linear models accurately. This has led to a growing need for models and modeling techniques that are able to adequately describe the behavior of these systems. Nonlinear system estimation is a very broad problem and it is impossible to propose a structure capable of describing efficiently every possible system. In this chapter, two intelligent nonlinear system identification methods have been introduced which will be used for fault detection of a mobile robot in subsequent chapters.

Chapter 3

Mobile Robot Model and Simulator

3.1 Introduction

This chapter introduces the mobile robot governing equations which are then used to create a simulator. Among several models for mobile robots, the model presented in [73]-[77] which is for directional two wheeled mobile robot is used in this thesis. It is the most appropriate model for fault detection purposes because the model contains dynamics and kinematics equations of the mobile robot and the available robot in our lab is also a directional two wheeled mobile robot (Pioneer P3DX). After presenting the equations of the mobile robot, a controller is designed by using an MIMO PD law. Finally, simulation results are provided to show various trajectory tracking performance of the mobile robot.

3.2 System Modeling

To model a mobile robot, a few assumptions are needed to be considered;

a) The robot moves in a planar surface.

b) Wheels rotate without any slippage.

c) The robot is considered as a solid rigid body, and movable parts are the wheels which are activated to follow a commanded control position.

Upon these assumptions, equations of a two wheeled mobile robot are derived below. Figure 3.1 shows a schematic of a mobile robot with its parameters which are used in modeling.

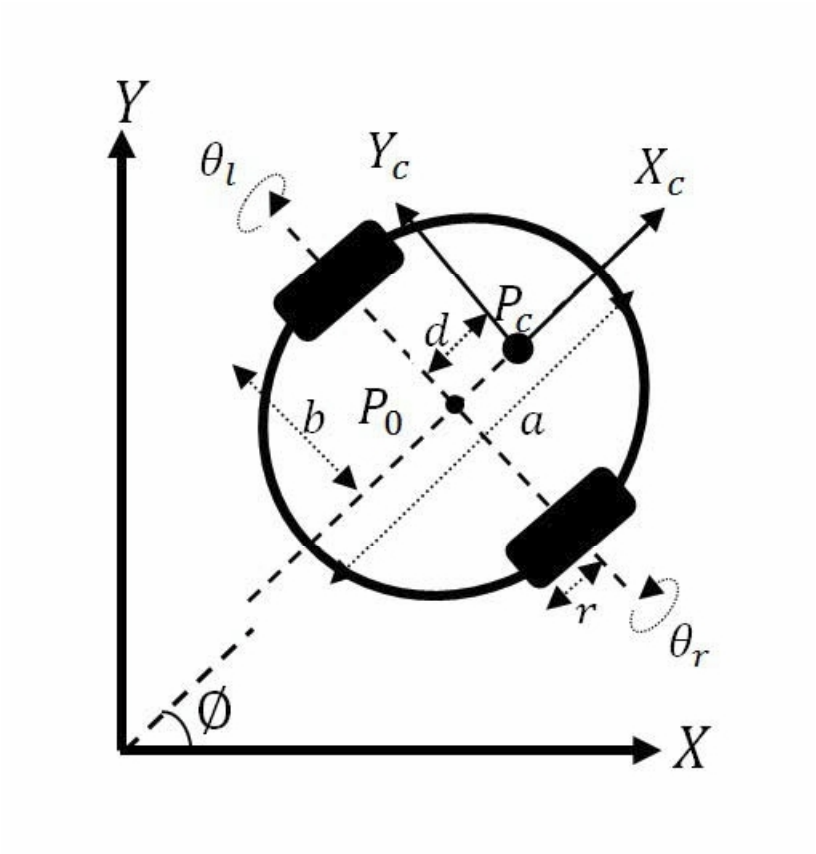

Figure 3.1: A schematic of a mobile robot [77].

The following notations are used in the modeling procedure of the mobile robot which are illustrated in Figures 3.1 and 3.6.

$P_o\ (x_o, y_o)$: the intersection of the axis of symmetry with the driving wheel axis;

$P_c\ (x_c, y_c)$: the center of platform's mass in world coordinate system;

ϕ : the heading angle of the platform;

d : the distance from P_o to P_c;

b : the distance between the driving wheels and the axis of symmetry;

r : the radius of each driving wheel;

m_c : the mass of the platform without the driving wheels and the rotors of the dc motors;

m_w : the mass of each driving wheel plus the rotor of its motor;

I_c : the moment of inertia of the platform without the driving wheels and the rotors of the motors about a vertical axis through Po;

I_w : the moment of inertia of each wheel;

I_m : the moment of inertia of each wheel and the motor rotor about the wheel diameter;

θ_r, θ_l : right and left wheels' angular positions;

v, w : linear and angular velocities of the mobile robot;

ω_r, ω_l : right and left wheels' angular velocities;

τ_r, τ_l : right and left wheels' torques;

U_r, U_l : right and left wheel armatures' voltages;

i_r, i_l : right and left wheel armatures' currents;

L_{ar}, L_{al} : right and left wheel armatures' inductances;

R_{ar}, R_{al} : right and left wheel armatures' resistors;

K_{emf} : speed constant of motor;

K_{motor} : motor constant

3.2.1 Kinematic Model

Most physical equations consist of constraints. For example, an inverted pendulum as shown in Figure 3.2 is restricted by Equation (3.1).

$$x^2 + y^2 - l^2 = 0 \tag{3.1}$$

Similar constrained equations are found in the kinematic equations of mobile robots. A wheel speed (Figure 3.3) in the direction x is calculated by the radius r and the angle speed rotation θ by

$$\dot{x} = r\dot{\theta} \tag{3.2}$$

where $\dot{x}$ is the speed in the x direction which is proportional to the angular velocity of

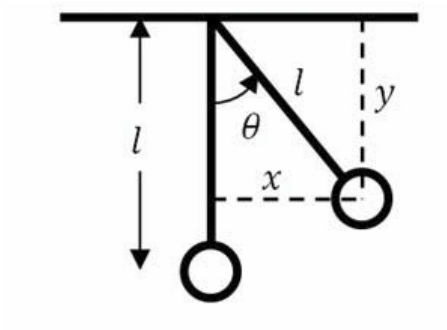

Figure 3.2: An inverted pendulum.

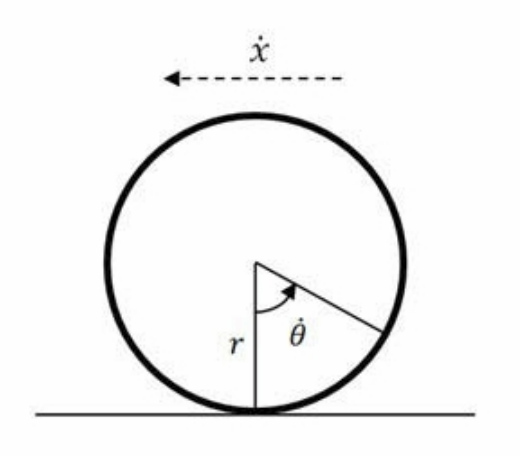

Figure 3.3: Speed of wheels.

the wheel. However, other constraints appear in wheels when the movement is in the XY plane. Assume the mobile robot has the velocity V with angular orientation of ϕ (Figure 3.4). Considering 3 assumptions in Section 3.2, constraints of a mobile robot can be extracted as

$$\begin{aligned} \dot{x_c}\sin\phi - \dot{y_c}\cos\phi + d\dot{\phi} &= 0 \\ \dot{x_c}cos\phi + \dot{y_c}\sin\phi + b\dot{\phi} &= r\dot{\theta}_r \\ \dot{x_c}cos\phi + \dot{y_c}\sin\phi - b\dot{\phi} &= r\dot{\theta}_l \end{aligned} \tag{3.3}$$

To derive the first equation, assume that a wheel has the speed V. Due to the symmetry

of the robot, both wheels can be considered in $P_o = (x_o, y_o)$ with the speed $V = \vec{x_o} + \vec{y_o}$ (Figure 3.4), as follows

$$\begin{aligned} V\cos\phi = \dot{x_o} \rightarrow V\cos\phi\sin\phi = \dot{x}_o\sin\phi \\ V\sin\phi = \dot{y_o} \rightarrow V\cos\phi\sin\phi = \dot{y}_o\cos\phi \\ \rightarrow \dot{x}_o\sin\phi - \dot{y}_o\cos\phi = 0 \end{aligned} \tag{3.4}$$

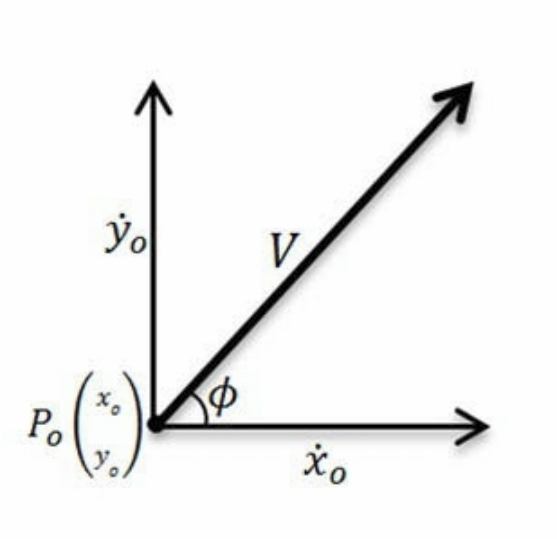

Figure 3.4: Velocity of the mobile robot.

$\begin{pmatrix} x_o \\ y_o \end{pmatrix}$ is the point in the middle of the wheels and $\begin{pmatrix} x_c \\ y_c \end{pmatrix}$ is the position of center mass of the mobile robot. Relation between x_o, x_c and y_o, y_c is given by

$$\begin{aligned} x_o = x_c + dcos\phi,\ y_o = y_c + dsin\phi \rightarrow \\ \dot{x}_o = \dot{x}_c + d\dot{\phi}sin\phi,\ \dot{y}_o = \dot{y}_c - d\dot{\phi}cos\phi (\dot{d} = 0) \end{aligned} \tag{3.5}$$

Considering Equation (3.5), Equation (3.4) can be rewritten as

$$\begin{aligned} \dot{x}_c sin\phi + d\dot{\phi}sin^2\phi - \dot{y}_c cos\phi + d\dot{\phi}cos^2\phi = 0 \rightarrow \\ \dot{x}_c sin\phi - \dot{y}_c cos\phi + d\dot{\phi} = 0 \end{aligned} \tag{3.6}$$

Equation (3.6) is the first constraint that is used for modeling the mobile robot.

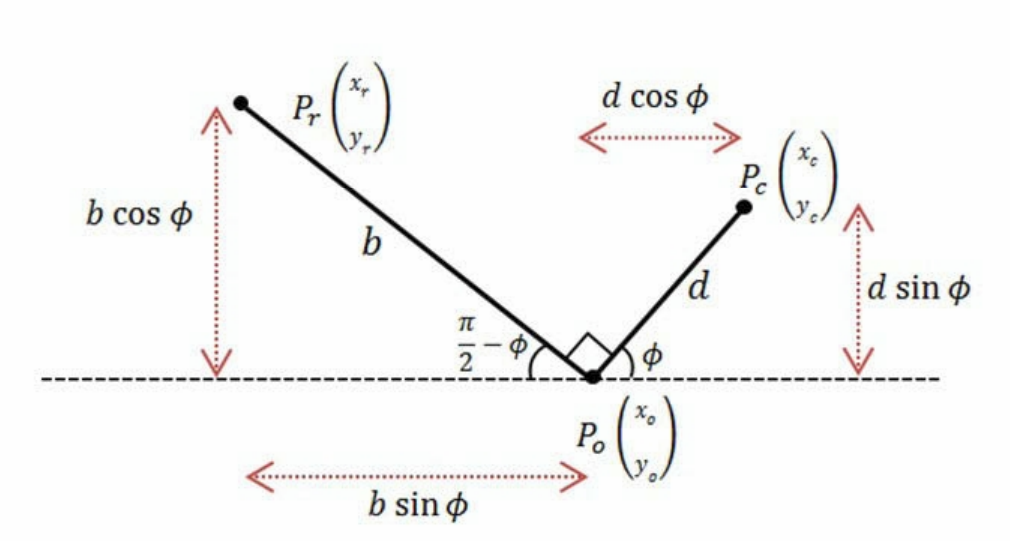

Figure 3.5: Relation between the hotspots of a mobile robot.

The other constraints are the rolling constraints, i.e., the driving wheels do not slip

$$\begin{aligned} \dot{x_c}cos\phi + \dot{y_c}\sin\phi + b\dot{\phi} &= r\dot{\theta}_r \\ \dot{x_c}cos\phi + \dot{y_c}\sin\phi - b\dot{\phi} &= r\dot{\theta}_l \end{aligned} \tag{3.7}$$

where θ_r and θ_l are the angular positions of the right and the left wheels, respectively.

To derive Equation (3.7), again consider the velocity equations of the mobile robot in Equation (3.4). As the robot is assumed to be a rigid body, all parts of it move with the same velocity. The equations for the right wheel can be written as

$$\begin{aligned} V_r cos\phi = \dot{x}_r \rightarrow V_r cos^2\phi = \dot{x}_r cos\phi \\ V_r sin\phi = \dot{y}_r \rightarrow V_r sin^2\phi = \dot{y}_r sin\phi \end{aligned} \tag{3.8}$$

$$V_r = r\dot{\theta}_r = \dot{x}_r cos\phi + \dot{y}_r sin\phi \tag{3.9}$$

where $\begin{pmatrix} x_r \\ y_r \end{pmatrix}$ is the position of the right wheel. With reference to Figure 3.5, the components of Equation (3.9) can be found as follows

$$
\begin{aligned}
x_r + b\sin\phi + d\cos\phi = x_c \rightarrow \dot{x}_r = \dot{x}_c + b\dot{\phi}\cos\phi - d\dot{\phi}\sin\phi \rightarrow \\
\dot{x}_r\cos\phi = \dot{x}_c\cos\phi + b\dot{\phi}\cos^2\phi - d\dot{\phi}\sin\phi\cos\phi \\
y_r - b\cos\phi + d\sin\phi = y_c \rightarrow \dot{y}_r = \dot{y}_c + b\dot{\phi}\sin\phi + d\dot{\phi}\cos\phi \rightarrow \\
\dot{y}_r\sin\phi = \dot{y}_c\sin\phi + b\dot{\phi}\sin^2\phi - d\dot{\phi}\sin\phi\cos\phi
\end{aligned}
\tag{3.10}
$$

Hence, considering Equations (3.9) and (3.10), the constraints in Equation (3.7) is derived as follows

$$
\begin{aligned}
V_r = r\dot{\theta}_r = \dot{x}_r\cos\phi + \dot{y}_r\sin\phi \\
= \dot{x}_c\cos\phi + b\dot{\phi}\cos^2\phi - d\dot{\phi}\sin\phi\cos\phi + \dot{y}_c\sin\phi + b\dot{\phi}\sin^2\phi + d\dot{\phi}\sin\phi\cos\phi \\
\rightarrow \dot{x}_c\cos\phi + \dot{y}_c\sin\phi + b\dot{\phi} = r\dot{\theta}_r
\end{aligned}
\tag{3.11}
$$

The same procedure can be used to find the constraint for the left wheel as given below

$$
\dot{x}_c\cos\phi + \dot{y}_c\sin\phi - b\dot{\phi} = r\dot{\theta}_l \tag{3.12}
$$

The three constraints given in Equation (3.3) can be written in the form

$$
A(q)\dot{q} = 0 \tag{3.13}
$$

where the generalized coordinates of the mobile robot are denoted by $q = (x, y, \phi, \theta_r, \theta_l)^T$ and

$$
\mathbf{A}(\mathbf{q}) = \begin{pmatrix}
-\sin\phi & \cos\phi & -d & 0 & 0 \\
-\cos\phi & -\sin\phi & -b & r & 0 \\
-\cos\phi & -\sin\phi & b & 0 & r
\end{pmatrix}
$$

3.2.2 Dynamic Model

Lagrange formulation is used to establish equations of motion for the mobile robot. Since there is no change in the potential energy in the system, the Lagrangian is simply kinetic energy term (it is assumed that the robot moves in a planar surface). The total kinetic energy of the mobile base and the two wheels is given by

$$K = \frac{m}{2}(\dot{x}^2 + \dot{y}^2) + m_c c d(\dot{\theta}_r - \dot{\theta}_l)(\dot{y}cos\phi - \dot{x}sin\phi) + \frac{I_w}{2}({\dot{\theta}_r}^2 + {\dot{\theta}_l}^2) + \frac{Ic^2}{2}(\dot{\theta}_r - \dot{\theta}_l)^2 \quad (3.14)$$

where

$$\begin{aligned} m &= m_c + 2m_w \\ I &= I_c + 2m_w(d^2 + b^2) + 2I_m \end{aligned} \quad (3.15)$$

Using Equation (3.14), the Lagrange equations of motion of the platform and the Lagrange multipliers λ_1,λ_2 and λ_3 are given by

$$\begin{aligned} m\ddot{x}_c - m_c d(\ddot{\phi}sin\phi + \dot{\phi}^2 cos\phi) - \lambda_1 sin\phi - (\lambda_2 + \lambda_3)cos\phi = 0 \\ m\ddot{y}_c + m_c d(\ddot{\phi}cos\phi - \dot{\phi}^2 sin\phi) + \lambda_1 cos\phi - (\lambda_2 + \lambda_3)sin\phi = 0 \\ -m_c d(\ddot{x}_c sin\phi - \ddot{y}_c cos\phi) + I\ddot{\phi} - d\lambda_1 + b(\lambda_3 - \lambda_2) = 0 \\ I_w\ddot{\theta}_r + \lambda_2 r = \tau_r \\ I_w\ddot{\theta}_l + \lambda_3 r = \tau_l \end{aligned} \quad (3.16)$$

where τ_1 and τ_2 are the torques that are acting on the wheel axis and generated by the right and the left motors, respectively. The above five equations of motion can be written in the compact form as

$$M(q)\ddot{q} + V(q,\dot{q}) = E(q)\tau - A^T(q)\lambda \quad (3.17)$$

The matrix $A(q)$ has been defined in Equation (3.13), $\tau = [\tau_r\ \tau_l]^T$ and the matrices $M(q)$, $V(q,\dot{q})$ and $E(q)$ are given by

$$\mathbf{M} = \begin{pmatrix} m & 0 & -m_c dsin\phi & 0 & 0 \\ 0 & m & m_c dcos\phi & 0 & 0 \\ -m_c dcos\phi & m_c dcos\phi & I & 0 & 0 \\ 0 & 0 & 0 & I_w & 0 \\ 0 & 0 & 0 & 0 & I_w \end{pmatrix}$$

$$\mathbf{V} = \begin{pmatrix} -m_c d\dot{\phi}^2 cos\phi \\ -m_c d\dot{\phi}^2 sin\phi \\ 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{E} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The dynamic Equation (3.17) and the constraint Equation (3.13) will be presented in the state space form by properly choosing a state vector. To do so, a 5×2-dimensional matrix $S(q)$ is defined such that $A(q)S(q) = 0$. It is straightforward to show that the following matrix has the required property (the constant c is $\frac{r}{2b}$), that is

$$\mathbf{S} = [s_1(q)\ s_2(q)] = \begin{pmatrix} c(bcos\phi - dsin\phi) & c(bcos\phi - dsin\phi) \\ c(bsin\phi - dcos\phi) & c(bsin\phi - dcos\phi) \\ c & -c \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

According to the constraint Equation (3.13), $\dot{q}$ is in the null space of $A(q)$. Since the two columns of $S(q)$ are in the null space of $A(q)$ and are linearly independent, it is possible to express $\dot{q}$ as a linear combination of the two columns of $S(q)$, that is

$$\dot{q} = S(q)\omega \tag{3.18}$$

The logic behind Equation (3.18) is to introduce a set of independent velocity variables $\dot{\theta}_r, \dot{\theta}_l$. Owing to the choice of $S(q)$ matrix we have:

$$\omega = \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} = \begin{pmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{pmatrix}$$

By differentiating Equation (3.18), substituting the expression for $\ddot{q}$ into Equation (3.17) and premultiplying it by S^T, one gets

$$\tau = S^T(MS\dot{\omega}(t) + M\dot{S}\omega(t) + V) \tag{3.19}$$

The relation between the linear velocity (v) and the angular velocity (w) of the mobile robot with the wheels velocity variables is given by

$$v = \frac{r}{2}(\omega_r + \omega_l),\ w = \frac{-r}{2b}(\omega_r - \omega_l) \tag{3.20}$$

which can be expressed as

$$W = T_q \omega \tag{3.21}$$

where $W = [v\ w]^T$ and T_q, the transform matrix is given by

$$\mathbf{T_q} = \begin{pmatrix} cb & cb \\ c & -c \end{pmatrix}$$

Hence, Equation (3.19) with respect to the relation between the current and the torque created by the motor ($\tau = IK_{motor}$, where $I = [i_r\ i_l]^T$), can be written as

$$IK_{motor} = S^T M S T_q^{-1} \dot{W}(t) + S^T M \dot{S} W(t) + S^T M V \tag{3.22}$$

This results in

$$\dot{W}(t) = (S^T M S T_q^{-1})^{-1} IK_{motor} - (S^T M S T_q^{-1})^{-1} S^T M \dot{S} W(t) - (S^T M S T_q^{-1})^{-1} S^T M V \tag{3.23}$$

3.2.3 Actuator Model

Electrical drives are basic components in a multitude of devices, processes, machinery and vehicles, and in the large areas of mechanical power and process engineering, manufacturing, transportation and precision mechanical devices. Permanent-field motors are the motors that are used in the Pioneer P3DX robots. Modeling of DC motors can be found in [79]-[81]. Figure 3.6 shows that the equation of the armature for a general Permanent-field motor is obtained as follows

$$L_a \frac{di}{dt} = U - R_a i - K_{emf}\omega \tag{3.24}$$

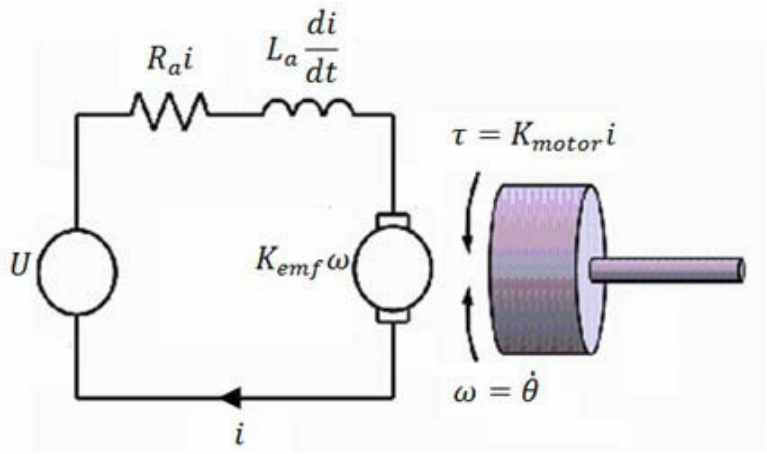

Figure 3.6: A DC motor [81].

Therefore, the left and the right motor equations can be written as

$$\begin{aligned} \dot{i_r} &= -\frac{R_{ar}}{L_{ar}} i_r + \frac{K_{emf}}{L_{ar}}\omega + \frac{U_r}{L_{ar}} \\ \dot{i_l} &= -\frac{R_{al}}{L_{al}} i_l + \frac{K_{emf}}{L_{al}}\omega + \frac{U_l}{L_{al}} \end{aligned} \tag{3.25}$$

The final equation of the system can be found by combining Equations (3.23) and (3.25):

$$\dot{\mathbf{X}}_{\mathbf{4\times 1}} = \begin{pmatrix} -diag(\frac{R_a^T}{L_a}) & diag(\frac{K_{emf}}{L_a} I_{2\times 2}) \\ K_{motor}(S^T M S T_q^{-1})^{-1} & K_{motor}(S^T M S T_q^{-1})^{-1} S^T M \dot{S} \end{pmatrix}_{4\times 4} \mathbf{X}_{\mathbf{4\times 1}} + \begin{pmatrix} \frac{V_s}{L_a} I_{2\times 2} \\ 0_{2\times 2} \end{pmatrix}_{4\times 2} \mathbf{U}_{2\times 1} + \begin{pmatrix} 0_{2\times 1} \\ -(S^T M S T_q^{-1})^{-1} S^T V \end{pmatrix}_{4\times 1}$$

where $X = [i_r\ i_l\ v\ w]^T$, $U = [U_r\ U_l]^T$, $L_a = [L_{ar}\ L_{al}]^T$ and $R_a = [R_{ar}\ R_{al}]^T$.

3.2.4 Control Design

Among a variety of control algorithms that are designed for mobile robots [73, 78, 82], the algorithm presented in [82] is used to control the system. Control signals are the left and the right motor voltages and outputs are linear and angular velocities. Hence, the system is an MIMO control problem. As the goal is to control the position of the robot, the relation between the desired position and the linear and angular velocities should be determined. Assume that the representative point (x,y) of the mobile robot must follow the Cartesian trajectory $(x_d(t), y_d(t))$. The relation between $x_d(t), y_d(t)$ and θ_d, can be defined as

$$\theta_d = ATAN(\dot{y}_d + \dot{x}_d) + k\pi \tag{3.26}$$

where $ATAN2$ is the four-quadrant inverse tangent function (undefined only if both arguments are zero). Therefore, the nominal feed forward commands are given by [30, 83]

$$v_d = \pm\sqrt{\dot{x}_d^2 + \dot{y}_d^2}, w_d = \frac{(\ddot{y}_d\dot{x}_d - \ddot{x}_d\dot{y}_d)}{(\dot{x}_d^2 + \dot{y}_d^2)} \tag{3.27}$$

As described earlier, the relation between the desired linear (v_d) and the angular velocity (w_d) with the desired wheels angular velocity (ω_{rd}, ω_{ld}) is given by

$$[\omega_{rd}\ \omega_{ld}]^T = T_q^{-1}[v_d\ w_d]^T \tag{3.28}$$

Therefore, by controlling the wheels angular velocities (ω_r, ω_l), the mobile robot's linear and angular velocities, and consequently position of the mobile robot can be controlled. By controlling ω_r, ω_l separately, the performance of the system will deteriorate because both control variables relate to both outputs (v,w). To handle this issue, $\omega_r + \omega_l$ and $\omega_r - \omega_l$ should be controlled as they are directly related to the linear and angular speed of the robot. The MIMO controller that is designed for this system is given by

$$\begin{aligned} \tau_{rd} &= \frac{(k_1+k_2)}{2}(\omega_{rd} - \omega_r) + \frac{(k_1-k_2)}{2}(\omega_{ld} - \omega_l) \\ \tau_{ld} &= \frac{(k_1+k_2)}{2}(\omega_{ld} - \omega_l) + \frac{(k_1-k_2)}{2}(\omega_{rd} - \omega_r) \end{aligned} \tag{3.29}$$

which are simply proportional controllers. Derivation of the presented controller can be found in [82]. Equation (3.29) is the desired right and left τs to control the mobile robot, while in the designed mobile robot, control signals are the right and the left motor voltages (U_r, U_l). Considering Equation (3.24) and $\tau_d = IK_{motor}$, the relation between U_r, U_l and τ_{rd}, τ_{ld} is given by

$$\begin{aligned} U_r &= \frac{L_{ar}}{k_{motor}} \frac{d\tau_{rd}}{dt} + \frac{R_{ar}}{k_{motor}} \tau_{rd} + k_{emf} \omega_r \\ U_l &= \frac{L_{al}}{k_{motor}} \frac{d\tau_{ld}}{dt} + \frac{R_{al}}{k_{motor}} \tau_{ld} + k_{emf} \omega_l \end{aligned} \tag{3.30}$$

In conclusion, to control the position of the mobile robot, first the desired wheel angular velocities should be calculated by using Equations (3.26) - (3.28), then the control signals are generated by using the MIMO proportional derivative (PD) controllers in Equations (3.29) and (3.30).

3.3 Simulation Results

Using MATLAB, the presented equations are simulated. The simulator diagram is shown in Figure 3.7. By using the derived equations, the simulator contains the model for all parts of the system and is capable of implementing typical controller, system, actuator and sensor faults in different ways. For simulation, robot parameters are derived from the datasheet of the Pioneer P3DX and the motor parameters are the parameters of a GM9236 motor with 38.3:1 gear ratio which is used as the Pioneer actuator. These parameters with the controller gains are summarized in Table 3.1.

To simulate the mobile robot, the sampling time is selected as $0.1s$. Also, in order to make the data more realistic, white Gaussian noise is added to all the measurements. Figures 3.8 - 3.13 show different mobile robot variables in the trajectory tracking of the cases defined in Table 3.2. As described in Section 3.2.4, the controlled variables are linear

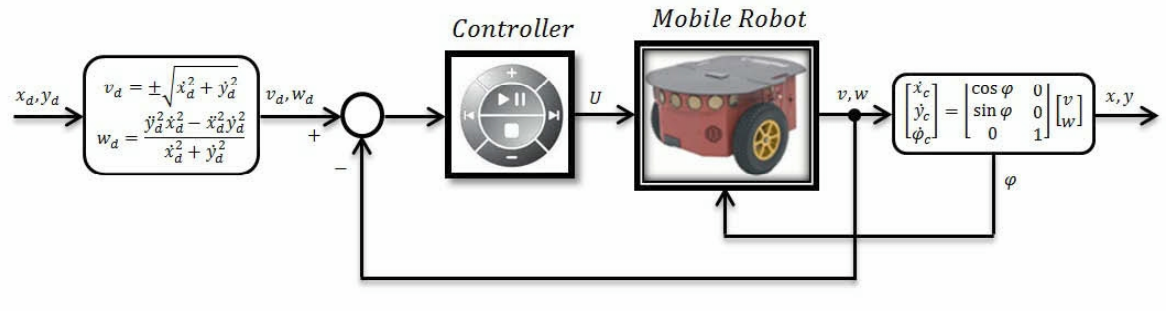

Figure 3.7: Control diagram of the mobile robot.

Parameter	Symbol	Value
Radius of each wheel	r	$0.1m$
Wheel base	b	$0.26m$
Forward distance of center of mass from center of rear axle	d	$0.05m$
The length of the robot	a	$0.5m$
Mass of the robot without the driving wheels and motors	m_c	$7Kg$
Mass of each driving wheel plus its motor	m_w	$1Kg$
Moment of inertia of the platform without the driving wheels and the rotors of the motors about a vertical	m_w	$1Kg$
Moment of inertia of each wheel and the motor rotor about the wheel axis	I_w	$0.005Kg.m^2$
Moment of inertia of each wheel and the motor rotor about the wheel diameter	I_m	$0.0025Kg.m^2$
Armature resistance	R_a	0.71Ω
Armature inductance	L_a	$0.66mH$
Reference voltage	V_s	$12V$
Electromagnetic force constant	K_{emf}	$0.023\frac{V}{\frac{rad}{s}}$
Torque constant	K_{motor}	$0.029\frac{N.m}{A}$
Controller gains	K_1, K_2	$20, 10$

Table 3.1: Parameters of the mobile robot.

Case	Path	X Trajectory	Y Trajectory
1	8-shaped	$3sin(t)$	$4sin(0.5t)$
2	8-shaped	$3sin(t)$	$4sin(0.5t)$
3	8-shaped	$3sin(t)+1$	$4sin(0.5t)$
4	8-shaped	$4sin(0.5t)$	$3sin(t)$
5	ellipse	$0.5cos(1.25t-2.35)cos(0.78)+$ $2sin(1.25t-2.35)(-sin(0.78_{)}$	$0.5cos(1.25t-2.35)sin(0.78)+$ $2sin(1.25t-2.35)cos(0.78)$
6	circle	$cos(1.75t-2.35)cos(0.78)+$ $sin(1.75t-2.35)(-sin(0.78))$	$cos(1.75t-2.35)sin(0.78)+$ $sin(1.75t-2.35)cos(0.78)$

Table 3.2: Different trajectory tracking case studies.

and angular velocities of the mobile robot. Figures 3.8 - 3.13 show that the controller is capable of controlling the linear and angular velocities with small error. The sum square error of the velocities in all the cases are shown in Table 3.3. In the provided set of simulations, the position of the mobile robot is controlled accurately if the initial robot configuration is assumed to be matched with the desired reference trajectory; in other words, we have $q(0) = q_d(0)$. Hence, the feed-forward commands of Equation (3.27) would acquiesce to precise trajectory tracking in ideal conditions [30]. This is illustrated in the figures corresponding to the cases 1, 2 and 6.

Also, Figures 3.14 - 3.19 show the mobile robot variables in an 8-shaped trajectory path with different initial conditions. These figures show that the controller is capable of controlling the velocities with different initial conditions. These different initial conditions applied to the system and the sum square error of the controlled linear and angular velocities in all cases are summarized in Table 3.4.

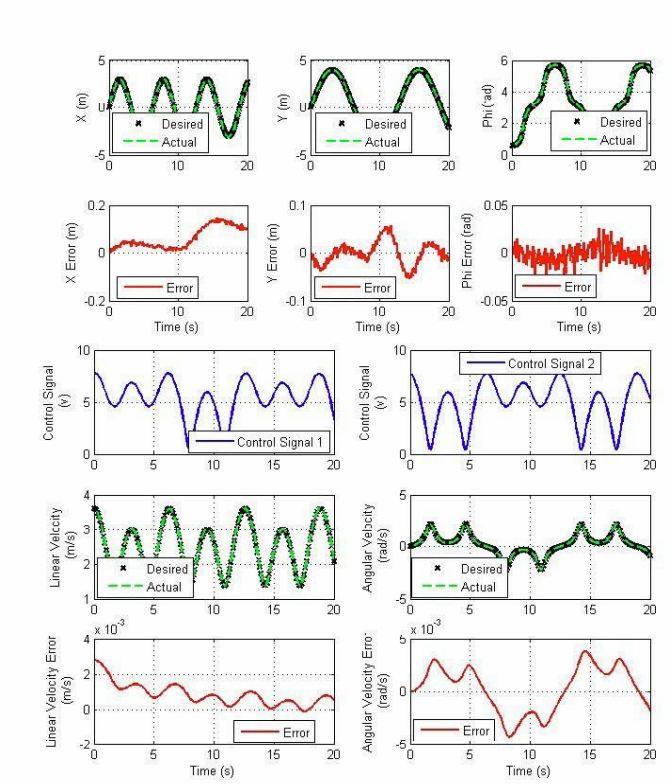

Figure 3.8: The mobile robot variables following an 8-shaped trajectory of case 1 defined in Table 3.3.

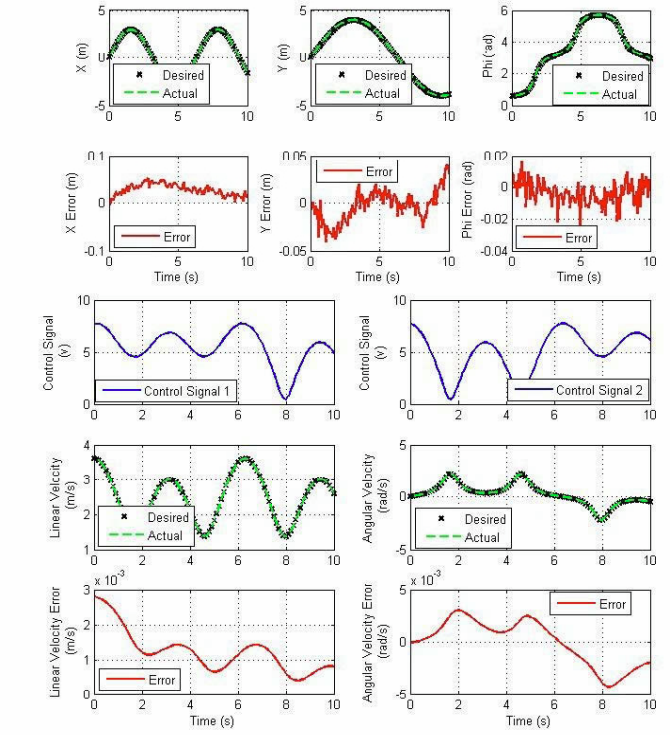

Figure 3.9: The mobile robot variables following an 8-shaped trajectory of case 2 defined in Table 3.3.

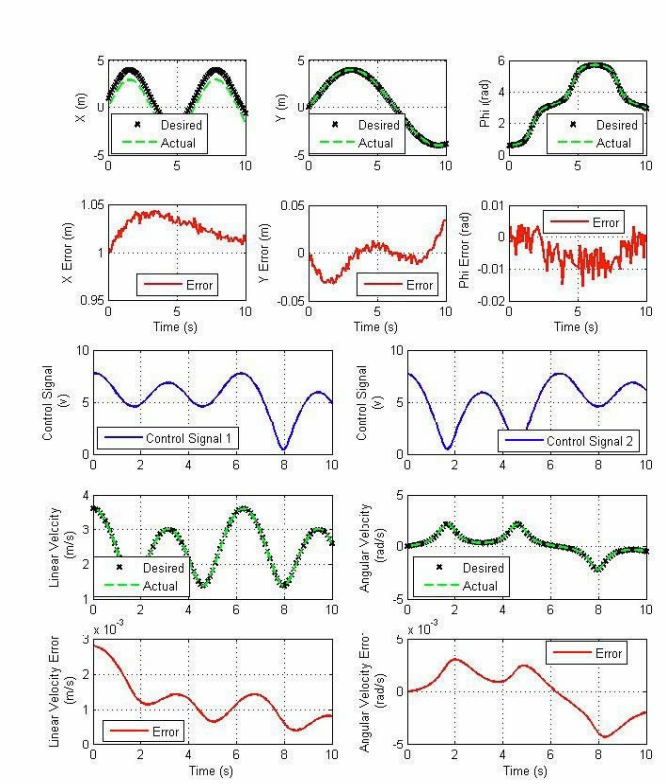

Figure 3.10: The mobile robot variables following an 8-shaped trajectory of case 3 defined in Table 3.3.

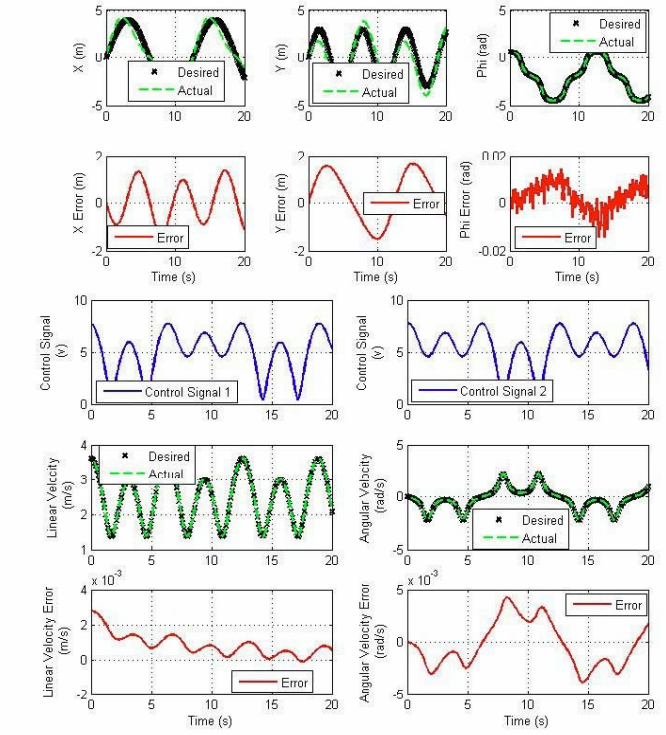

Figure 3.11: The mobile robot variables following an 8-shaped trajectory of case 4 defined in Table 3.3.

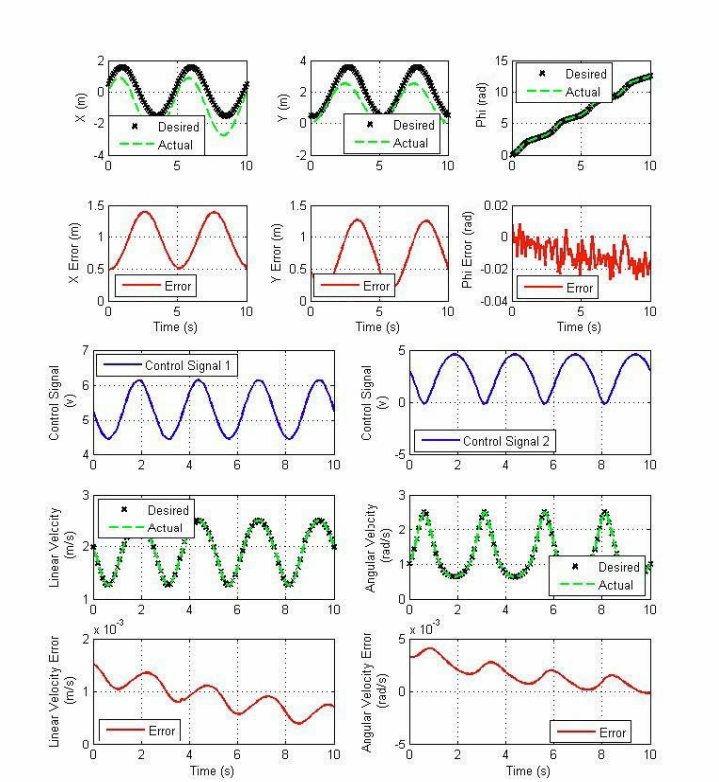

Figure 3.12: The mobile robot variables following an ellipse trajectory of case 5 defined in Table 3.3.

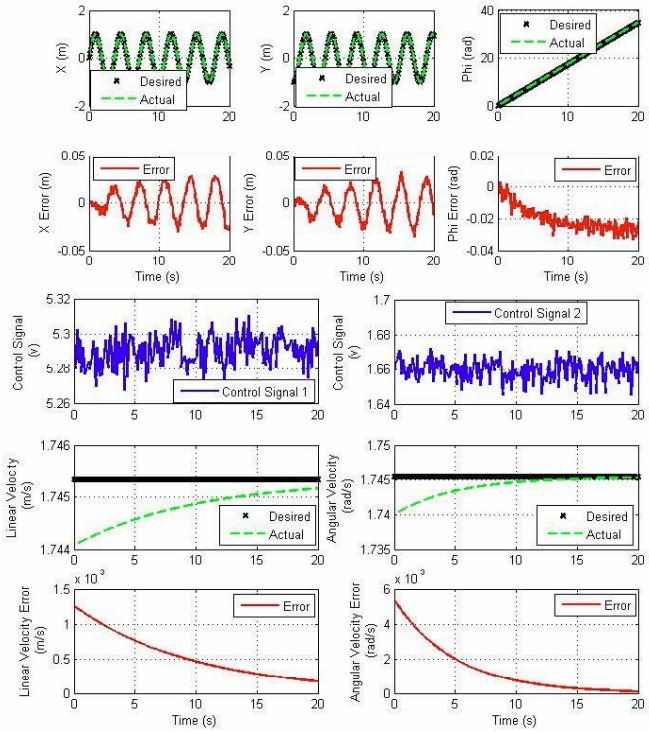

Figure 3.13: The mobile robot variables following a circle trajectory of case 6 defined in Table 3.3.

Case	v_{SSE}	w_{SSE}	TNDS
1	0.2176×10^{-3}	0.9570×10^{-3}	201
2	0.1894×10^{-3}	0.4706×10^{-3}	101
3	0.1895×10^{-3}	0.4713×10^{-3}	101
4	0.2177×10^{-3}	0.9561×10^{-3}	201
5	0.0921×10^{-3}	0.4034×10^{-3}	101
6	0.0782×10^{-3}	0.7356×10^{-3}	201

Table 3.3: Controller errors corresponding the defined trajectory cases of Table 3.2, where v_{SSE} is the sum square error corresponding to the linear velocity of the mobile robot, w_{SSE} is the sum square error corresponding to the angular velocity of the mobile robot, and TNDS denotes the total number of data samples.

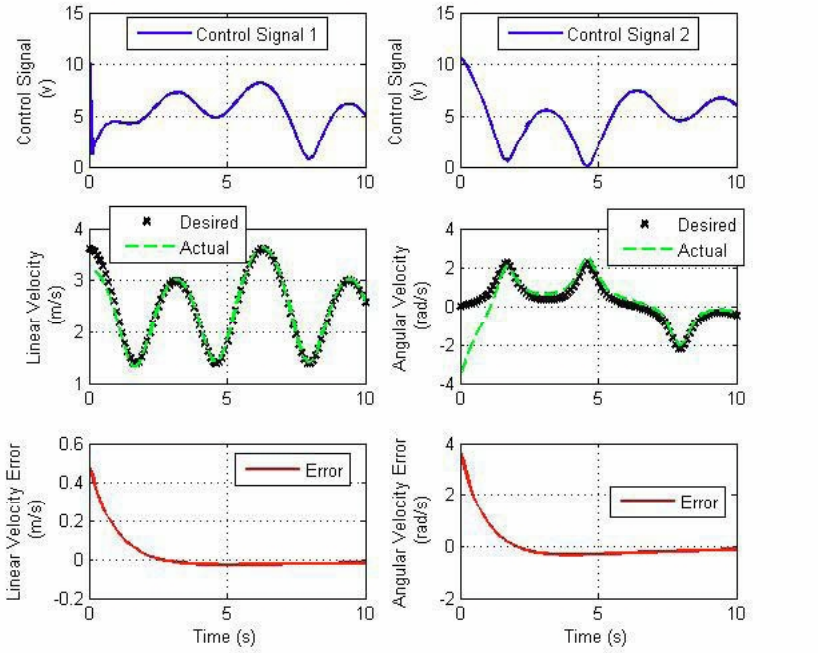

Figure 3.14: The mobile robot variables following an 8-shaped trajectory of case 1 defined in Table 3.4.

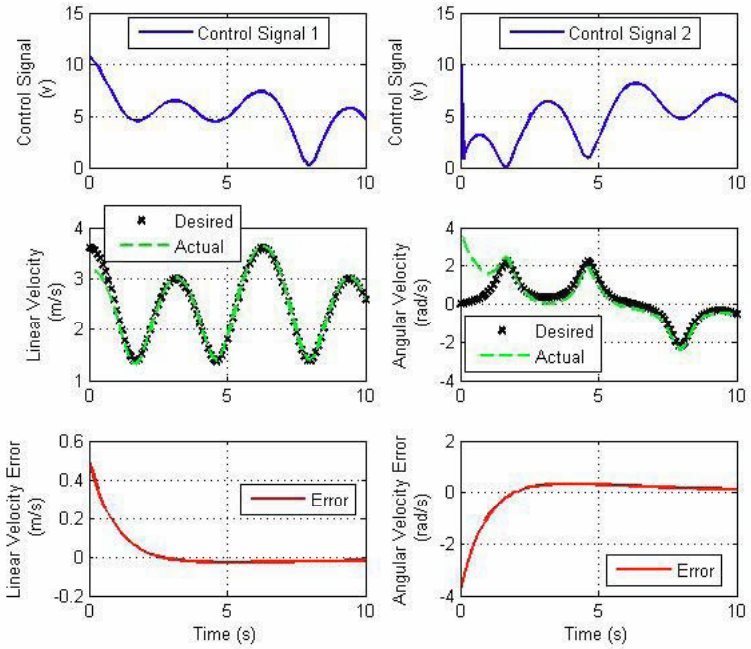

Figure 3.15: The mobile robot variables following an 8-shaped trajectory of case 2 defined in Table 3.4.

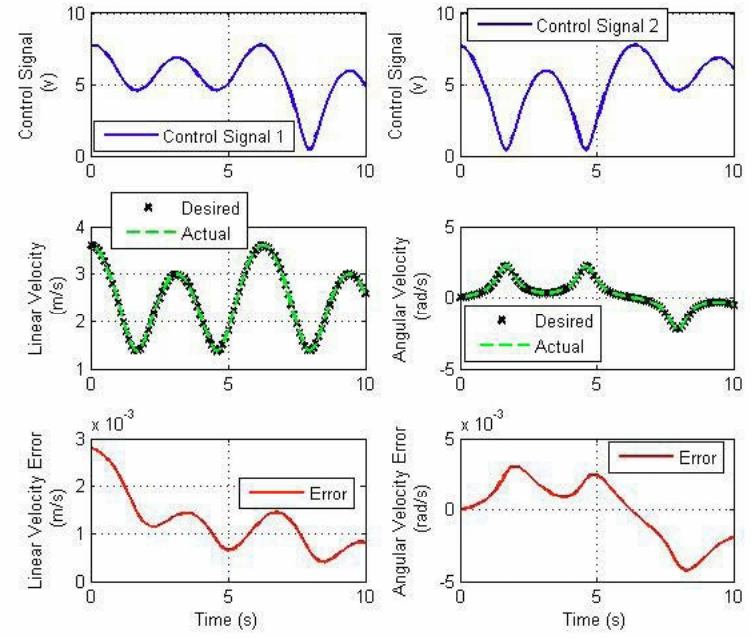

Figure 3.16: The mobile robot variables following an 8-shaped trajectory of case 3 defined in Table 3.4.

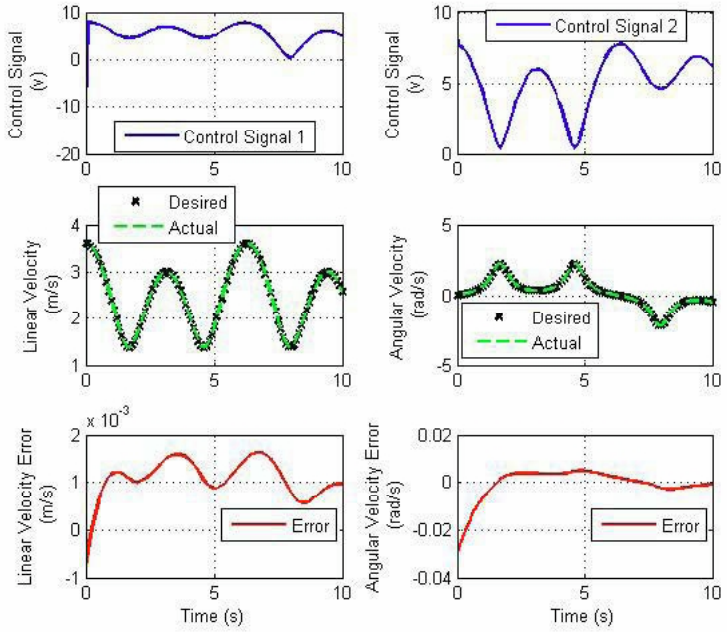

Figure 3.17: The mobile robot variables following an 8-shaped trajectory of case 4 defined in Table 3.4.

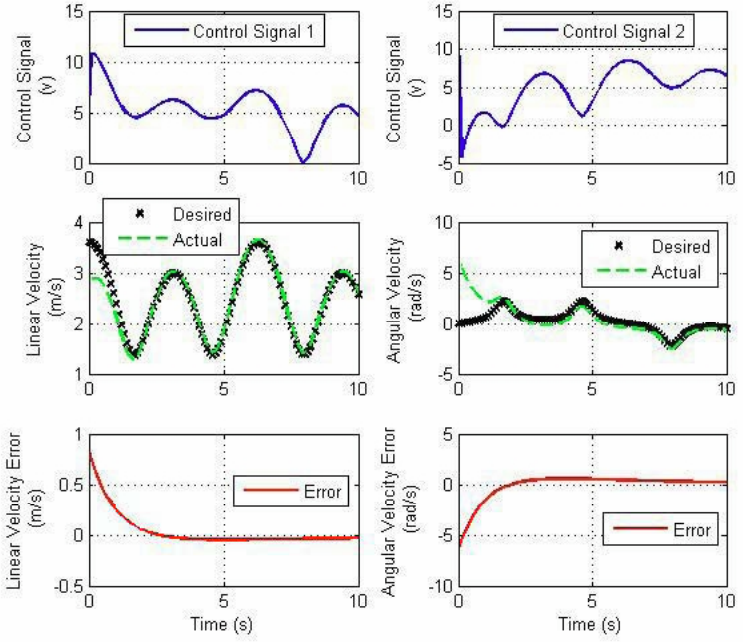

Figure 3.18: The mobile robot variables following an 8-shaped trajectory of case 5 defined in Table 3.4.

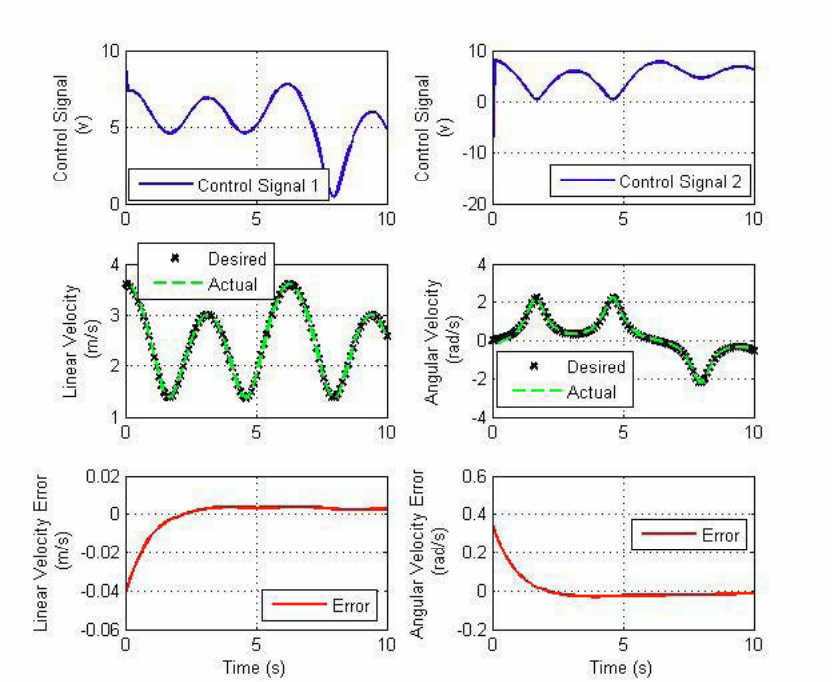

Figure 3.19: The mobile robot variables following an 8-shaped trajectory of case 6 defined in Table 3.4.

Case	Initial Conditions	v_{SSE}	w_{SSE}
1	$x(1)=2.5, x(2)=0, x(3)=0, x(4)=0$	1.1664	63.3228
2	$x(1)=0, x(2)=0.4, x(3)=0, x(4)=0$	1.2284	66.703
3	$x(1)=0, x(2)=0, x(3)=5, x(4)=0$	0.1890×10^{-3}	0.4709×10^{-3}
4	$x(1)=0, x(2)=0, x(3)=0, x(4)=-10$	0.0001	0.0044
5	$x(1)=0, x(2)=-4, x(3)=-0.5, x(4)=0$	3.373	185.609
6	$x(1)=0, x(2)=1.2, x(3)=3, x(4)=6$	0.0083	0.5589

Table 3.4: Controller errors corresponding to the 8-shaped trajectory of case 2 defined in Table 3.2 with different initial conditions, where v_{SSE} is the sum square error corresponding to the linear velocity of the mobile robot, w_{SSE} is the sum square error corresponding to the angular velocity of the mobile robot, and TNDS denotes the total number of data samples.

3.4 Conclusions

In the first section, equations of the robot platform and its actuators are derived. The robot linear and angular velocities are controlled by using MIMO PD controllers. Simulation of the mobile robot show that the suggested controller can control the linear and angular velocities of the mobile robot in different trajectory paths with small error. The simulator will be used in the next chapters to test the performance of the proposed fault detection and identification methods.

Chapter 4

Fault Detection Design of Mobile Robots

4.1 Introduction

In this chapter, by using the data from the simulator described in the previous chapter, the mobile robot is modeled by using a locally linea model (LLM) and a radial basis function (RBF) neural network. The importance of fault detection in mobile robots is presented and different faults that are implemented in the system are introduced. Finally, the implemented faults in the system are shown to be detected. Two methods of adaptive threshold band generations are proposed to improve the fault detection performance and minimize the false alarms. This performance is then compared with a fixed threshold band. To detect faults enjoying improved performance, a moving average technique developed by using a window of three samples is used in order to filter high frequency oscillations of the residual signals.

4.2 Modeling the System

By using the two methods described in Chapter 2, the mobile robot is modeled. To model the system, the training and testing data are chosen as in the cases 1 and 2 defined in Table 3.2. In this thesis, indirect system identification approach is applied, implying that the

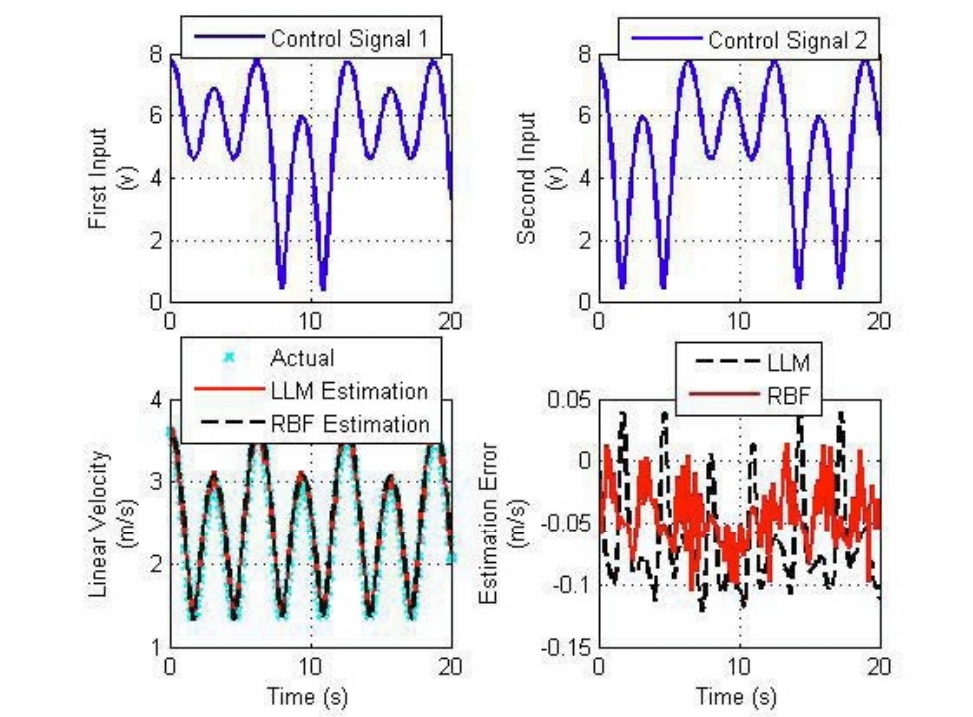

Figure 4.1: The performance of model representative, case 1 as defined in Table 3.2.

control signals are chosen as the model inputs and the linear velocity of the mobile robot is chosen as the output of the model.

It is worthwhile to noting that the system can be trained on-line if a window of data is used instead of all the measured data in the training phase. The dynamic order of $m = 4$ is found to be appropriate for modeling the system. Also, 7 neurons for the LLM model and 40 neurons for the RBF neural networks are used. Figures 4.1 and 4.2 show that the estimation error is low and bounded by using both modeling methods. Also, the models performances in cases 3 to 5 of Table 3.2 are shown in Figures 4.3 to 4.5. These figures show that the output is well modeled and the estimation error is bounded. The sum square estimation error of all cases are summarized in Table 4.1.

The models performances on the data of 8-shaped trajectory with various initial conditions are shown in Figures 4.6 to 4.11. The sum square estimation error of all cases are summarized in Table 4.2.

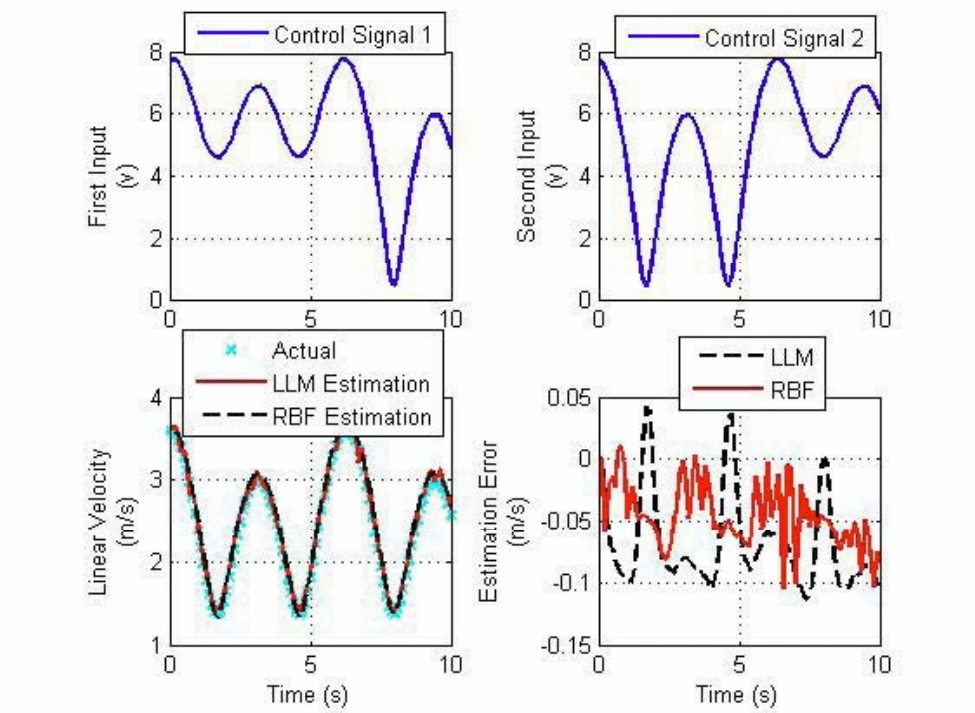

Figure 4.2: The performance of model representative, case 2 as defined in Table 3.2.

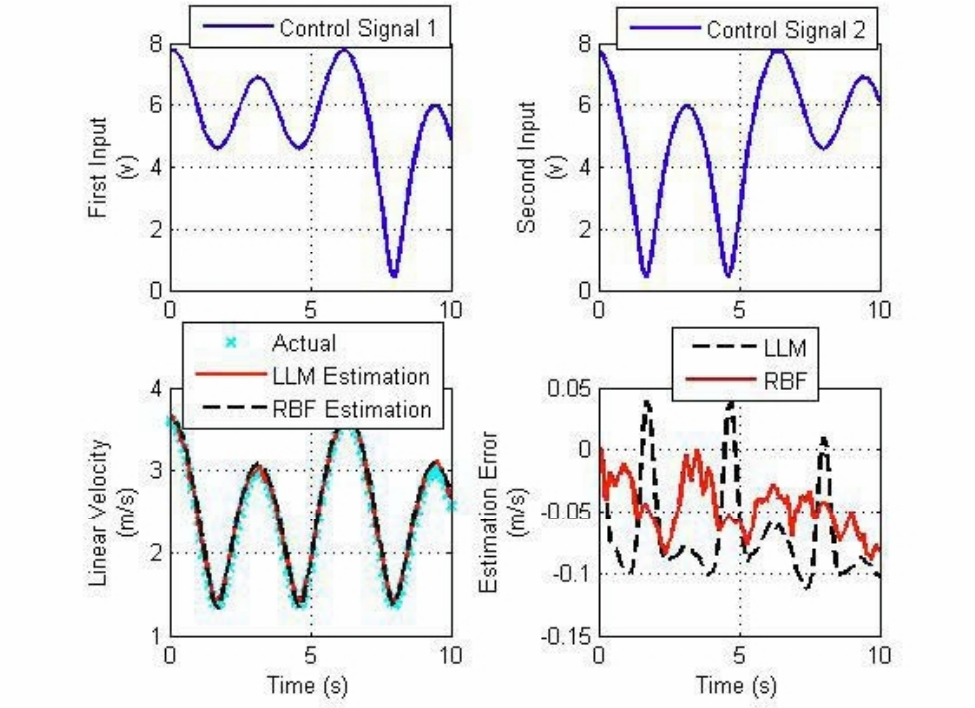

Figure 4.3: The performance of model representative, case 3 as defined in Table 3.2.

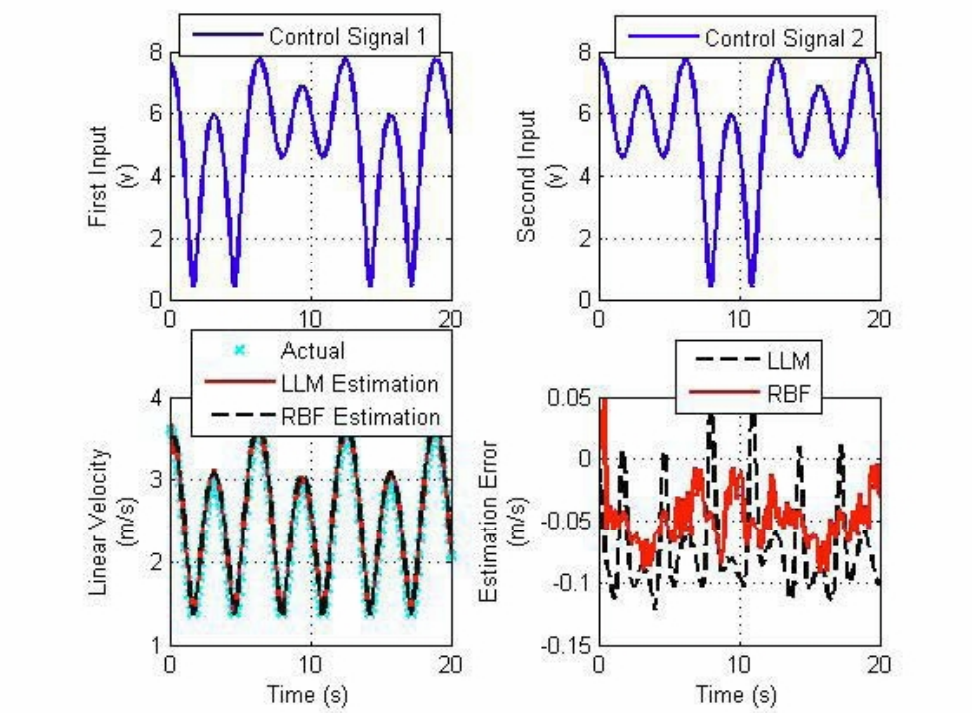

Figure 4.4: The performance of model representative, case 4 as defined in Table 3.2.

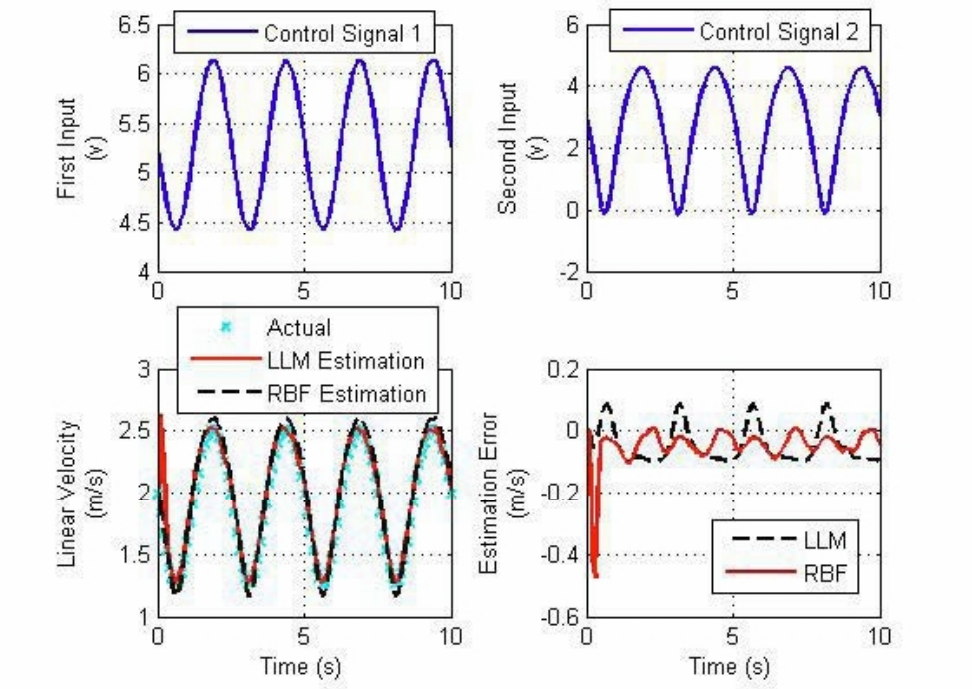

Figure 4.5: The performance of model representative, case 5 as defined in Table 3.2.

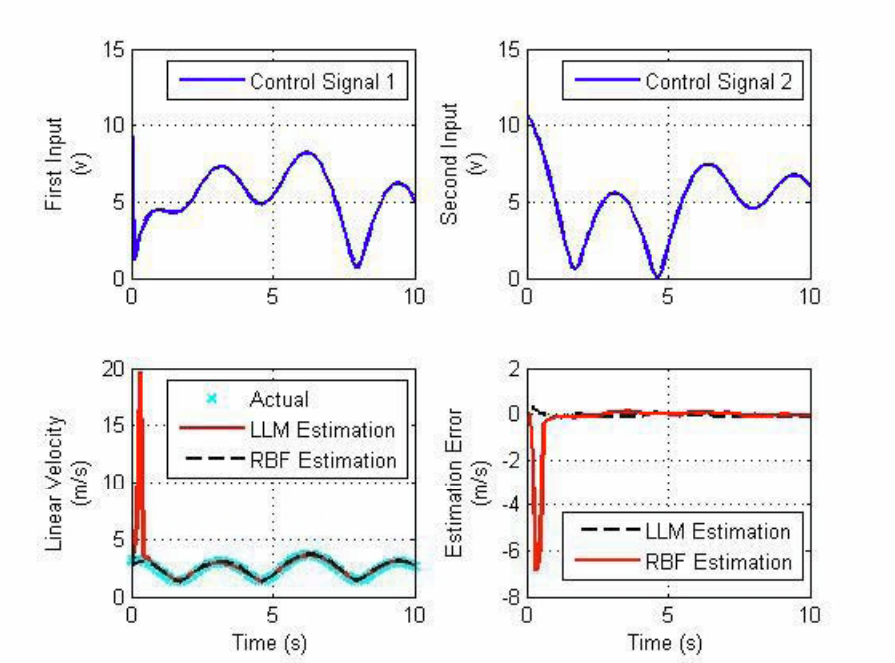

Figure 4.6: The performance of model representative, case 1 as defined in Table 4.2.

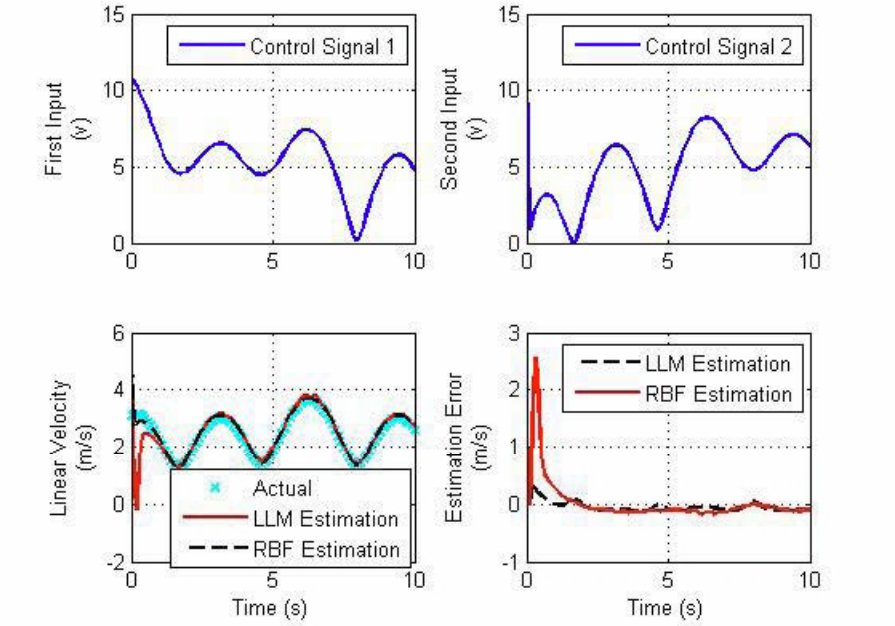

Figure 4.7: The performance of model representative, case 2 as defined in Table 4.2.

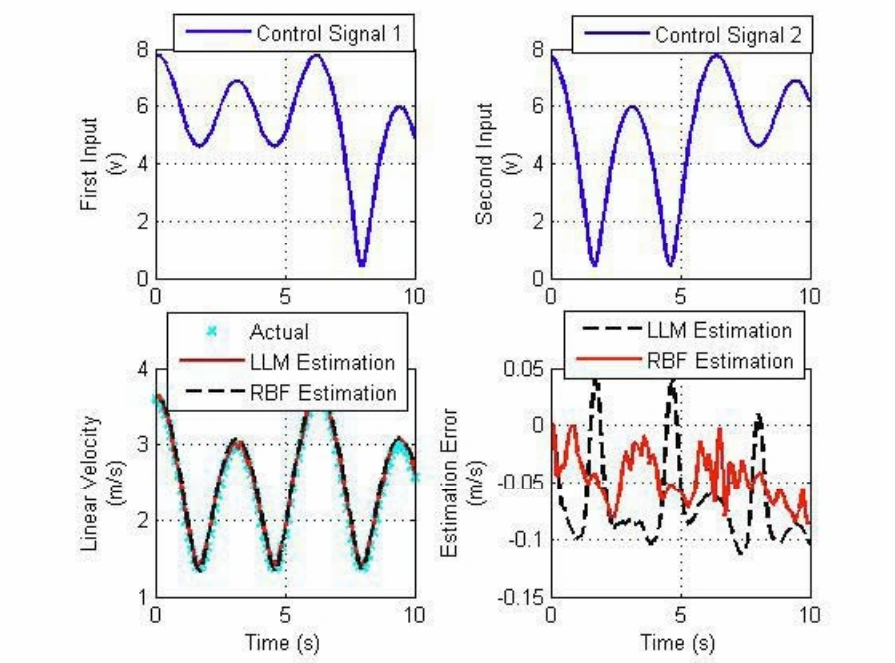

Figure 4.8: The performance of model representative, case 3 as defined in Table 4.2.

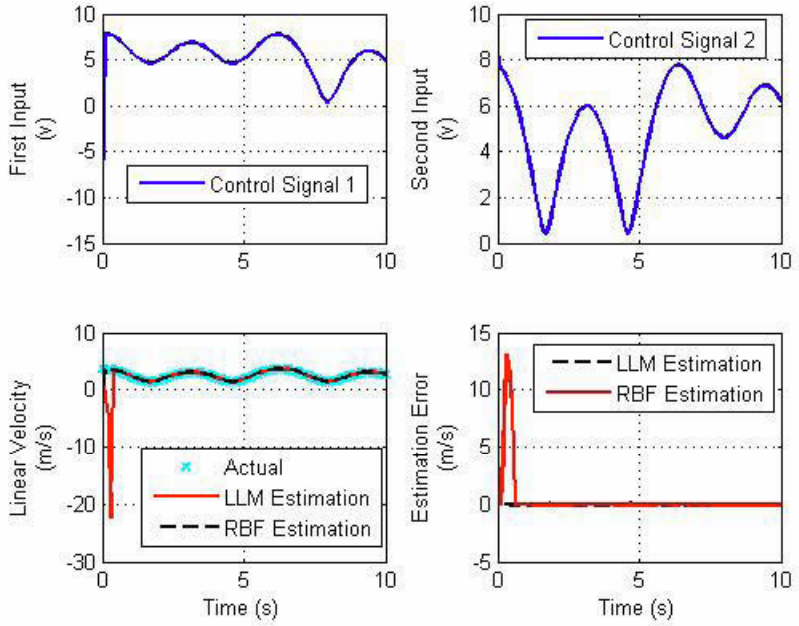

Figure 4.9: The performance of model representative, case 4 as defined in Table 4.2.

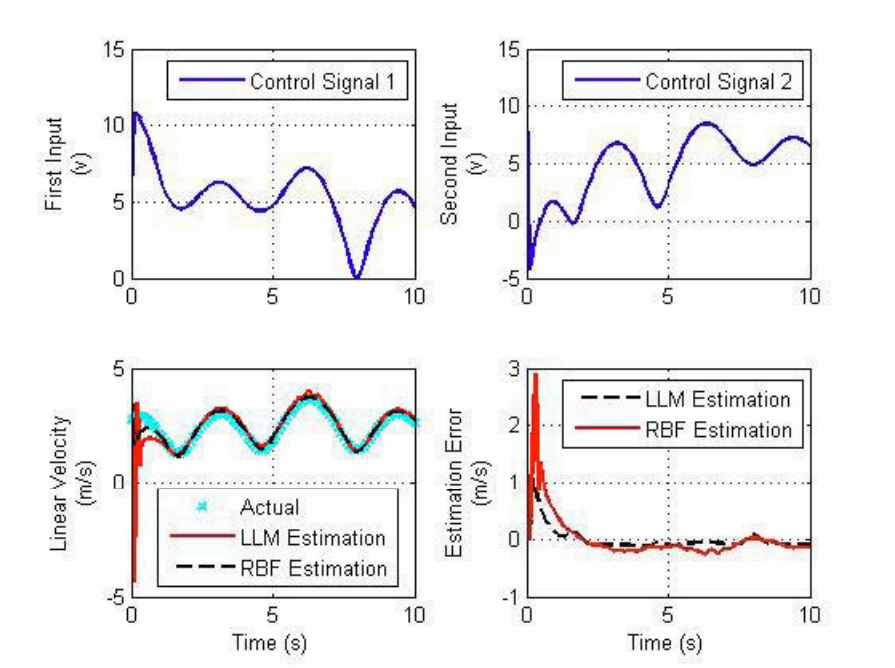

Figure 4.10: The performance of model representative, case 5 as defined in Table 4.2.

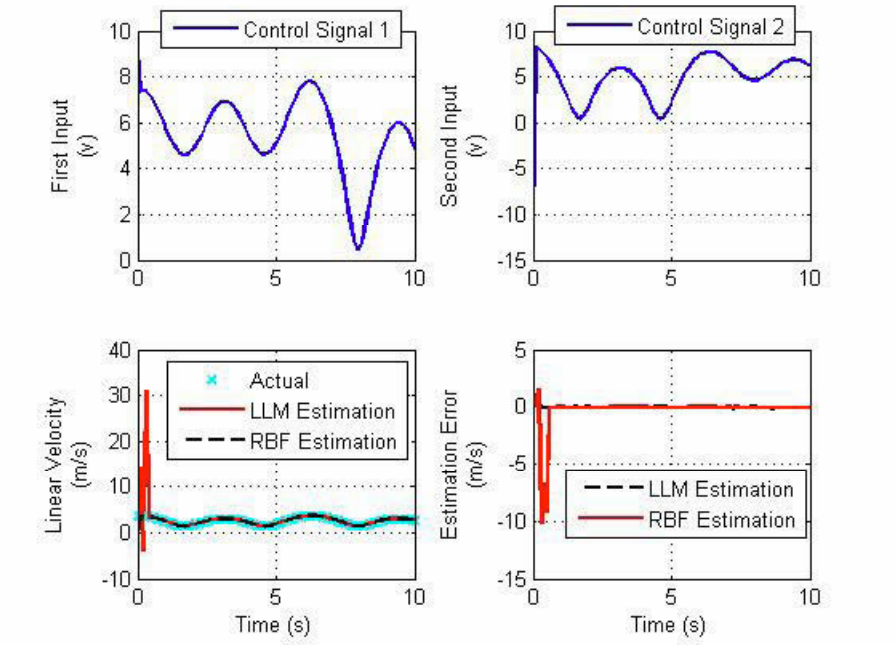

Figure 4.11: The performance of model representative, case 6 as defined in Table 4.2.

Case	LLM_{SSE}	RBF_{SSE}	TNDS
1 (Train data)	0.5361	1.24	201
2 (Test data)	0.2736	0.6221	101
3	0.2908	0.6245	101
4	0.5743	1.319	201
5	0.6711	0.5639	101
6	0.5814	0.0247	201

Table 4.1: LLM and RBF estimation errors in modeling the linear velocity of the mobile robot as defined in cases of Table 3.2. LLM_{SSE} is the sum square error corresponding to the estimation error of LLM model, RBF_{SSE} is the sum square error corresponding to the estimation error of RBF model, and TNDS denotes the total number of data samples.

4.3 Faults in a Mobile Robot

Due to the importance of reliability and safe operation of mobile robots, especially those in unknown environments such as in planetary exploration, fault detection and diagnosis of these systems are becoming more important. An example of a crashed and a failed system without an FDI module is discussed in [84]. In 1994, Dante II was deployed in a remote Alaskan volcano to demonstrate remote robotic exploration. While ascending out of the crater, it encountered steep slope and cross-slope conditions that changed the system dynamics. Failure to identify the mentioned issue resulted in the robot falling on its side. Dante II was later saved by a helicopter because it did not have a fault tolerant system. Several researchers have recently investigated mobile robot's reliability and fault tolerance [85]-[88], [94]-[98]. A recent practical example is from the Mars Exploration Rover, Spirit. The fault was a lubricant leak in one of the wheels which was detected by a large team of engineers who barely analyzed rover telemetry each night. Using a fully autonomous rover, this kind of fault would be detected independent of the ground station operator [85].

Case	Initial Conditions	LLM_{SSE}	RBF_{SSE}
1	$x(1)=2.5, x(2)=0, x(3)=0, x(4)=0$	130.37	0.7195
2	$x(1)=0, x(2)=0.4, x(3)=0, x(4)=0$	14.454	0.89
3	$x(1)=0, x(2)=0, x(3)=5, x(4)=0$	0.2767	0.6197
4	$x(1)=0, x(2)=0, x(3)=0, x(4)=-10$	395.191	0.616
5	$x(1)=0, x(2)=-4, x(3)=-0.5, x(4)=0$	18.058	3.56
6	$x(1)=0, x(2)=1.2, x(3)=3, x(4)=6$	235.51	0.612

Table 4.2: LLM and RBF estimation errors in modeling the linear velocity of the mobile robot in the trajectory of case 2 as defined in Table 3.2 with different initial conditions. LLM_{SSE} is the sum square error corresponding to the estimation error of LLM model, and RBF_{SSE} is the sum square error corresponding to the estimation error of RBF model.

To monitor failures in mobile robots, in 2004, Carlson *et al.* [46] collected failure types and frequency data from fifteen mobile robots operating under a variety of environments. In [46], Carlson *et al.* present taxonomy of mobile robot failures, according to the source of failure, including physical and human failures. Physical failures have occurred, on average, once every 24 hours and human failures have occurred once every 17 minutes of robot usage time over an additional 1082 hours of robot usage. The authors in [46] subdivided physical and human failures into classes based on common systems that are found in all the robot platforms. Figure 4.12 depicts these classes with probabilities of failure corresponding to each class. In below, a brief explanation about the potential faults in each subsystem of the mobile robot and a few studies that are done about them are introduced.

Power Systems and Effectors: Wheeled mobile robots are built with their wheels' drive machine; motors. Dependent on the robots' desire design, technicians use DC motors for motion control. Although the design of DC motors has improved a lot during the past few years, they can suffer from failures. Especially the overload and the overheating can damage the coils which result in lower performance of the motor. Figure 10 shows that 40% of physical failures in mobile robots are related to the power systems and effectors. During the past few years, a number of research have done on fault diagnosis of the DC

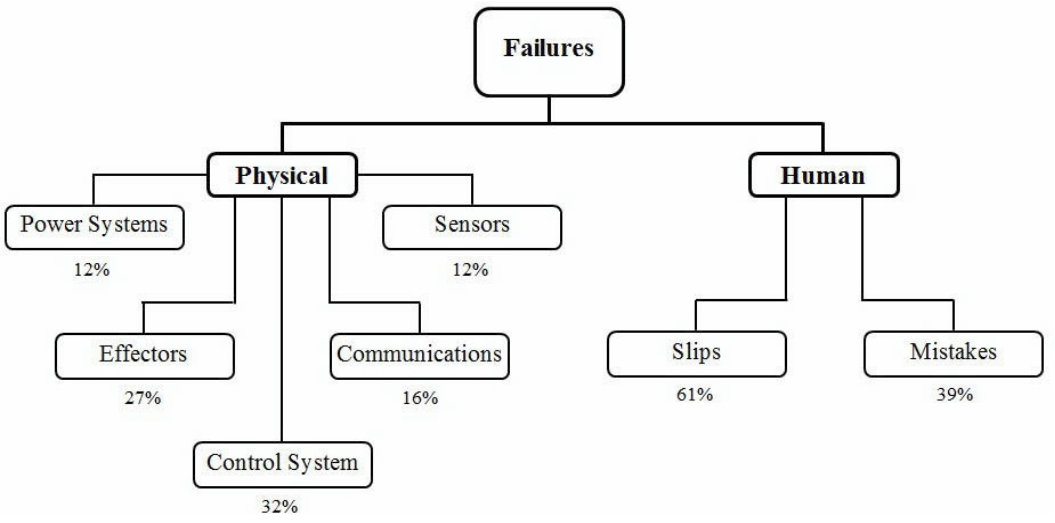

Figure 4.12: A taxonomy of mobile robot faults [46].

motors [89]-[91].

The main fault in the motor of the mobile robot is the change of resistance which may happen for several reasons [80]. During the process, a coil of the motor might become interrupted. Usually in this situation, the motor keeps on tuning with lower speed and less torque and in the closed-loop operation the desired position of the actuator is still reached. One may not realize that a crucial fault has occurred as the impact may be covered up by the robust control loop, although this issue needs to be diagnosed as soon as possible [89]. Other possible faults in DC motors are change in the ratio of the inertia [80], change in the motor inductance [90] and also broken motors and gears [85].

In several studies, actuator faults are implemented as locked in the place of actuator and loss of effectiveness of actuator signals [30, 35]. Such actuator faults have attracted great deal of interests not only for mobile robots, but also for other systems [92]. These are common faults which are usually considered as case studies [93].

Power systems such as battery cells, wires, switches and safety circuits are more reliable than the other systems since they are least affected by the environmental hazards.

Control System: The most common sources of physical failures in the Carlson's

study are the control systems. In most of these cases the robot was unresponsive and the solution was to cycle the power as the source and cause of these problems remains unknown. Other examples of the control system failure include a corrupted hard drive and electrical problems of the controller. To reduce these kinds of failures, design of software-based controllers is suggested. Also, further attention on the methodology for controller design is required to reduce such failures.

Communications and Sensors: Using communication systems have become more common due to the increased use of wireless robots over the past few years. The main failure in this component is the communication loss. This is due to the fact that the manufacturers purchase massly produced sensors. Conversely, the robot's effectors, control systems, and power systems are custom built.

Although faults in sensors have attracted a lot of interest in the FDI of mobile robots and even other systems [32, 47, 87], according to the Carlson's study, sensing and power failures are the least common sources of failure in robots. Sensor faults are implemented as loss of calibration, bias, and locked in the place faults. Among the sensors used in mobile robots one can name gyros, encoders, cameras, and sonars, as the most common failed sensor are the cameras.

Human mistakes: The studies recorded in [46] contain the number and types of failures in human mistakes that are encountered as well as the duration of the tasks performed. Through automation of mobile robots the human mistakes can be reduced in some application domains as robots are required to operate without human intervention.

Slips: Slips and faults that are related to the wheels are categorized in the domain of failures related to humans in the Carlson's study [46]. Faults such as slips and stuck in the wheels have been studied a lot during the recent years [94]-[98] and as shown in Figure 4.12, they have high percentage of failures. In [97], the analysis of a slip fault is performed based on the following equation,

$$\lambda = (\frac{V_w - V_r}{V_w}) \times 100 \tag{4.1}$$

where V_w and V_r are the velocities of the wheel and the robot, respectively. λ is the wheel slip ratio which is defined as the percentage of robot linear velocity. Specifically, nonzero λ means wheel slippage has occurred. In the study of the traction control system, this value is regulated to optimize the robot traction performance. The slip factor can be easily obtained if the two velocity parameters (V_w, V_r) are measurable [97].

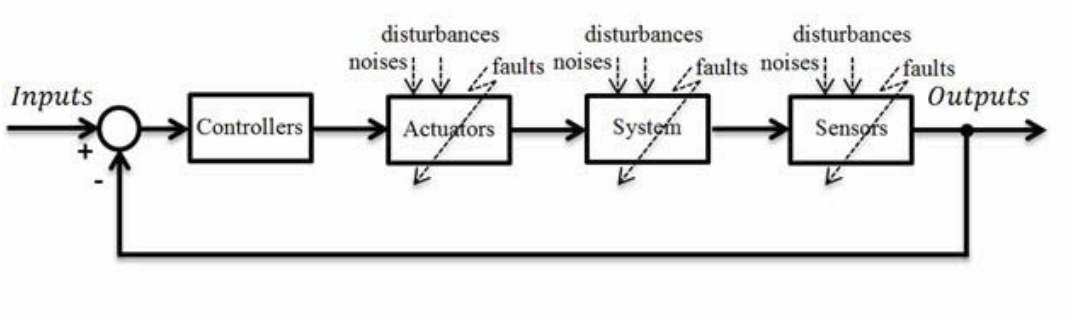

Figure 4.13: A typical feedback control system with potential faults [15].

Figure 4.13 shows a typical feedback control system with potential faults [15]. For a mobile robot platform, faults can be enumerated as follows:

Actuator: Faults in the motor of a mobile robot are the most common actuator faults. Changes in the armature resistance and inductance have been extensively studied. Loss of effectiveness and locked in place of actuator signals have been studied for mobile robots in [80], [89]-[91].

System: Faults in the wheels of the system are the most common system faults in the mobile robots. Wheel slippage, stuck and changes in the radius of the wheel have also been studied [94]-[98].

Sensors: Faults can occur in all sensors of a mobile robot. Most common sensors that are used are the encoders, gyros, sonars and cameras. Sensor bias, locked in place and loss of calibration are the common sensor faults [32, 47, 87].

4.3.1 Implemented Faults on the Mobile Robot System

In this subsection, implemented faults on the simulator are described. Actuator and system faults are implemented in different simulation scenarios when the mobile robot is following the 8-shaped trajectory. Actuator faults are implemented abruptly after the 5^{th} second. These faults consist of loss of control signal effectiveness and change in the right motor resistance.

Loss of control signal effectiveness is the most common actuator fault in case studies. This can occur due to several problems in the actuator. In this thesis, control signal is decreased abruptly after the 5^{th} second by 5 (Figure 4.14) and 20 percent. Also, resistor faults are implemented in the actuator. As described earlier, resistor faults are the most common faults in DC motors. Resistor might decrease when it is partly or totally short circuited and also it might increase in value due to skin effects or high temperatures. In simulations, 0.1Ω decrease (Figure 4.15) and 0.1Ω increase of the right motor resistor are studied.

The other kind of fault that is implemented in the system in the 5^{th} second incipiently is by reducing the left wheel the radius of mobile robot for $0.5cm/s$ (Figure 4.38) and $0.2cm/s$. This fault can occur in mobile robots due to deformations, flat tires, broken spokes, etc. [86]. The final system fault is the slippage fault which is implemented intermittently in the $4-5^{th}$ and $7-8^{th}$ second of the simulation for slip ratios of $\lambda = 3$ (Figure 4.17) and $\lambda = 5$. In these faults two wheels are imagined to slip with the same slip ratio. All the faults considered in this thesis are summarized in Table 4.3.

Figure 4.14 - Figure 4.17 show that the implemented faults on the system do not affect the position and velocities and their effects are compensated by the passive robust controller.

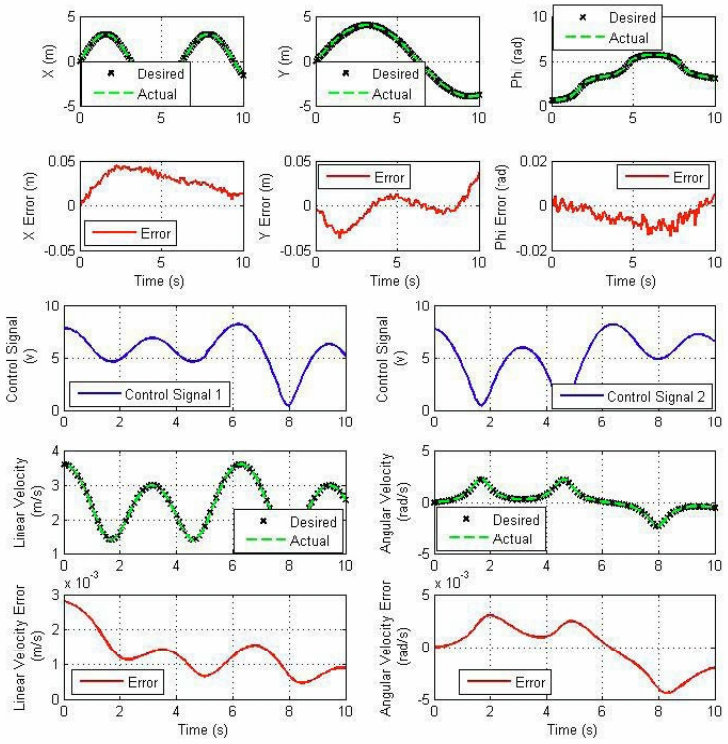

Figure 4.14: Robot state variables when a 5% actuator loss occurs after the 5^{th} second.

4.4 Adaptive Threshold Generation with Local Linear Models of the System

In this subsection, a new concept for generating a threshold band is presented. This method, in contrast with standard methods of developing fixed threshold bands is related to the estimation error under healthy conditions, by suggesting an adaptive threshold band with respect to the maximum error of the estimated output for each local linear model. In this algorithm, local threshold values should be determined for each LLM or neuron. For this purpose,corresponding to the maximum contribution of each LLM (the largest Φ_i in Equation (2.3)), maximum and minimum errors will be used to produce upper and lower local thresholds for the local model. In other words, for each neuron (locally linear model),

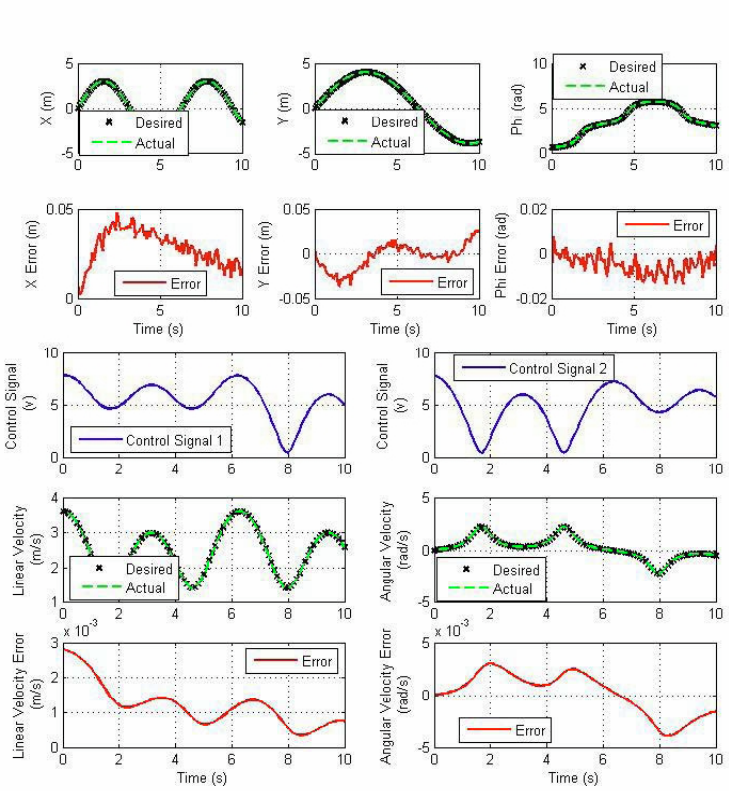

Figure 4.15: Robot state variables when a 0.1 Ω of right motor resistor decreases after the 5^{th} second.

specific threshold values can be found instead of a fixed upper and lower thresholds for the entire model. Specifically, local maximum and minimum errors are used to select the threshold bands instead of using the maximum and the minimum error of the entire model. Consider y and $\hat{y}$ as the system and the LLM output, respectively. The upper and lower threshold values for the i^{th} locally linear models are obtained by using the maximum and the minimum of the estimation error when the i^{th} locally linear model is selected, that is

$$when\ \Phi_i > \Phi_j\ (j = 1,2,...,m,\ j \neq i) \rightarrow \begin{cases} T_i(upper) = maximum(y - \hat{y}) \\ T_i(lower) = minimum(y - \hat{y}) \end{cases} \tag{4.2}$$

The above implies that when a local model's validity function is larger than the other validity functions, the maximum and the minimum estimation errors can be considered as

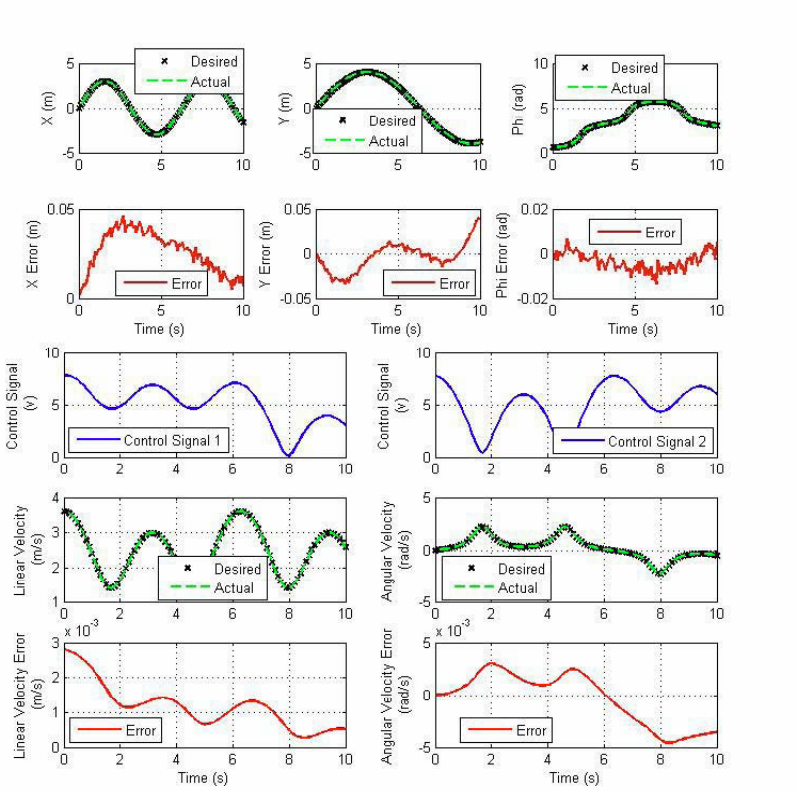

Figure 4.16: Robot state variables when the left wheel radius decreases at the rate of $0.5\,cm/s$ after the 5^{th} second.

the threshold bands of that local linear model. Therefore, T_is are the thresholds associated with the local linear models.

After obtaining local thresholds, for the real-time processes, the validity functions Φ_is for each model are calculated by using Equation (2.3) to estimate the system output. The computed Φ_is specify the contribution of the locally linear models in the overall output

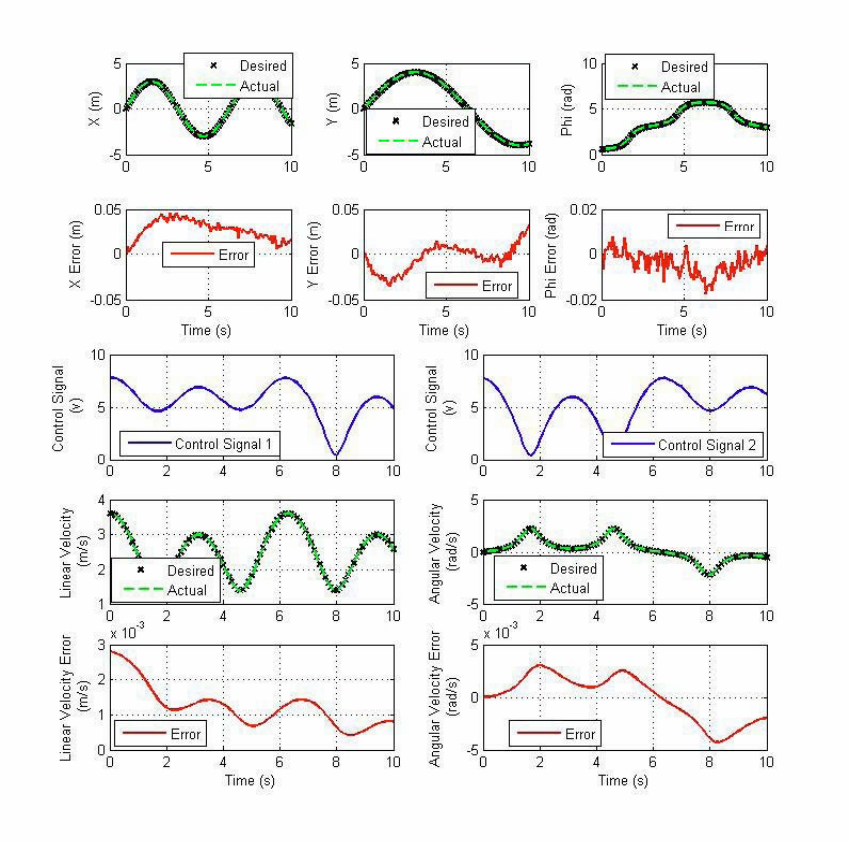

Figure 4.17: Robot state variables when the wheels slip at the ratio of $\lambda = 3$ in the $4-5^{th}$ and $7-8^{th}$ seconds.

estimation. Furthermore, they can be used to generate adaptive threshold bands as follows:

$$\begin{aligned} T_U &= Upper\ Adaptive\ Threshold\ Band = \sum_{i=1}^{M} \Phi_i T_i(upper) \\ T_D &= Lower\ Adaptive\ Threshold\ Band = \sum_{i=1}^{M} \Phi_i T_i(lower) \end{aligned} \tag{4.3}$$

Consequently, the computed threshold values for each neuron after being multiplied by the corresponding validity functions produces the local model thresholds (LMT). The final adaptive threshold band is obtained by summing all the local model thresholds. This concept is illustrated in Figure 4.18. Hence, instead of using a fixed threshold band, an

Case	Status	AL	R_r	r_l	λ
0	No fault	0%	0.71Ω	$10cm$	0
1	Actuator Fault 1 (Abrupt)	20%	0.71Ω	$10cm$	0
2	Actuator Fault 2 (Abrupt)	5%	0.71Ω	$10cm$	0
3	Actuator Fault 3 (Abrupt)	0%	0.61Ω	$10cm$	0
4	Actuator Fault 4 (Abrupt)	0%	0.81Ω	$10cm$	0
5	System Fault 5 (Incipient)	0%	0.71Ω	$0.2cm/s \downarrow$	0
6	System Fault 6 (Incipient)	0%	0.71Ω	$0.5cm/s \downarrow$	0
7	System Fault 7 (Intermittent)	0%	0.71Ω	$10cm$	3
8	System Fault 8 (Intermittent)	0%	0.71Ω	$10cm$	6

Table 4.3: Implemented faults; *AL* is the Actuator Loss, R_r is the Right Motor Resistor, r_l is the Left Wheel Radius and λ is the Wheel Slippage Ratio.

adaptive threshold band is obtained for each input corresponding to its validity function.

As illustrated in Figure 4.18, the generated validity functions are used for both the output estimation as well as for the adaptive threshold generation. Our proposed algorithm can be summarized as provided follows.

In the learning phase, for each locally linear model or neuron we have:

- Determine the section having the largest contribution to the output estimation (the larger the validity function, the more contribution a neuron has) ; and
- Save the maximum and the minimum estimation errors when a given validity function is higher than the others (Equation (4.2)).

In the recall phase, for each input we have:

- Determine the validity functions; and
- Calculate the adaptive thresholds by using Equation (4.3).

The proposed adaptive threshold is obtained so that small faults can be detected at their early stages and so the rate of false alarms is minimized. The derivation in Equation (4.4) shows that the proposed adaptive threshold band is always smaller or equal to that of a fixed threshold (T), if the fixed threshold is chosen as the maximum estimation error

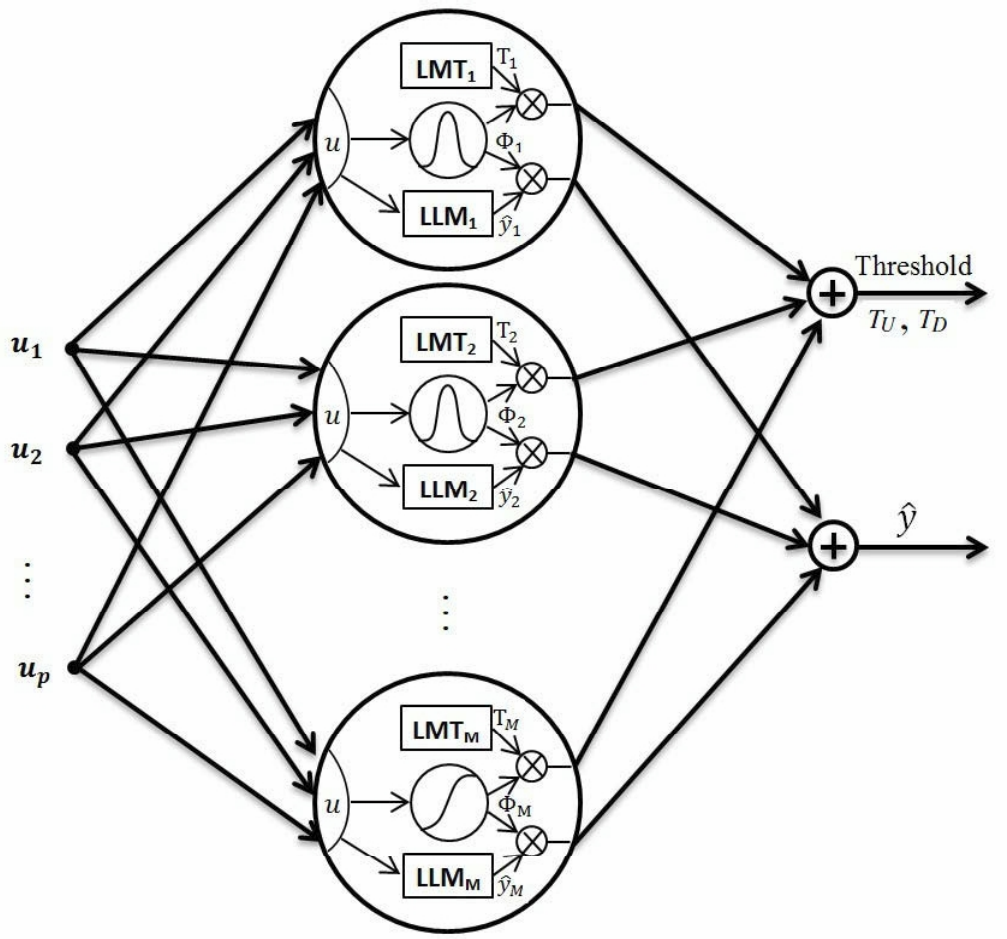

Figure 4.18: The network structure of an LLM model with an LMT adaptive threshold generator.

corresponding to the healthy system.

$$
\begin{aligned}
|T_i| \leq |T| \Longrightarrow |\Phi_1 T_1| \leq |\Phi_1 T|, |\Phi_2 T_2| \leq |\Phi_2 T|, ..., |\Phi_i T_i| \leq |\Phi_i T| \Longrightarrow \\
|\Phi_1 T_1| + |\Phi_2 T_2| + ... + |\Phi_i T_i| \leq |\Phi_1 T| + |\Phi_2 T| + ... + |\Phi_i T| \Longrightarrow \\
|\Phi_1 T_1 + \Phi_2 T_2 + ... + \Phi_i T_i| \leq |\Phi_1 T_1| + |\Phi_2 T_2| + ... + |\Phi_i T_i| \Longrightarrow \\
|\Phi_1 T_1 + \Phi_2 T_2 + ... + \Phi_i T_i| \leq (|\Phi_1| + |\Phi_2| + ... + |\Phi_i|)|T| \Longrightarrow \\
(|\Phi_1| + |\Phi_2| + ... + |\Phi_i|) = 1 \Longrightarrow |\sum_{i=1}^{M} \Phi_i T_i| \leq |T| \quad (4.4)
\end{aligned}
$$

Figure 4.19 shows the implementation of the proposed model on the system. The system is modeled by using locally linear models and the LoLiMoT algorithm as described in Chapter 2. The external dynamics approach without an output feedback is utilized to construct the dynamics to static approximator. The residual r is generated by comparing the output of the system y with its estimations $\hat{y}$. The adaptive threshold bands T_D and T_U are also generated by using the proposed method. If the residual is outside of the adaptive threshold bands, a fault is detected in the system. Otherwise, the system is determined to be healthy. This algorithm can be summarized as follows:

$$r = y - \hat{y} = y - \sum_{i=1}^{M} \Phi_i \hat{y}_i \rightarrow \begin{cases} \textit{No Fault} & \text{if } \sum_{i=1}^{M} \Phi_i T_i(down) \leq r \leq \sum_{i=1}^{M} \Phi_i T_i(up); \\ \textit{Fault Alarm} & \text{otherwise.} \end{cases} \tag{4.5}$$

To summarize, an adaptive threshold band is generated to improve the FDI performance by slightly increasing the computational complexity to the process. To demonstrate the advantages of the proposed method, its performance will be compared with a fixed threshold band and another adaptive threshold band which will be explained in the next part.

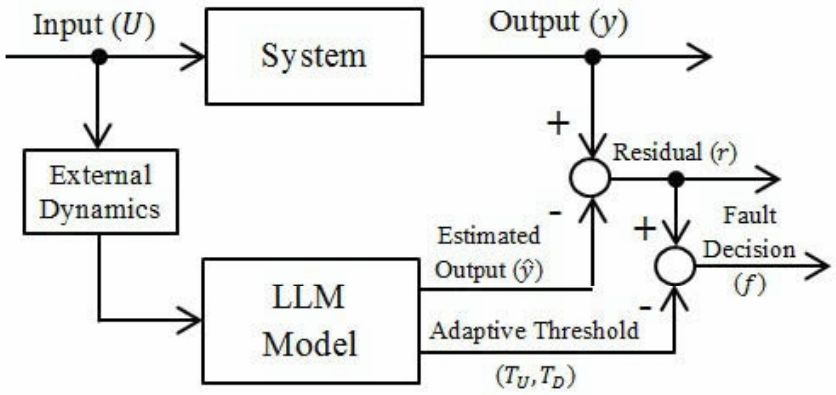

Figure 4.19: Fault detection strategy by using the LLM model and the LMT algorithm. The LLM model box is illustrated in Figure 4.18.

4.5 Adaptive Threshold Generation Using Model Error Modeling (MEM)

Early detection of faults with few false alarms is the main goal of any FD system. For this purpose, a new method of threshold generation is suggested in the previous section which is named LMT. In this section, another method of adaptive threshold band [55] is introduced which will be compared with the LMT and fixed threshold bands. The method is named Model Error Modeling (MEM) approach which is inspired from robust identification theory [99].

A robust learning model should provide a reliable estimate of uncertainties besides an accurate representation of the system. There are two main ideas to deal with uncertainties associated with a given model [55]. Set membership identification [101] or the bounded error approach [102], is the first group which relies on the assumption that the error is unknown but bounded. In this case, robustness is not integrated with the identification process. Another approach is to model the process without robustness considerations first, and then consider robustness as an additional and next step. This usually leads to the least squares estimation and prediction error methods [55]. MEM employs prediction error methods to identify a model from input-output data [99]. The original method has been developed in the frequency-domain [99] which can be simply mapped into the time domain and utilized in robust FD of dynamic systems [55, 100].

To use MEM algorithm in fault detection, first the system should be modeled with an intelligent method of modeling (Figure 2.2) as described in Chapter 2. After modeling the system, the output estimation error should be modeled by using another intelligent learning model. The procedure of training the error model is illustrated in Figure 4.20. Figure 4.20 shows that in the first step, the system model estimates the output of the system. Estimation error r is then calculated by comparing the output of the system y with output of the system model $\hat{y}$. In the next step, parameters and structure of the error

model is optimized by using data set of $\{U_i, r_i\}_{i=1}^{N}$, where U is the input set and N is the total number of data. The error model is used to form uncertainty bands in order to avoid detecting unmodeled dynamics, noise and disturbances as faults. After the error model is trained, the upper and lower adaptive threshold bands are calculated by using the output of the estimated error and its standard deviation as follows (Figure 4.21):

$$\begin{aligned} \textit{Upper Adaptive Threshold band} &= t_\beta \upsilon + \hat{r} \\ \textit{Lower Adaptive Threshold band} &= -t_\beta \upsilon + \hat{r} \end{aligned} \tag{4.6}$$

where $\hat{r}$ is the output of the error model, υ is the standard deviation of $\hat{r}$ and t_β is a constant number between 0 and 1 to add a confidence level to the threshold bands.

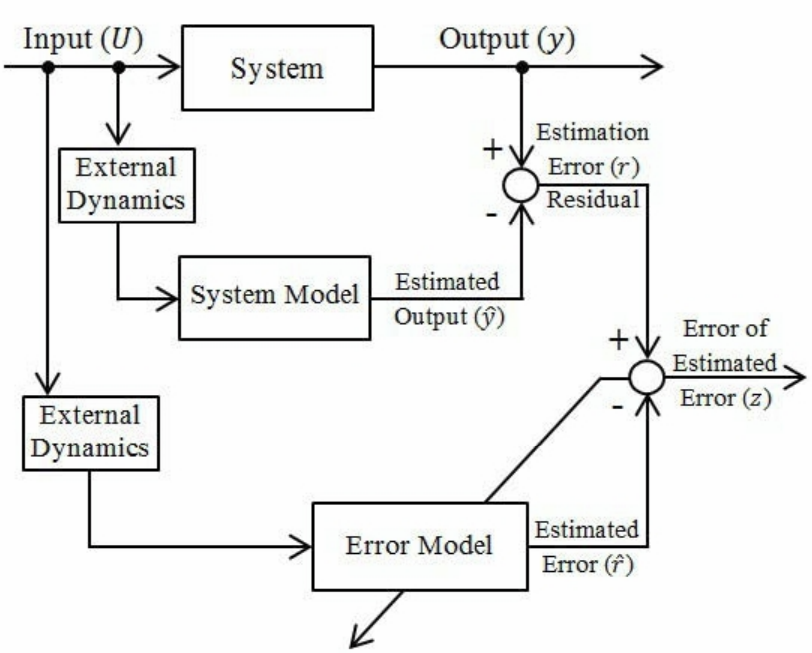

Figure 4.20: Training of the error model.

The designed approach for fault detection using MEM algorithm is shown in Figure 4.21. In this case, output of the system is estimated by using the system model. Residual

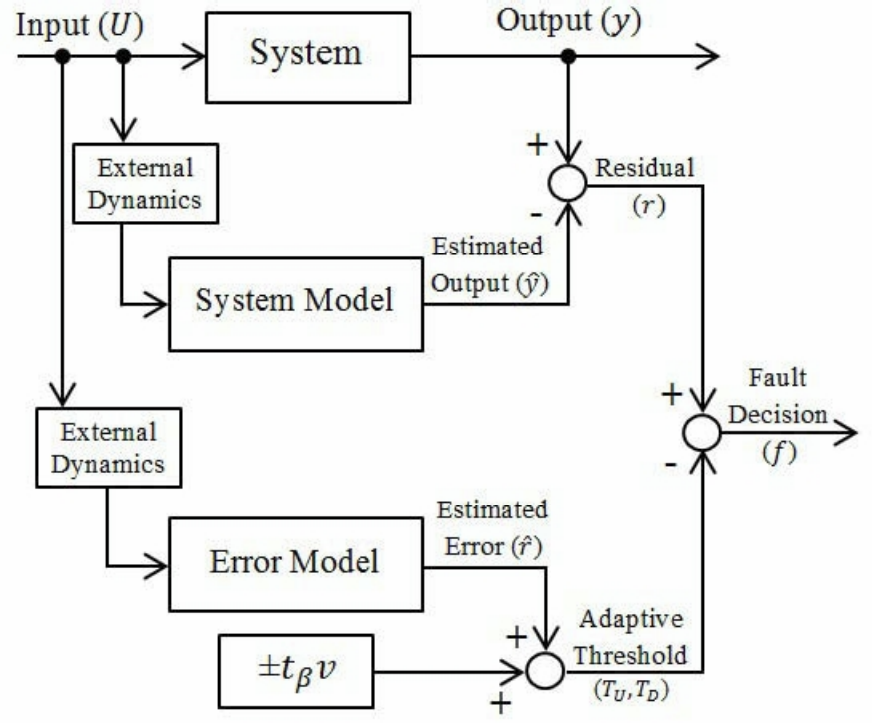

Figure 4.21: Implementation of the MEM algorithm for fault detection.

r is defined as the difference between measured output y and estimated one $\hat{y}$. The adaptive threshold bands T_D and T_U are also generated by using the error model, the constant number t_β and the standard deviation of estimated error υ. If the residual is outside of the adaptive threshold bands ($T_D - T_U$), a fault is detected in the system. Otherwise, the system is determined to be healthy. The fault detection logic is as follows:

$$r = y - \hat{y} = y - \sum_{i=1}^{M} \Phi_i \hat{y}_i \rightarrow \begin{cases} \textit{No Fault}, & \text{if } -t_\beta \upsilon + \hat{r} \leq r \leq t_\beta \upsilon + \hat{r}; \\ \textit{Fault Alarm}, & \text{otherwise.} \end{cases} \tag{4.7}$$

The presented algorithm can be summarized as provided below ([55]).

In the learning phase, we have:

- Model the system and compute the error signal $r = y - \hat{y}$, where y and $\hat{y}$ are the system and the model outputs, respectively.

- Collect the data $\{U_i, r_i\}_{i=1}^{N}$ (U is the the input set) and identify an error model using these data (Figure 4.20). This model constitutes an estimate of the error $\hat{r}$ and it is called the error model.

- Calculate υ as the standard deviation of the estimated error $\hat{r}$.

In the recall phase, we have:

- Use the topology illustrated in Figure 4.21 and calculate adaptive threshold bands from Equation (4.6).

- Use Equation (4.7) to detect faults.

4.6 The Performance of Fault Detection on the Mobile Robot by Using LLM

In this section, simulation results for fault detection of the mobile robot by using LLM model and the proposed adaptive threshold bands and a fixed ones are presented. Fixed threshold bands are developed to determine the minimum and maximum values that each residual signal can reach under a fault free operating condition by using Equations (4.8) as given below:

$$\begin{aligned} T'_U &= \textit{Upper fixed Threshold band} = +s(e) + m(e) \\ T'_D &= \textit{Lower fixed Threshold band} = -s(e) + m(e) \end{aligned} \tag{4.8}$$

where $s(e)$ is the standard deviation of the test data estimation error and $m(e)$ is the average of the test data estimation error. Test data is chosen the same as the test data for the adaptive threshold bands.

Process faults are detected by using fixed threshold bands, LMT adaptive threshold bands (Figure 4.19) and MEM adaptive threshold bands (Figure 4.21). The training procedure of the LLM model which represents the mobile robot system is described in Chapter

2 and Section 4.2. The structure of the second LLM model for MEM algorithm which represents the error model is chosen the same as the first one. The parameters of the error model is optimized using the input and estimation error of the test data.

As described before, testing and validation data are chosen as cases 2 and 5 of Table 3.2. Figures 4.22 and 4.23 (a) and (b) show that the developed model can follow the testing and the validation data with small estimation errors and Figures 4.22 and 4.23 (c) show that the methods have few false alarms corresponding to the healthy data.

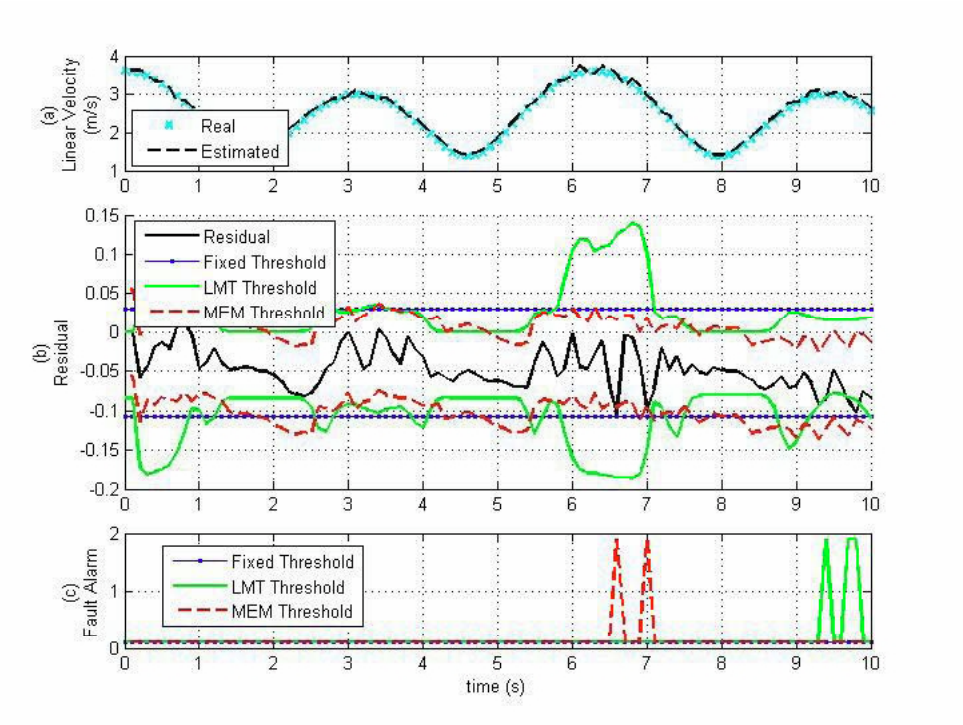

Figure 4.22: The LLM fault detection performance to the testing data, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

Figure 4.24 shows the actuator fault 1 which is a 20% loss of control effectiveness applied after the 5^{th} second. This fault is detected without a delay by using all the threshold bands. This is due to the occurrence of a high severity fault in the system and the accuracy of the LLM model.

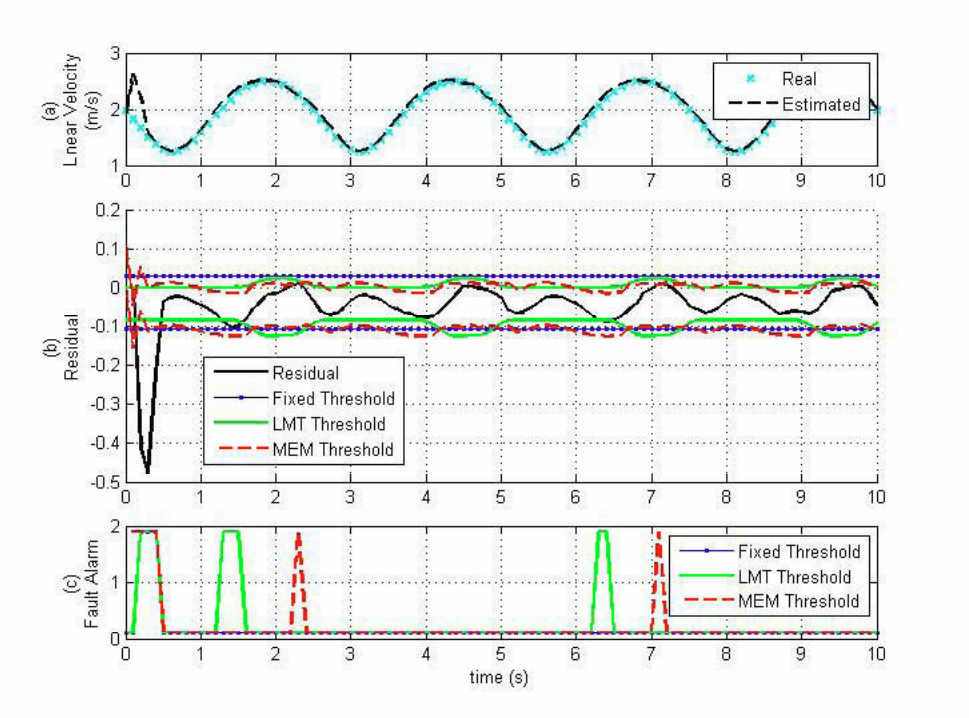

Figure 4.23: The LLM fault detection performance to the validation data, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

Figure 4.25 shows an actuator fault 2 which is a 5% loss of control effectiveness applied after the 5^{th} second. The fault is detected with a 0, a 0.1 and a 0.1 second delay by using the LMT, MEM and fixed threshold bands, respectively. The MEM and fixed threshold bands have detected this permanent fault as an intermittent fault which reduces their reliability.

Figure 4.26 shows an actuator fault 3 which is a $0.1\,\Omega$ reduction of the right motor resistance applied after the 5^{th} second. The fault is detected with a 0.3 second delay by both adaptive threshold bands while the fixed threshold band has detected this permanent fault as an intermittent fault with a 0.4 second delay.

Figure 4.27 shows an actuator fault 4 which is a $0.1\,\Omega$ increase of the right motor resistance applied after the 5^{th} second. The fault is detected with a 0.1, a 0 and a 0.3 second delay by using the fixed, LMT and MEM threshold bands, respectively. The MEM

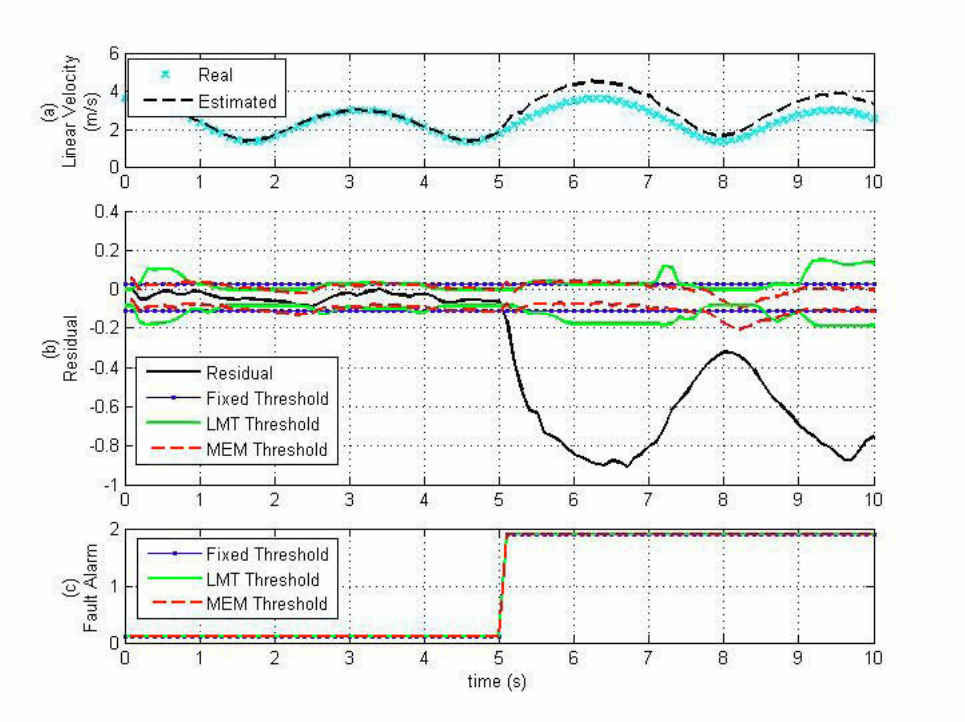

Figure 4.24: The LLM fault detection performance for actuator fault 1: A 20% loss of control effectiveness applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

threshold band has detected this permanent fault as an intermittent fault which reduces its reliability.

Figure 4.28 shows a system fault 5 which is a decrease of the left robot's wheel with the rate of 0.2 cm/s applied after the 5^{th} second. The fault is detected with a 3.5, a 2.3 and a 1 second delay by the fixed, LMT and MEM threshold bands, respectively. The MEM threshold band has detected this permanent fault as an intermittent fault which reduces its reliability.

Figure 4.29 shows a system fault 6 which is a decrease of the left robot's wheel with the rate of 0.5 cm/s applied after the 5^{th} second. The fault is detected with a 0.8, a 1.8 and 0.7 second delay by fixed, LMT and MEM threshold bands, respectively. The fixed threshold band has detected this permanent fault as an intermittent fault which reduces its reliability.

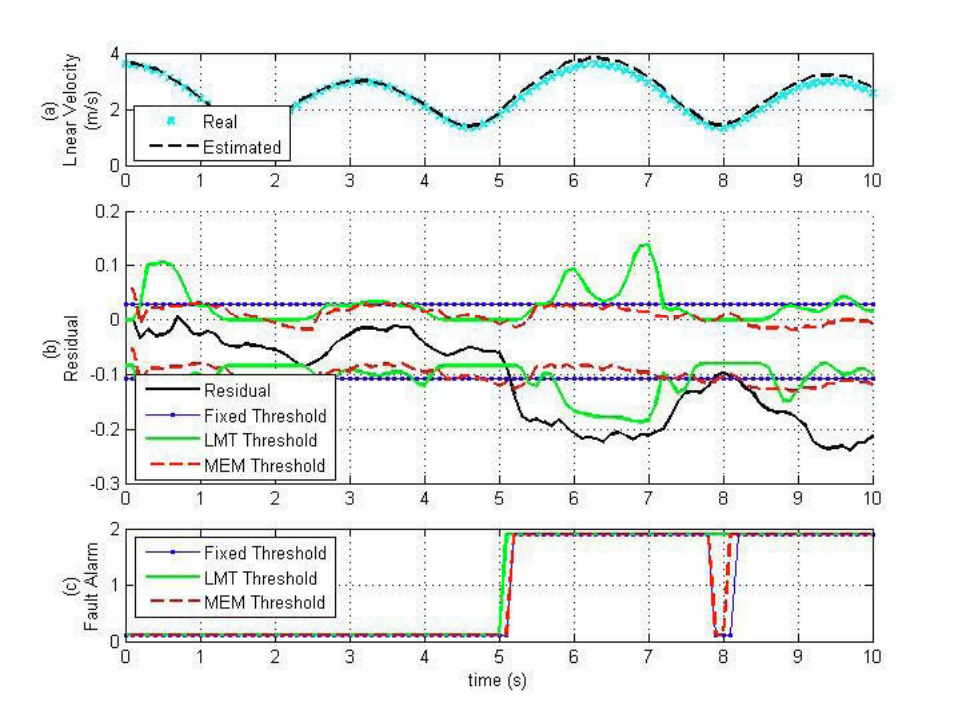

Figure 4.25: The LLM Fault Detection Performance for actuator fault 2: 5% loss of control signal effectiveness applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, (c) Fault alarms generated.

Figure 4.30 shows a system fault 7 which is a wheel slip intermittently with $\lambda = 3$ in the $4-5^{th}$ and $7-8^{th}$ seconds. The faults are detected with a 0.1 and a 0.3 second delay by using fixed and LMT threshold methods and with a 0.2 and a 0.3 second delay by using MEM threshold method.

Figure 4.31 shows a system fault 8 which is a wheel slip intermittently with $\lambda = 6$ in the $4-5^{th}$ and $7-8^{th}$ seconds. The faults are detected with a 0 and a 0.1 second delay by using fixed and LMT threshold methods and with a 0 and a 0.4 second delay by using MEM threshold method.

The delay in detection in all of the fault scenarios are summarized in Table 4.4. Also, the confusion matrix parameters (introduced in Section 1.2.1) corresponding to all threshold methods is provided in Table 4.5.

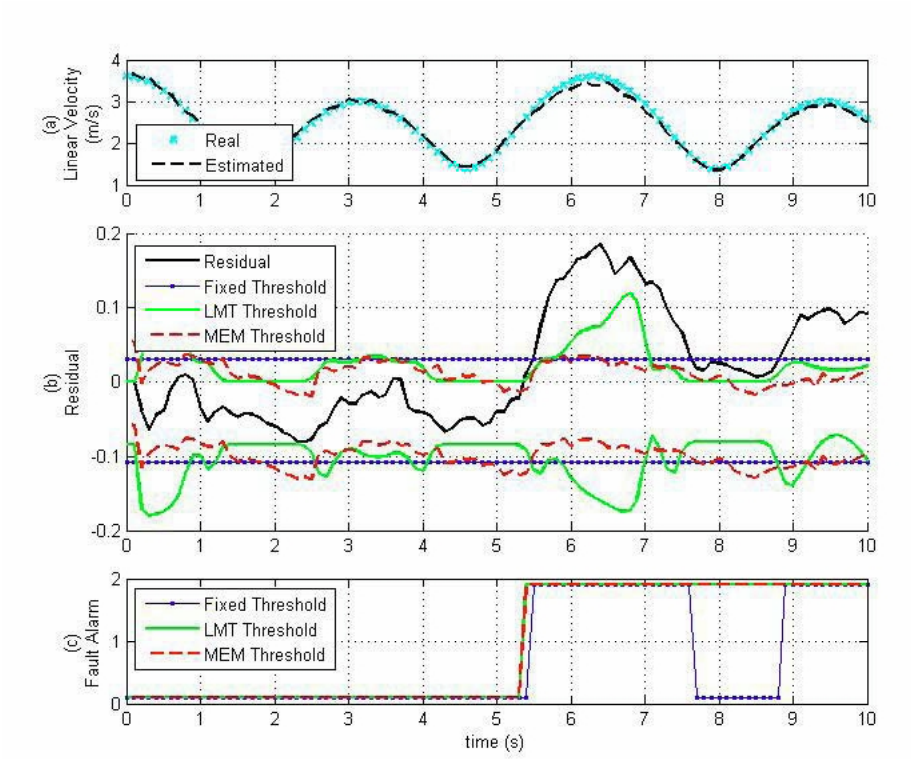

Figure 4.26: The LLM Fault Detection Performance for actuator fault 3: 0.1 Ω decrease of R_r applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, (c) Fault alarms generated.

Simulation results and tables summarizing these results show that the adaptive threshold bands performances in faulty cases are significantly improved over the fixed one. In case 1, due to the accuracy of the LLM model and occurrence of a high severity fault in the system, the fault is detected without delay with all methods. Under other faulty cases that contain low severity faults in the system, the advantages of using the adaptive threshold bands become evident. Corresponding to a total of 340 faulty data, 285 and 283 data are correctly detected by the LMT and MEM methods respectively, while the fixed threshold band has has detected 254 data. This is a meaningful benefit of the adaptive threshold bands. The LMT method has slightly better performance than the MEM method, although it imposes less computational cost to the process while the MEM needs another model to

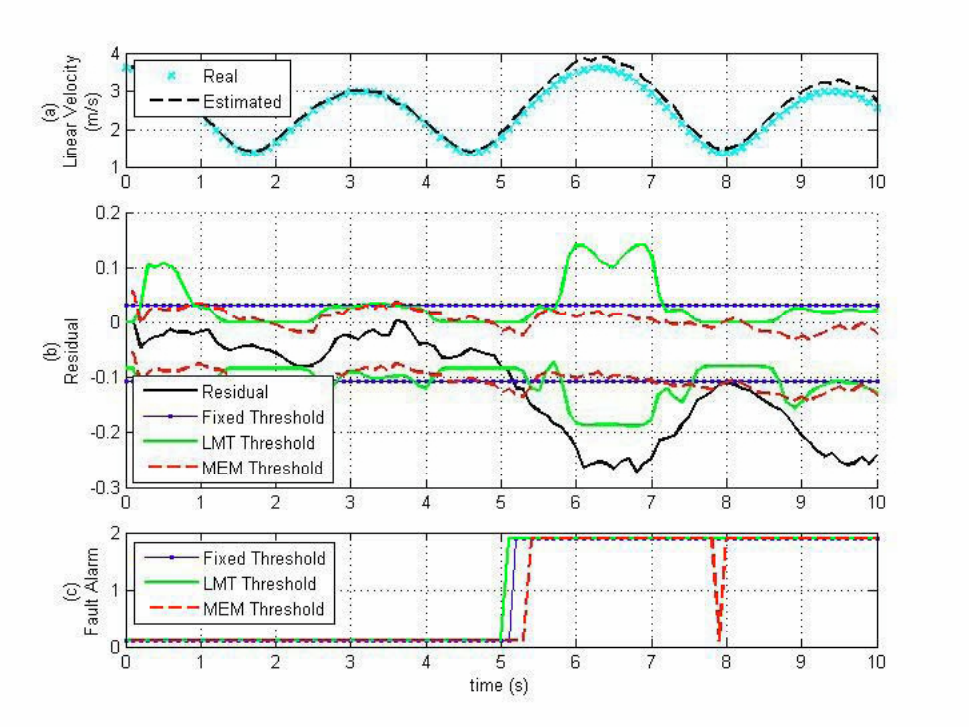

Figure 4.27: The LLM fault detection performance for actuator fault 4: A 0.1 Ω increase of R_r applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

generate the threshold bands.

However by using the LLM models and adaptive threshold bands the faults of the system are well detected; The performance of the fault detection is worth to be compared with another modeling method. In the next section, the performance of the fault detection by using the RBF model is analyzed.

4.7 The Performance of Fault Detection on the Mobile Robot by Using the RBF Network

In this section, an RBF neural network is used to model the mobile robot and its performance is tested on the fault detection of the system. The fault detection is performed by

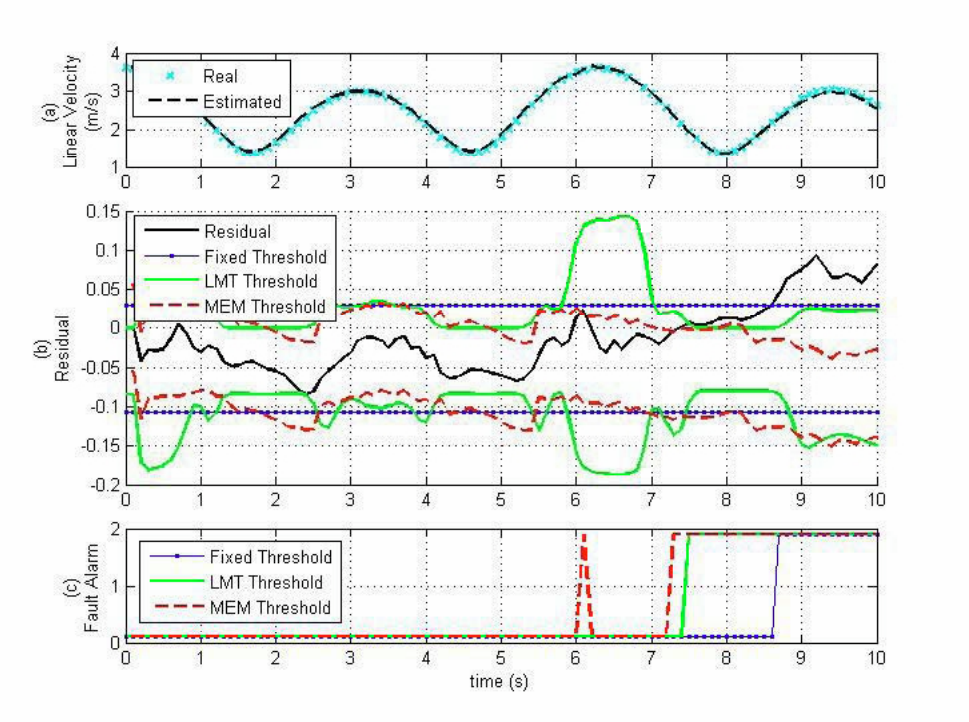

Figure 4.28: The LLM fault detection performance for system fault 5: A decrease of r_l with the rate of $0.2\,cm/s$ applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

using a fixed threshold band and an adaptive threshold band generated with the MEM algorithm (Figure 4.21). The LMT adaptive threshold generation can be extended to the RBF neural network by defining local threshold values for each neuron, but this does not lead to good results. The reason is that LMT uses the variations in the validity functions in generating adaptive threshold band, while in the trained RBF model the validity functions do not change enough to give an appropriate adaptive threshold band.

To implement the MEM algorithm by using the RBF neural networks, two neural networks are trained to detect process faults as described in Section 4.5. The training procedure of the first RBF neural network which represents the system model is described in Chapter 2 and Section 4.2. The structure of the second RBF neural network which represents the error model is chosen the same as the first one. The parameters of the second the RBF neural network are trained with the input and estimated error of the test

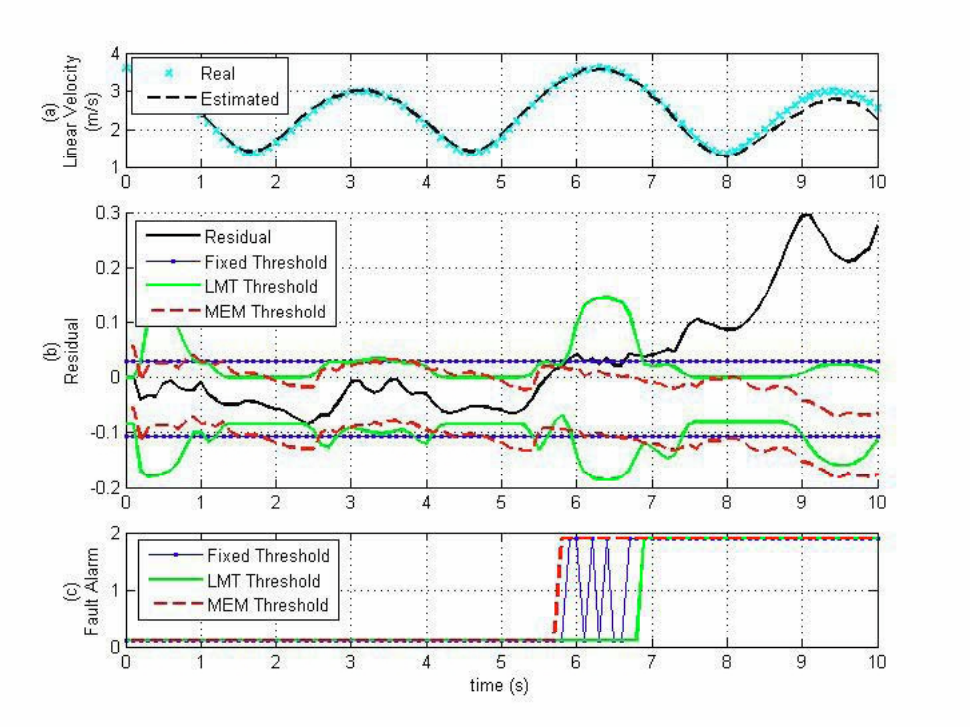

Figure 4.29: The LLM fault detection performance for system fault 6: A decrease of r_l with the rate of $0.5\,cm/s$ applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

data.

To demonstrate the advantages of the MEM method by using the RBF neural network as the models, its performance will be compared with the fixed threshold band. Lower and upper fixed threshold bands are developed to determine the minimum and maximum values that each residual signal can reach under a fault free operating condition by using Equation (4.9) as given below:

$$\begin{aligned} T'_U &= Upper\ fixed\ Threshold\ band = +s(e) + m(e) \\ T'_D &= Lower\ fixed\ Threshold\ band = -s(e) + m(e) \end{aligned} \tag{4.9}$$

where $s(e)$ is the standard deviation of the estimation error and $m(e)$ is the average of the

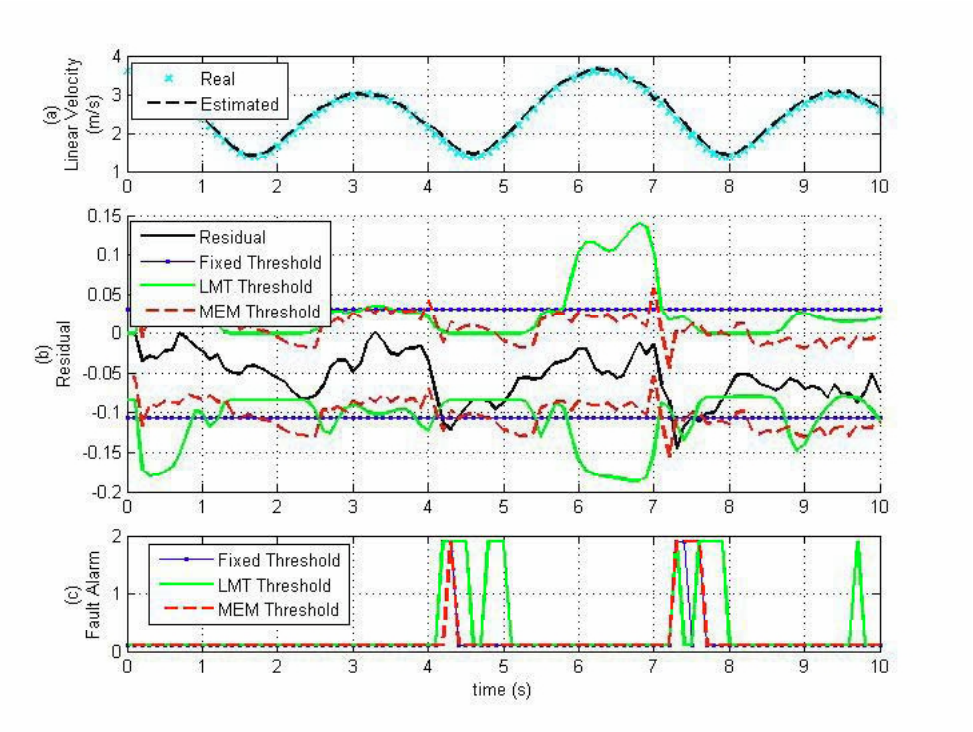

Figure 4.30: The LLM fault detection performance for system fault 7: A $\lambda = 3$ applied in the $3-4^{th}$ and the $6-7^{th}$ seconds, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

test data estimation error. Test data is chosen the same as the test data for the adaptive threshold band.

As described before, testing and validation data are chosen as the cases 2 and 5 of Table 3.2. Figure 4.32 (a) and (b) show that the developed RBF model can follow the testing data with small estimation error. Figure 4.32 (c) shows that by using the MEM adaptive threshold band there is no false alarm in healthy test data while a few false alarms are detected by the fixed threshold band. Figure 4.33 shows few false alarms with both threshold bands corresponding to the healthy validation data.

In the remained of this section, faulty scenarios and the fault detection performances are provided. Figure 4.34 shows the actuator fault 1 which is a 20% loss of control effectiveness applied after the 5^{th} second. This fault is detected without a delay by using both

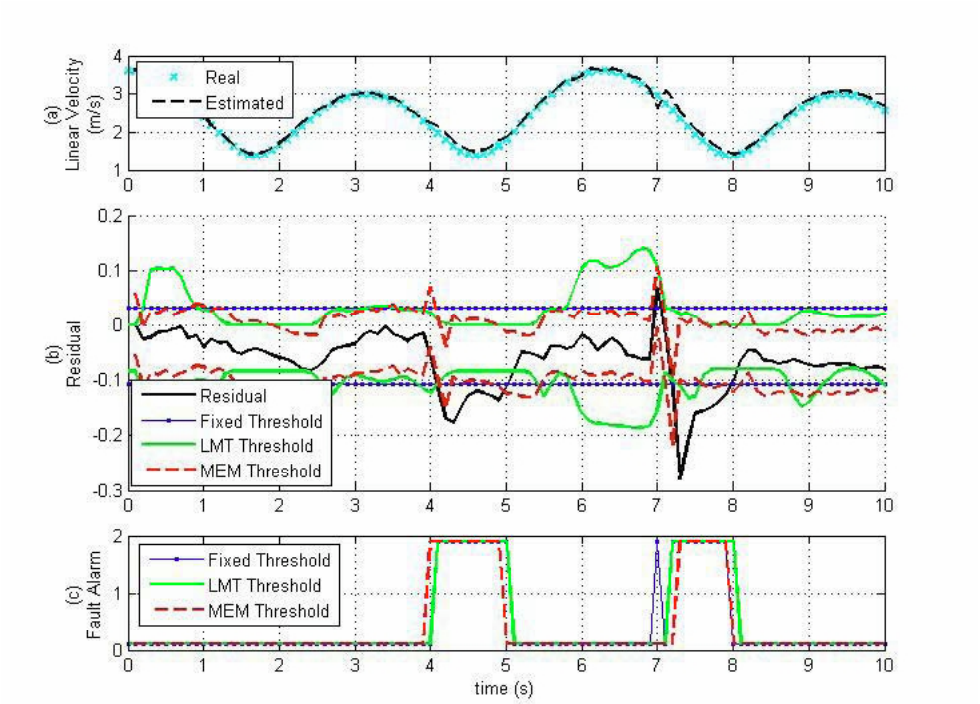

Figure 4.31: The LLM fault detection performance for system fault 8: A $\lambda = 6$ applied in the $3-4^{th}$ and the $6-7^{th}$ seconds, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

the threshold generation methods. This is due to the occurrence of a high severity fault in the system and the accuracy of the RBF model.

Figure 4.35 shows an actuator fault 2 which is a 5% loss of control effectiveness after the 5^{th} second. The fault is detected with no delay by both methods but this permanent fault is detected as an intermittent fault by the fixed threshold band which reduces its reliability.

Figure 4.36 shows an actuator fault 3 which is a $0.1\,\Omega$ reduction of the right motor resistance and is applied after the 5^{th} second. The fault is detected with a 0.1 second delay by the adaptive threshold band while the fixed threshold band has detected this permanent fault with a 0.8 second delay as an intermittent fault which reduces its reliability.

Figure 4.37 shows an actuator fault 4 which is a $0.1\,\Omega$ increase of the right motor

Case	Fixed Threshold	LMT Threshold	MEM Threshold
1 ($AL = 20\% Abruptly$)	$0S$	$0S$	$0S$
2 ($AL = 5\% Abruptly$)	0.1^*S	$0S$	0.1^*S
3 ($R_r : 0.1\Omega \downarrow \% Abruptly$)	0.4^*S	$0.3S$	$0.3S$
4 ($R_r : 0.1\Omega \uparrow \% Abruptly$)	$0.1S$	$0S$	0.3^*S
5 ($r_l : 0.2 cm/s \downarrow Incipiently$)	$3.5S$	$2.3S$	1^*S
6 ($r_l : 0.5 cm/s \downarrow Incipiently$)	0.7^*S	$1.8S$	$0.7S$
7 ($\lambda = 3\% Intermittently$)	$0.1, 0.3^*S$	$0.1S^*, 0.3^*S$	$0.2, 0.3^*S$
8 ($\lambda = 6\% Intermittently$)	$0.1S, 0.3S$	$0.1S, 0.3S$	$0S, 0.4S$

Table 4.4: Fault detection delays by using the LLM with the fixed, the LMT and the MEM threshold bands. "*" denotes that a permanent fault is detected as an intermittent fault.

resistance applied after the 5^{th} second. The fault is detected with no delay by both methods but this permanent fault is detected as an intermittent fault by fixed threshold band which reduces its reliability.

Figure 4.38 shows a system fault 5 which is a decrease of the left robot's wheel with the rate of 0.2 cm/s applied after the 5^{th} second. The fault is detected with a 0.7 second delay by the adaptive threshold band while the fixed threshold band has detected this permanent fault with a 2.9 second delay as an intermittent fault.

Figure 4.39 shows a system fault 6 which is a decrease of the left robot's wheel with the rate of 0.5 cm/s applied after the 5^{th} second. The fault is detected with a 0.4 second and a 0.9 second delay by the adaptive and the fixed threshold bands, respectively.

Figure 4.40 shows system fault 7 which is a wheel slip intermittently with $\lambda = 3$ in the $4-5^{th}$ and $7-8^{th}$ seconds. The faults are detected with a 0.1 and a 0 second delay with the adaptive threshold band and no delay with the fixed threshold band.

Figure 4.41 shows system fault 8 which is a wheel slip intermittently with $\lambda = 6$ in the $4-5^{th}$ and $7-8^{th}$ seconds. All the faults are detected with no delay with both methods.

The delay in detection in all the fault scenarios are summarized in Table 4.6. Also,

Case	Threshold	a	b	c	d	AC	TP	FP	TN	FN	P
All	Fixed	565	4	86	254	91	76	1	99	25	98
	LMT	560	9	55	285	93	84	1	99	16	97
	MEM	561	8	57	283	93	83	1	98	17	97
0	Fixed	98	3	0	0	97	-	3	97	-	0
	LMT	93	8	0	0	92	-	8	92	-	0
	MEM	94	7	0	0	93	-	7	93	-	0
1	Fixed	51	0	0	50	100	100	0	100	0	100
	LMT	51	0	0	50	100	100	0	100	0	100
	MEM	51	0	0	50	100	100	0	100	0	100
2	Fixed	51	0	4	46	96	92	0	100	8	100
	LMT	51	0	1	49	99	98	0	100	2	100
	MEM	51	0	3	47	97	94	0	100	6	100
3	Fixed	51	0	16	34	84	68	0	100	32	100
	LMT	51	0	3	47	97	94	0	100	6	100
	MEM	51	0	3	47	97	94	0	100	6	100
4	Fixed	51	0	1	49	99	98	0	100	2	100
	LMT	51	0	0	50	100	100	0	100	0	100
	MEM	51	0	4	46	96	92	0	100	8	100
5	Fixed	51	0	36	14	64	28	0	100	72	100
	LMT	51	0	24	26	76.2	52	0	100	48	100
	MEM	51	0	21	29	79.2	58	0	100	42	100
6	Fixed	51	0	12	38	88	76	0	100	24	100
	LMT	51	0	18	32	82.1	64	0	100	36	100
	MEM	51	0	7	43	93	86	0	100	14	100
7	Fixed	81	0	15	5	85	25	0	100	75	100
	LMT	80	1	8	12	91	60	1	99	40	92
	MEM	81	0	15	5	85	25	0	100	75	100
8	Fixed	80	1	2	18	97	90	1.2	98	10	95
	LMT	81	0	1	19	99	95	0	100	5	100
	MEM	80	1	4	16	95	80	1.2	99	20	94

Table 4.5: The confusion matrix corresponding to the LLM fixed, LMT and MEM threshold bands.

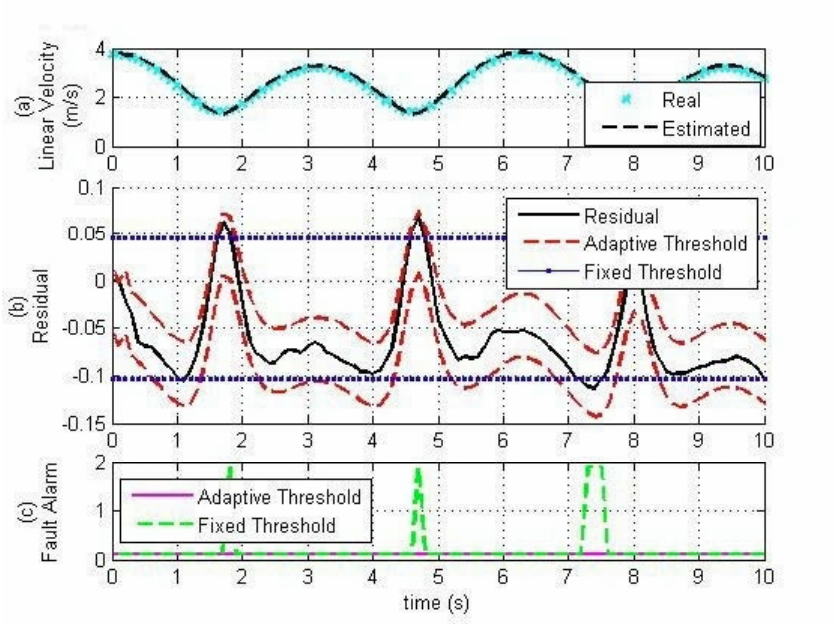

Figure 4.32: The RBF fault detection performance for the testing data, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

the confusion matrix corresponding to both threshold methods is provided in Table 4.7.

Simulation results and tables summarizing these results show that the MEM adaptive threshold band performances in faulty cases are much better than the fixed one. In case 1, due to the accuracy of the RBF model and occurrence of a high severity fault in the system, the fault is detected without delay with all methods. Under other faulty cases that contain low severity faults in the system, the advantages of using the adaptive threshold bands become evident. Corresponding to a total of 340 faulty data, 313 data are correctly detected by the MEM adaptive threshold band, while the fixed threshold band detected 224 data. This is a significant benefit of the adaptive threshold bands, although calculating the threshold bands in this case needs to perform the second neural network which imposes computational complexity to the process.

Case	Fixed Threshold	Adaptive Threshold
1 ($AL = 20\% Abruptly$)	$0S$	$0S$
2 ($AL = 5\% Abruptly$)	0^*S	$0S$
3 ($R_r : 0.1\Omega \downarrow \% Abruptly$)	0.8^*S	$0.1S$
4 ($R_r : 0.1\Omega \uparrow \% Abruptly$)	0^*S	$0S$
5 ($r_l : 0.2 cm/s \downarrow Incipiently$)	2.8^*S	$0.6S$
6 ($r_l : 0.5 cm/s \downarrow Incipiently$)	$0.9S$	$0.4S$
7 ($\lambda = 3\% Intermittently$)	$0S, 0S$	$0.1^*S, 0S$
8 ($\lambda = 6\% Intermittently$)	$0^*S, 0S$	$0S, 0S$

Table 4.6: Fault detection delays by using the RBF neural network with the fixed and the MEM threshold bands. "*" denotes that a permanent fault is detected as an intermittent Fault.

Case	Threshold	a	b	c	d	AC	TP	FP	TN	FN	P
All	Fixed	555	14	106	234	87	69	2	98	31	94
	Adaptive	554	15	27	313	95	92	3	97	8	96
0	Fixed	91	10	0	0	90	-	10	90	-	0
	Adaptive	97	4	0	0	96	-	4	96	-	0
1	Fixed	51	0	0	50	100	100	0	100	0	100
	Adaptive	50	1	0	50	99	100	2	98	0	98
2	Fixed	51	0	4	46	96	92	0	100	8	100
	Adaptive	50	1	0	50	99	100	2	98	0	98
3	Fixed	51	0	37	13	63	26	0	100	74	100
	Adaptive	50	1	1	49	98	98	2	98	2	98
4	Fixed	51	0	4	46	96	92	0	100	8	100
	Adaptive	50	1	0	50	99	100	2	98	0	98
5	Fixed	51	0	35	15	65	30	0	100	70	100
	Adaptive	50	1	6	44	93	88	2	98	12	98
6	Fixed	51	0	9	41	91	82	0	100	18	100
	Adaptive	50	1	4	46	95	92	2	98	8	98
7	Fixed	79	2	10	10	88	50	2	98	50	83
	Adaptive	79	2	5	15	93	75	2	98	25	88
8	Fixed	79	2	7	13	91	65	2	98	35	87
	Adaptive	78	3	1	19	96	95	4	96	5	86

Table 4.7: The confusion matrix corresponding to the RBF fixed and the MEM threshold bands.

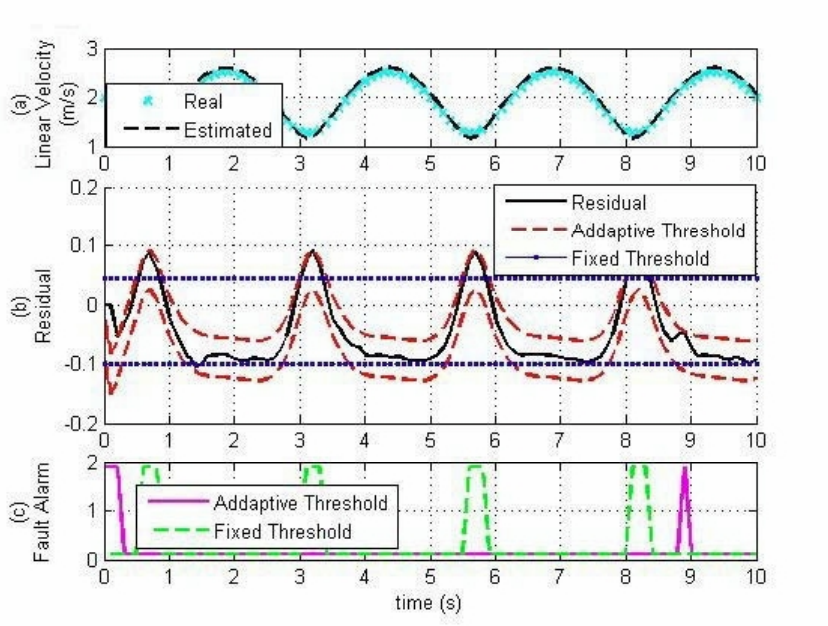

Figure 4.33: The RBF fault detection performance for the validation data, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

4.8 Conclusions

Fault detection of mobile robots by using the LLM models and RBF neural network is addressed in this chapter. The LLM fault detection performance is improved by using LMT and MEM adaptive threshold bands. Also, MEM algorithm is used to improve the RBF fault detection performance. The results are shown in simulation figures and summarized tables.

Tables 4.4 - 4.7 show both methods of adaptive threshold bands detected fault with good accuracy while RBF with MEM algorithm has a better performance in most cases. However, due to designing the second neural network in MEM algorithm, the computational complexity of this method is much more than computational complexity of the LMT method.

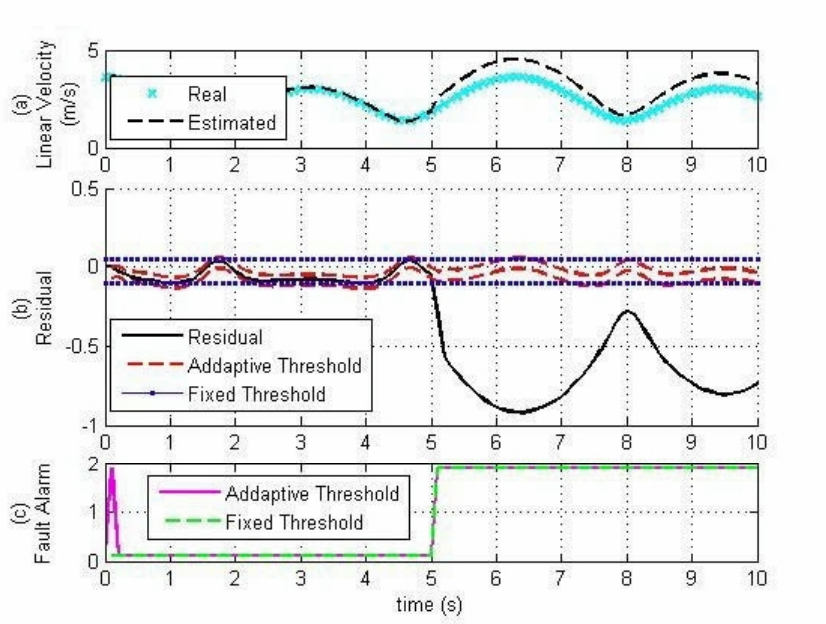

Figure 4.34: The RBF fault detection performance for actuator fault 1: A 20% loss of control effectiveness applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

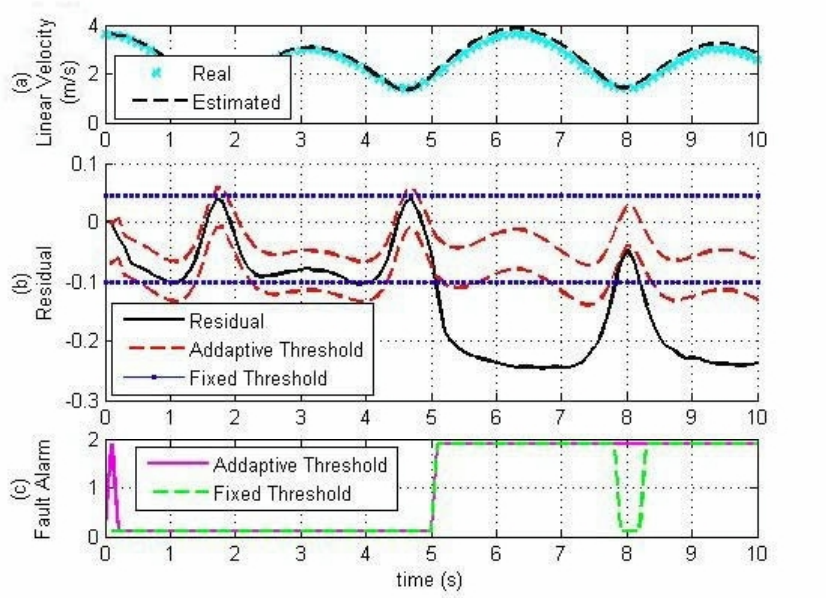

Figure 4.35: The RBF fault detection performance for actuator fault 2: A 5% loss of control effectiveness applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

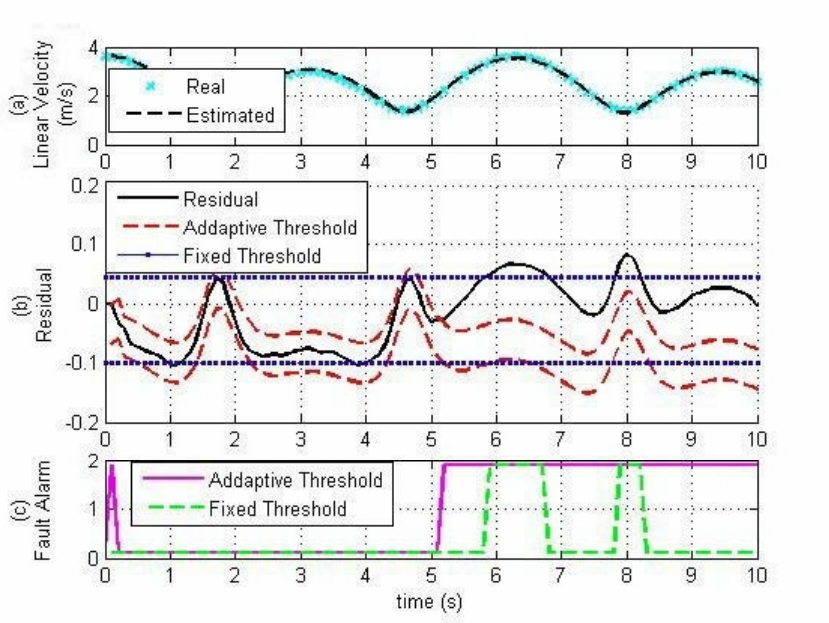

Figure 4.36: The RBF Fault detection performance for actuator fault 3: A 0.1 Ω decrease of R_r applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, (c) and Fault alarms generated.

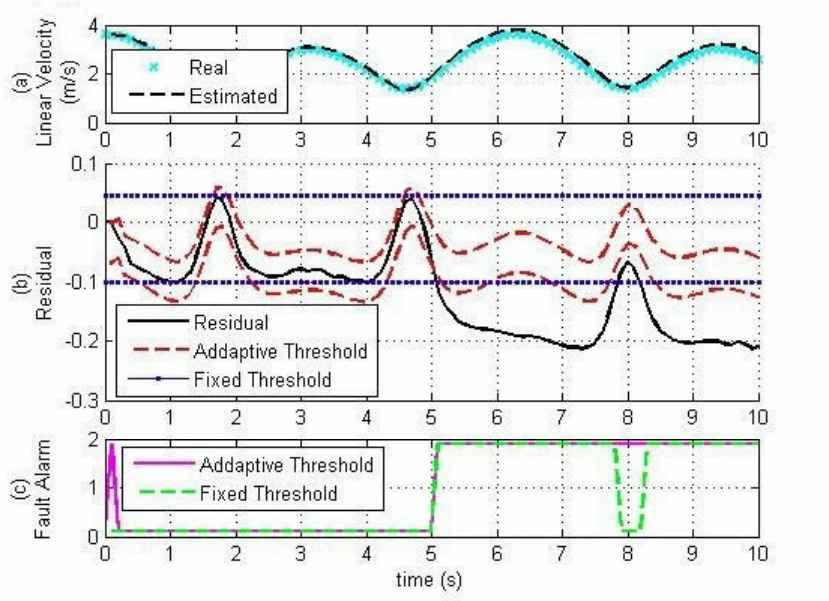

Figure 4.37: The RBF Fault detection performance for actuator fault 4: A 0.1 Ω increase of R_r applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

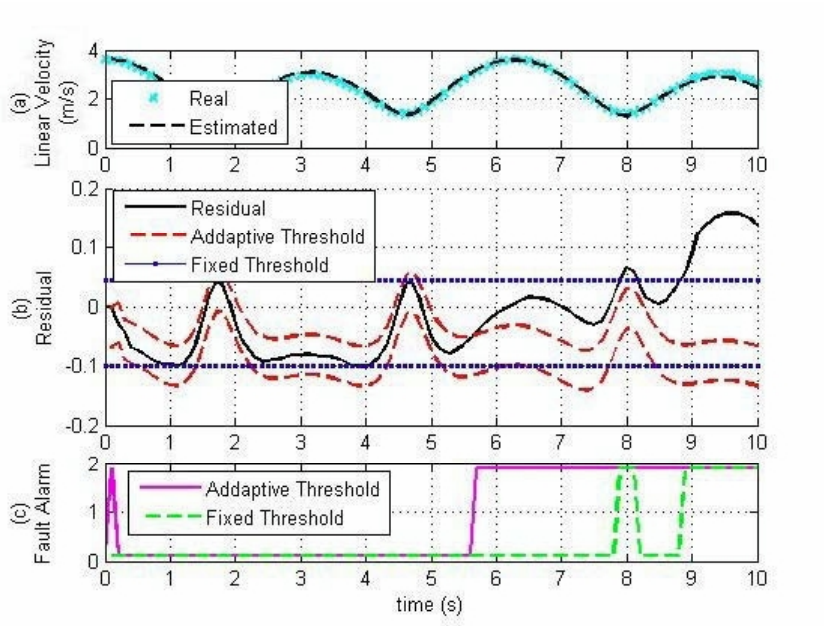

Figure 4.38: The RBF Fault detection performance for system fault 5: A decrease of r_l with the rate of $0.2\,cm/s$ applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

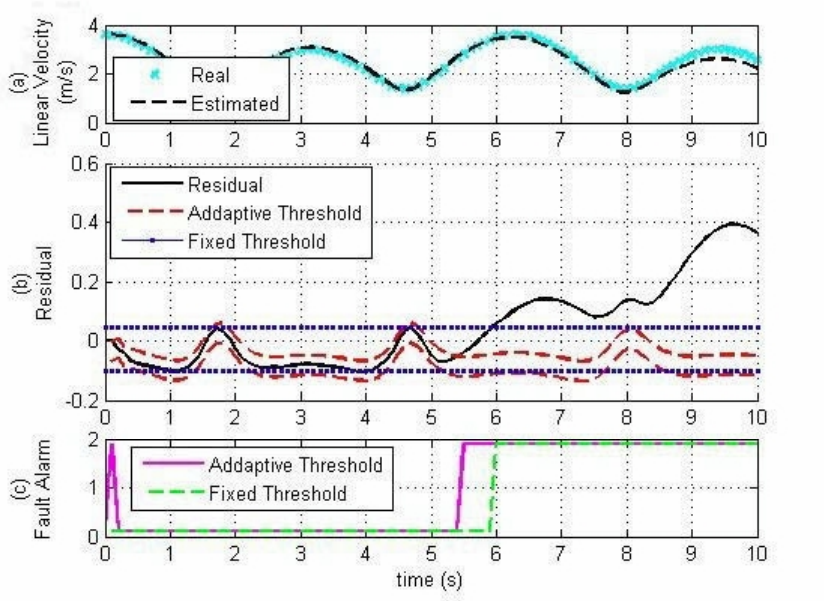

Figure 4.39: The RBF Fault detection performance for system fault 6: A decrease of r_l with the rate of $0.5\,cm/s$ applied after the 5^{th} second, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

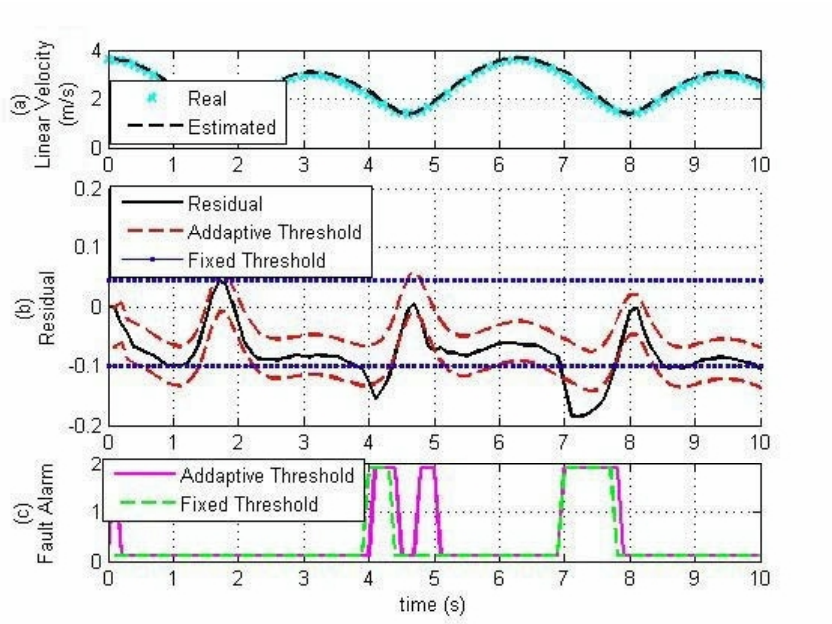

Figure 4.40: The RBF Fault detection performance for system fault 7: A $\lambda = 3$ applied in the $3-4^{th}$ and $6-7^{th}$ seconds, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

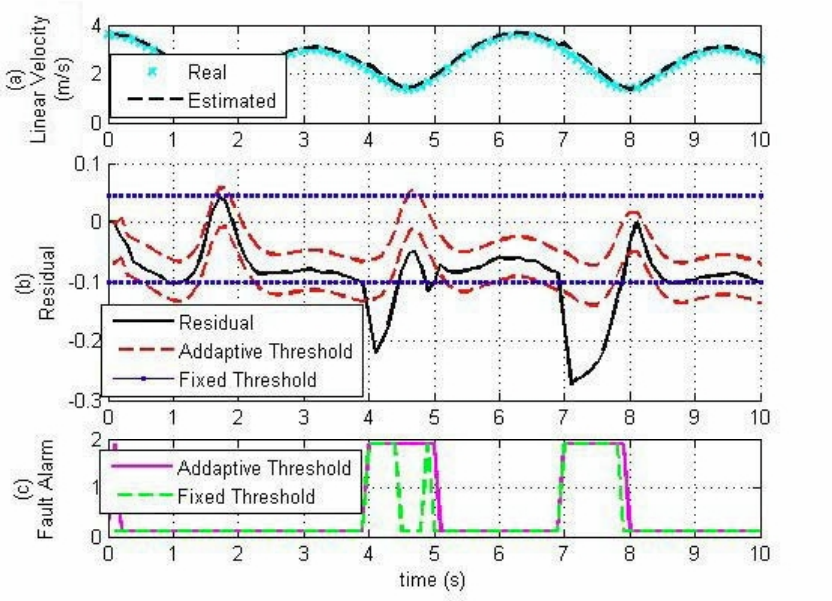

Figure 4.41: The RBF Fault detection performance for system fault 8: A $\lambda = 6$ applied in the $3-4^{th}$ and $6-7^{th}$ seconds, (a) Real and estimated linear velocity of the mobile robot, (b) Residual and the thresholds, and (c) Fault alarms generated.

Chapter 5

Fault Identification Design of Mobile Robots

5.1 Introduction

This chapter addresses fault identification of mobile robots by using a multiple model technique. For this purpose, different system faults are considered to develop a bank of models to identify faults. First, different faults having various severities are detected by using the best two different modeling techniques with adaptive threshold bands that are explained in Chapters 2 and 4 namely radial basis function (RBF) neural networks with model error modeling (MEM) adaptive threshold bands and locally linear models (LLM) with local model threshold (LMT) bands. By using the multiple model fault identification logic, faults of the system are identified. Simulation results consisting of testing the methods on 50 different cases are summarized in two confusion matrices to validate and verify the capabilities and comparisons of the proposed methods.

5.2 Fault Identification Design for Mobile Robots

In this section, the fault detection methodology is considered to be the same as in the previous chapter. The two best methods in fault detection of mobile robots are used, namely the LLM model with the LMT adaptive threshold bands and the RBF with the MEM adaptive threshold bands. In order to identify faults, for each severity range an LLM and an RBF model is trained by using the data of that severity. This implies that multiple models corresponding to the severity ranges are designed separately by each modeling method. The severity ranges are adjusted in order to design the least fault classes to cover all the studied fault ranges. The fault classes are defined corresponding to different severity ranges as described below (the fault types are explained in Section 4.3.1):

Class 0: This class contains healthy data. To detect faults in the system, the LLM model with the LMT adaptive threshold bands (Figure 4.19) and the RBF model with the MEM adaptive threshold bands (Figure 4.21) are used which are described in the previous chapter.

Class 1: This class contains loss of control effectiveness faults from 8% to 12%. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding to the 10% loss of control effectiveness fault. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 2: This class contains loss of control effectiveness faults from 13% to 17%. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding to the 15% loss of control effectiveness fault. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 3: This class contains loss of control effectiveness faults from 18% to 23%. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using

data corresponding to the 20% loss of control effectiveness fault. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 4: This class contains left wheel radius decrease faults from 0.2 cm/s to 0.5 cm/s. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding to the wheel radius decrease fault with the rate of 3 cm/s. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 5: This class contains left wheel radius decrease faults from 0.6 cm/s to 0.9 cm/s. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding to the wheel radius decrease fault with the rate of 0.7 cm/s. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 6: This class contains slippage faults from $\lambda = 2$ to $\lambda = 5$. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding the slippage fault with $\lambda = 3$. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 7: This class contains slippage faults from $\lambda = 6$ to $\lambda = 10$. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding to the slippage fault with $\lambda = 8$. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 8: This class contains right motor resistor faults from $0.53\,\Omega$ to $0.57\,\Omega$. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using

data corresponding to the resistor fault with $R_r = 55\,\Omega$. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 9: This class contains right motor resistor faults from $0.58\,\Omega$ to $0.62\,\Omega$. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding to the resistor fault with $R_r = 60\,\Omega$. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Class 10: This class contains right motor resistor faults from $0.63\,\Omega$ to $0.67\,\Omega$. To identify this class of faults, an LLM fault model with the LMT adaptive threshold bands and an RBF fault model with the MEM adaptive threshold bands are developed by using data corresponding to the resistor fault with $R_r = 65\,\Omega$. Adaptive threshold bands are constructed by using another data set corresponding to this fault.

Figure 5.1 shows the proposed methodology to identify faults. In Figure 5.1, using the current and previous inputs, output of the system and adaptive threshold bands are generated by using the methods that are explained in the previous chapter (refer to Figures 5.1 (b) and (c)). A small constant k is added to each threshold's value to increase the confidence level of the fault identification. The healthy model estimates the system's output when there is no fault in the system. Model fault 1 to model fault 10 (10 fault classes are defined, $n = 10$), estimate the output of the system when faults related to the fault class 1 to fault class of 10 occur in the system. The residuals (r_0, r_1, ..., r_{10}) are generated by comparing the estimated outputs ($\hat{y}_0$, $\hat{y}_1$, ..., $\hat{y}_{10}$) with the measured output of the system (y). For each fault, the fault decisions (f_0, f_1, ..., f_{10}) are generated by comparing the corresponding residuals with the adaptive threshold bands ($T_{U0} - T_{D0}$, $T_{U1} - T_{D1}$, ..., $T_{U10} - T_{D10}$). If the residual signal is inside the threshold band, the fault decision is set to zero, otherwise, the fault decision is set to 1. When all the fault decisions are evaluated, decision making block decides which fault (faults) has (have) occurred by using the following logic:

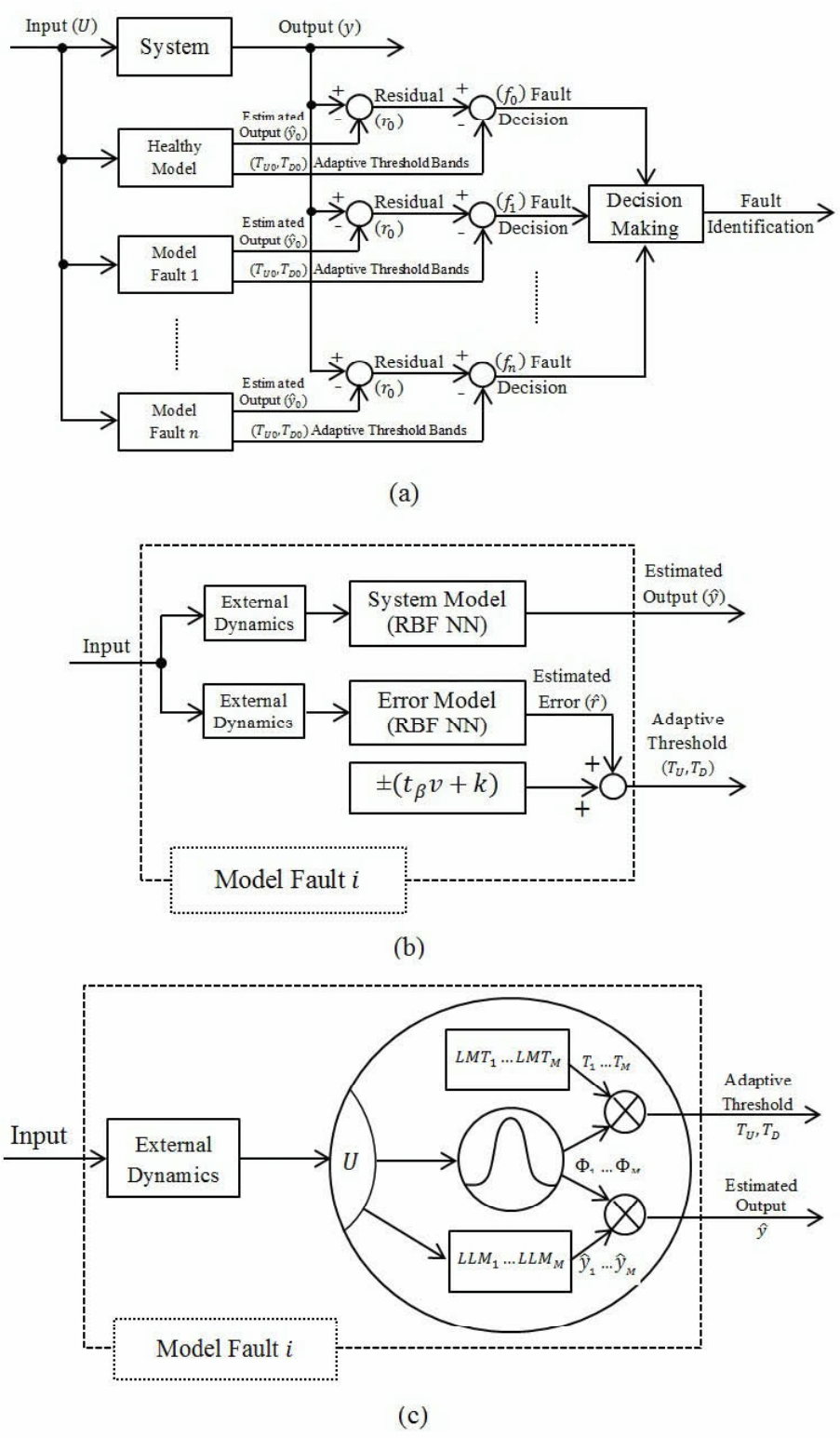

Figure 5.1: (a) Fault identification methodology, (b) Model block in the subfigure (a) if the RBF method is used, (c) Model block in the subfigure (a) if the LLM method is used.

$$\text{for i = 1 , 2, ..., n} \begin{cases} Fault\ i\ has\ occured & \text{if } f_0 = 1 \text{ and } f_i = 0; \\ No\ fault & \text{otherwise.} \end{cases}$$

The above logic is used to minimize false alarms. The reason for using the decision of the healthy model (f_0) prior to the decision of the faulty models (f_i) is due to the more reliability of the healthy model. Also, the simulations show that some models that are corresponding to the low severity faults, detect normal data as their own class of fault. As a result, by using this logic, their false alarms and neglected unless the healthy model confirms this.

5.3 Simulation Results

Using the described methodology in the previous section, faults of the system are then identified. This section provides the results for 50 different simulations on the system. These contain 3 healthy scenarios on the mobile robot when it passes through different trajectories and 47 different faulty scenarios when it passes through an eight-shaped trajectory which was described in Chapter 4. Faults consist of 15 loss of control effectiveness ranging from 8% to 22% with the resolution of 1%, 8 left wheel radius decrease faults ranging from 0.2 cm/s to 0.5 cm/s resolution of 1 cm/s, 9 slippage faults, ranging from $\lambda = 2$ to $\lambda = 10$ with the resolution of 1 λ and 15 right motor resistor faults ranging from 0.04 Ω to 0.18 Ω with the resolution of 0.01 Ω.

The performance of the fault identification on all the 50 scenarios by using the LLM models are illustrated in Figures 5.2 - 5.12. Each figure shows the corresponding model's performance on all the 50 cases by depicting the residuals and the adaptive threshold bands. The number on the top of each figure relates to the label of each case which is defined as follows:

1-3 (Class 0): No fault.

4-8 (Class 1): The control loss of effectiveness of 8, 9, 10, 11, 12% applied abruptly after the 5^{th} second.

9-13 (Class 2): The control loss of effectiveness of 13, 14, 15, 16, 17% applied abruptly after the 5^{th} second.

14-18 (Class 3): The control loss of effectiveness of 18, 19, 20, 21, 22% applied abruptly after the 5^{th} second.

19-22 (Class 4): Left wheel radius decrease faults with the rates of 0.2, 0.3, 0.4, 0.5 cm/s applied incipiently after the 5^{th} second.

23-26 (Class 5): Left wheel radius decrease faults with the rates of 0.6, 0.7, 0.8, 0.9, cm/s applied incipiently after the 5^{th} second.

27-30 (Class 6): Slippage faults of $\lambda = 2, 3, 4, 5$ applied intermittently at the $4-5^{th}$ and the $6-7^{th}$ seconds.

31-35 (Class 7): Slippage faults of $\lambda = 6, 7, 8, 9, 10$ applied intermittently at the $4-5^{th}$ and the $6-7^{th}$ seconds.

36-40 (Class 8): Right motor resistor faults of $R_r = 53, 54, 55, 56, 57\,\Omega$ applied abruptly after the 5^{th} seconds.

41-45 (Class 9): Right motor resistor faults of $R_r = 58, 59, 60, 61, 62\,\Omega$ applied abruptly after the 5^{th} second.

46-50 (Class 10): Right motor resistor faults of $R_r = 63, 64, 65, 66, 67\,\Omega$ applied abruptly after the 5^{th} second.

Figure 5.2 shows the performance of the model labeled zero, representing the healthy system, in detecting faults. It shows that by using the LLM models, healthy data are very well distinguished. Also, faults related to the first, second, third, seventh, eighth and ninth faulty classes are very well detected, and faults related to the fifth faulty class are well detected and faults related to the fourth, sixth and tenth faulty classes are barely detected due to the low severity of the faults.

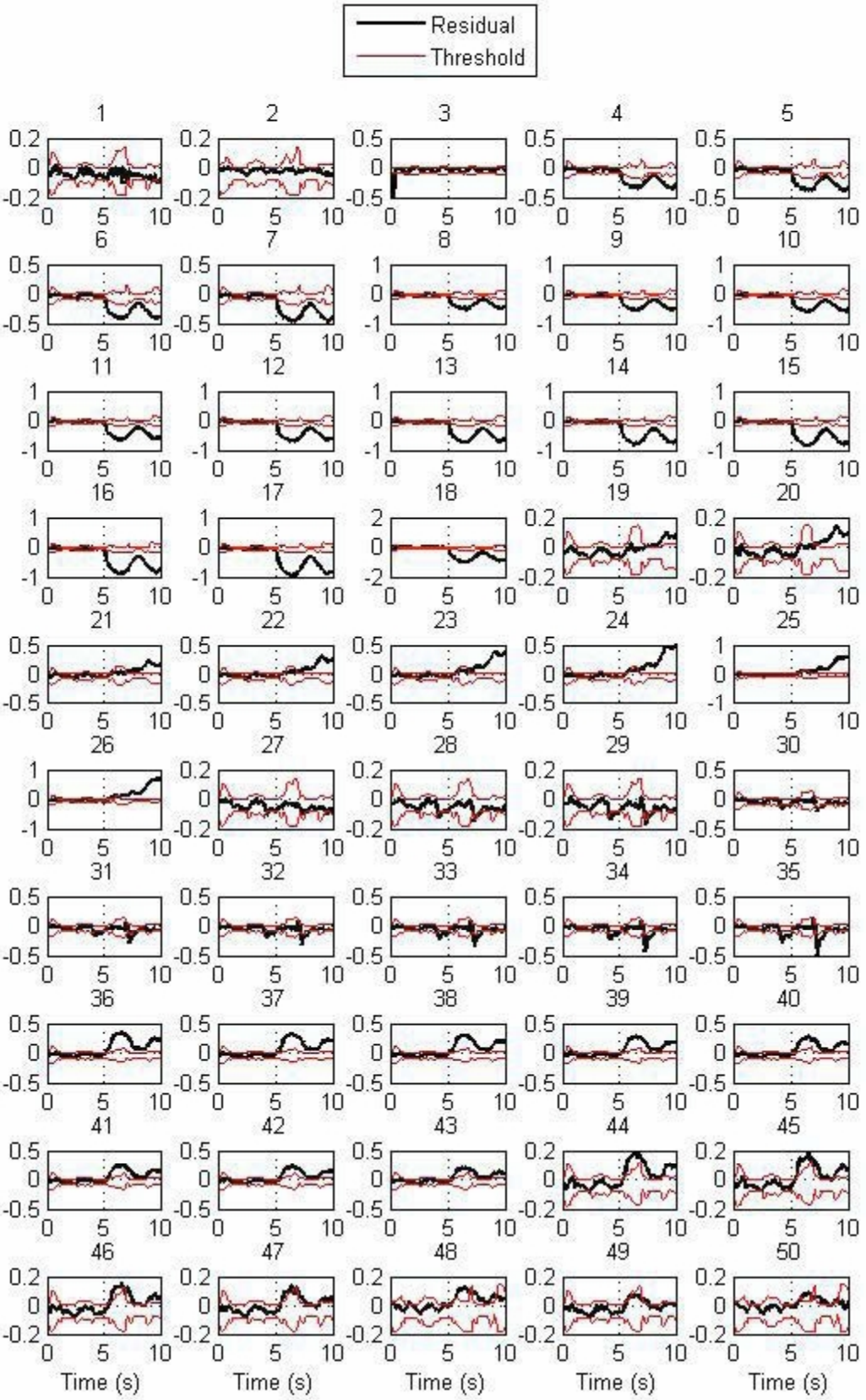

Figure 5.2: Fault detection of all the 50 cases by using the LLM models corresponding to the healthy system

Figure 5.3 shows the performance of the first LLM model, representing the fault class 1, in identifying different faults in the system. It shows that faults of this class are identified with a good performance. A few data points in the second and the third classes of faults are identified as the first class of fault. The worst performance of this model is in detecting faults of the seventh fault class in the first fault class. Figures 5.4 and 5.5 show the performance of the second and the third LLM models, representing the for fault classes of 2 and 3, in identifying different faults in the system. The good performance of these models in identifying the corresponding fault classes can be clearly concluded.

Figures 5.6 and 5.7 show the performance of the fourth and the fifth LLM models, representing fault classes of 4 and 5, in identifying different faults in the system. As far as Figure 5.6 and Figure 5.2 are concerned faults of the fourth class are poorly identified. The main reason is that the low severity faults of this class are not detected well with the healthy model. Also, the fifth class faults are barely identified. As far as these figures are concerned, certain data are misclassified using these models.

Figures 5.8 and 5.9 show the performance of the sixth and the seventh LLM models, representing fault classes of 6 and 7, in identifying different faults in the system. As far as Figures 5.8 and 5.9 and Figure 5.2 are concerned, faults of these classes are barely identified. The worst performance of the seventh model is in detecting certain first class faults as the seventh class.

Figures 5.10, 5.11 and 5.12 show the performance of the eighth, the ninth and the tenth LLM models, representing fault classes of 8, 9 and 10, in identifying different faults in the system. As far as Figures 5.10, 5.11, 5.12 and Figure 5.2 are concerned, faults of the eighth and the ninth classes are well identified and faults of the tenth class are barely identified. The main reason for the identifying the tenth class barely is that the low severity faults of this class are not detected well with the healthy model.

The fault identification results using the LLM models are summarized in Tables 5.1 and 5.2 which show the confusion matrix using the number and the percentage of the

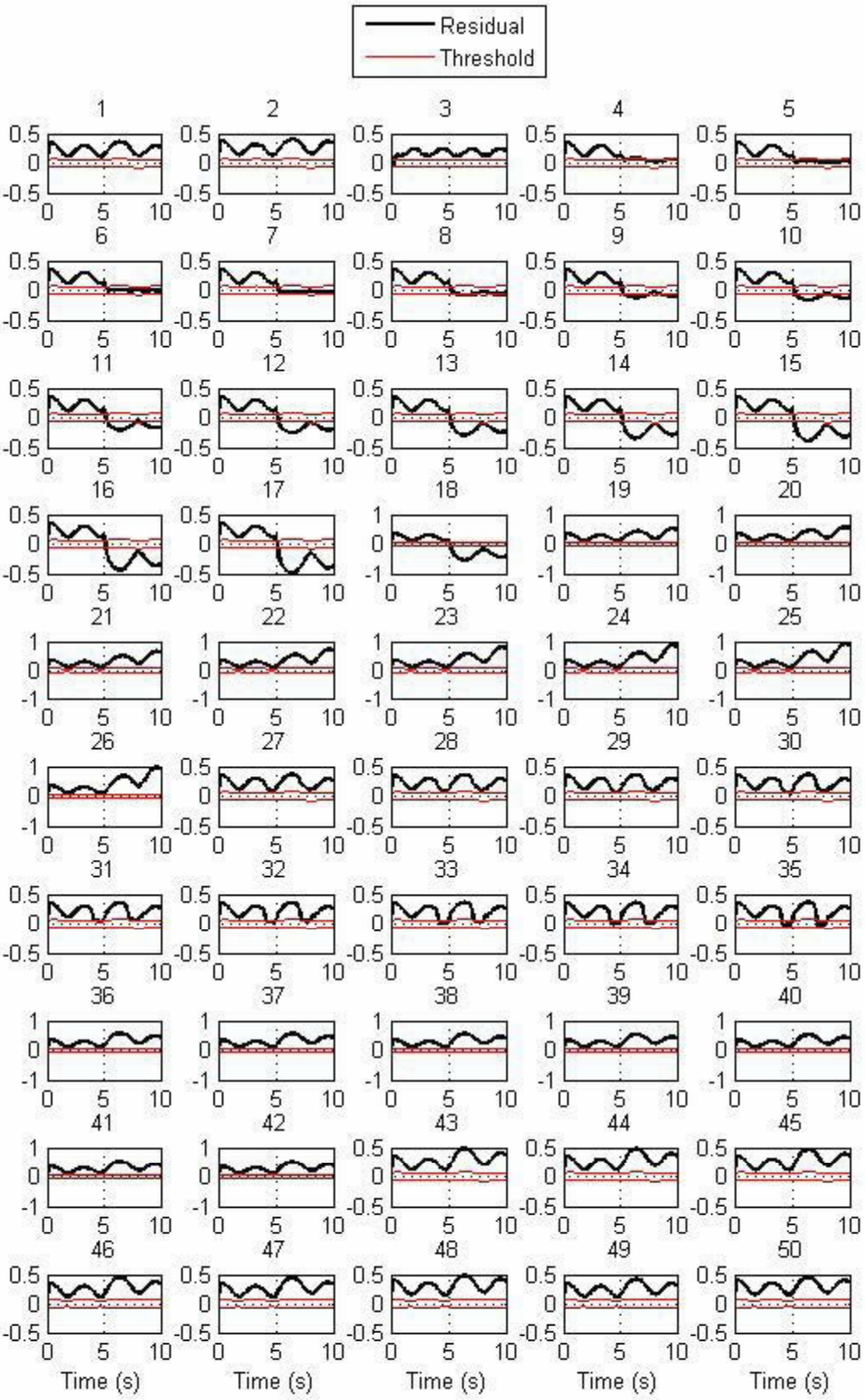

Figure 5.3: Fault identification of the mobile robot by using LLM models: First faulty model performance for all the 50 cases.

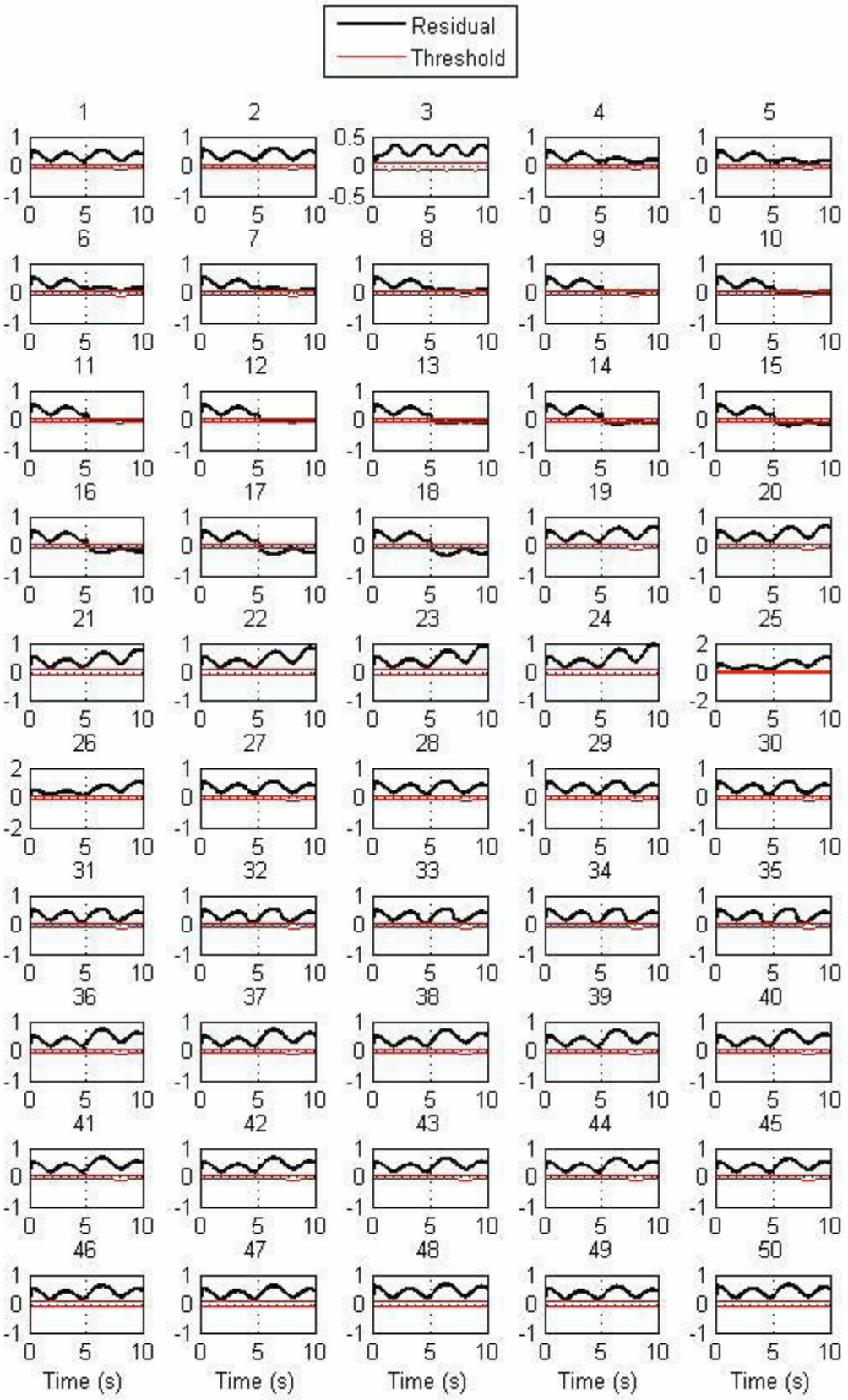

Figure 5.4: Fault identification of the mobile robot by using LLM models: Second faulty model performance for all the 50 cases.

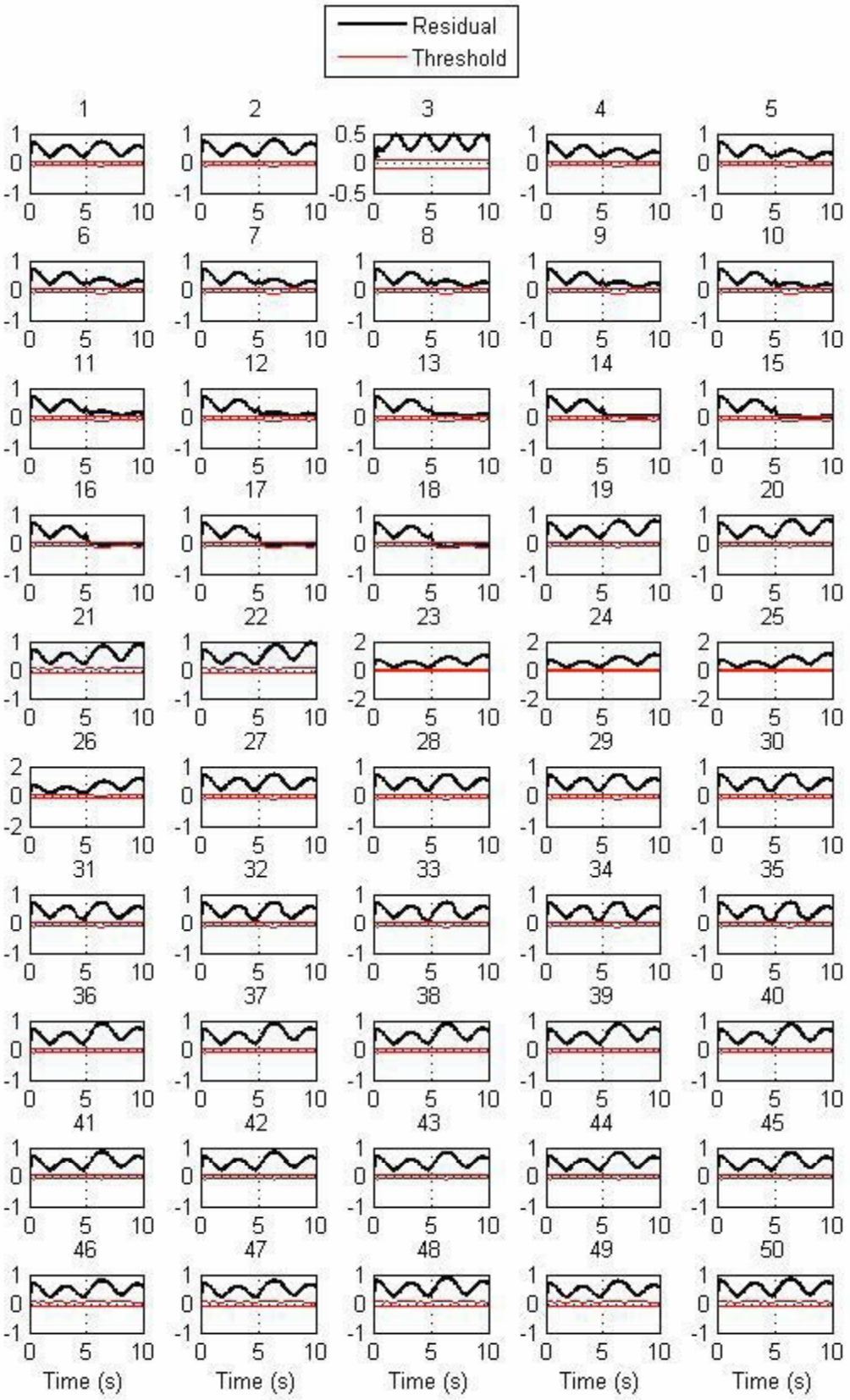

Figure 5.5: Fault identification of the mobile robot by using LLM models: Third faulty model performance for all the 50 cases.

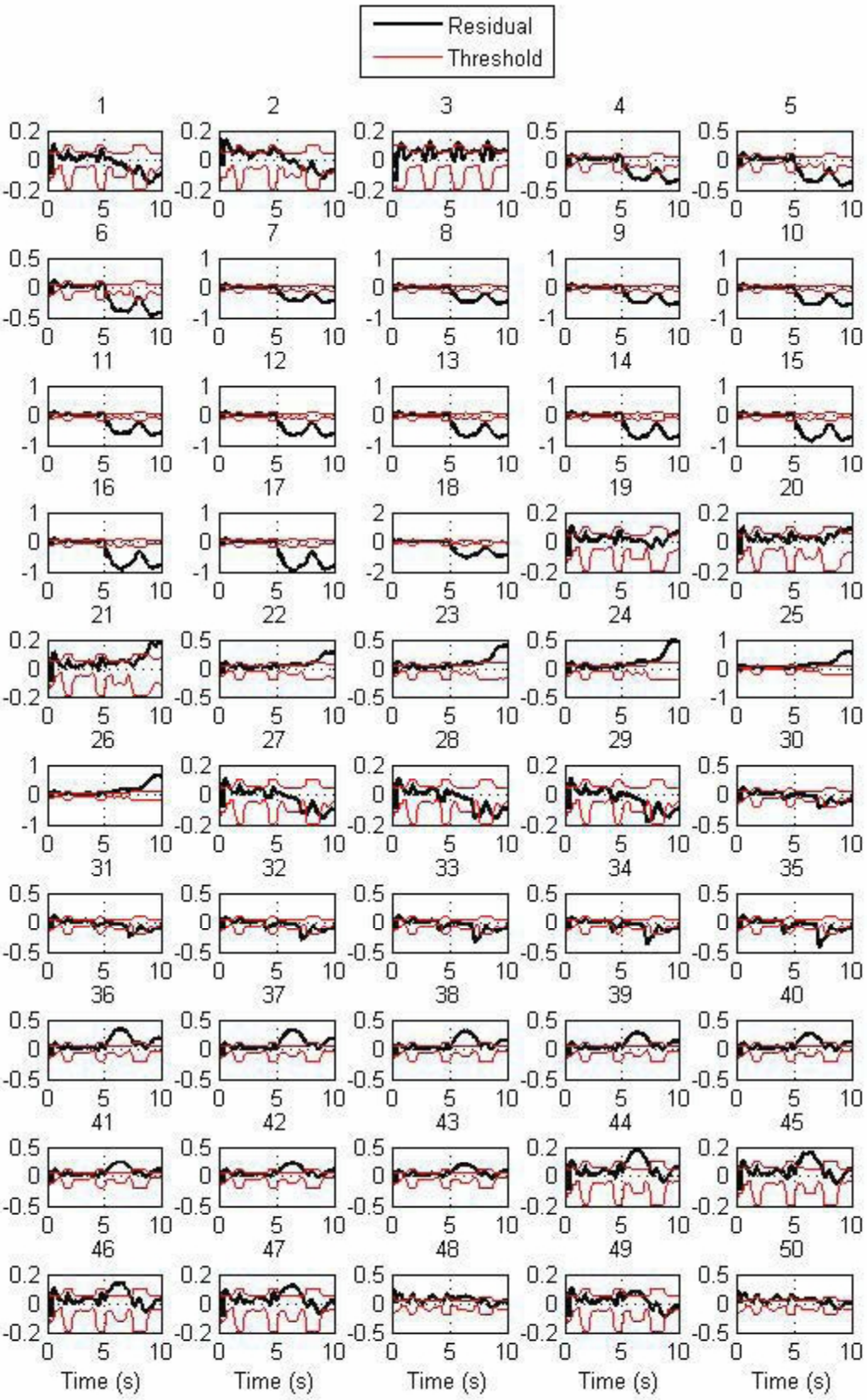

Figure 5.6: Fault identification of the mobile robot by using LLM models: Fourth faulty model performance for all the 50 cases.

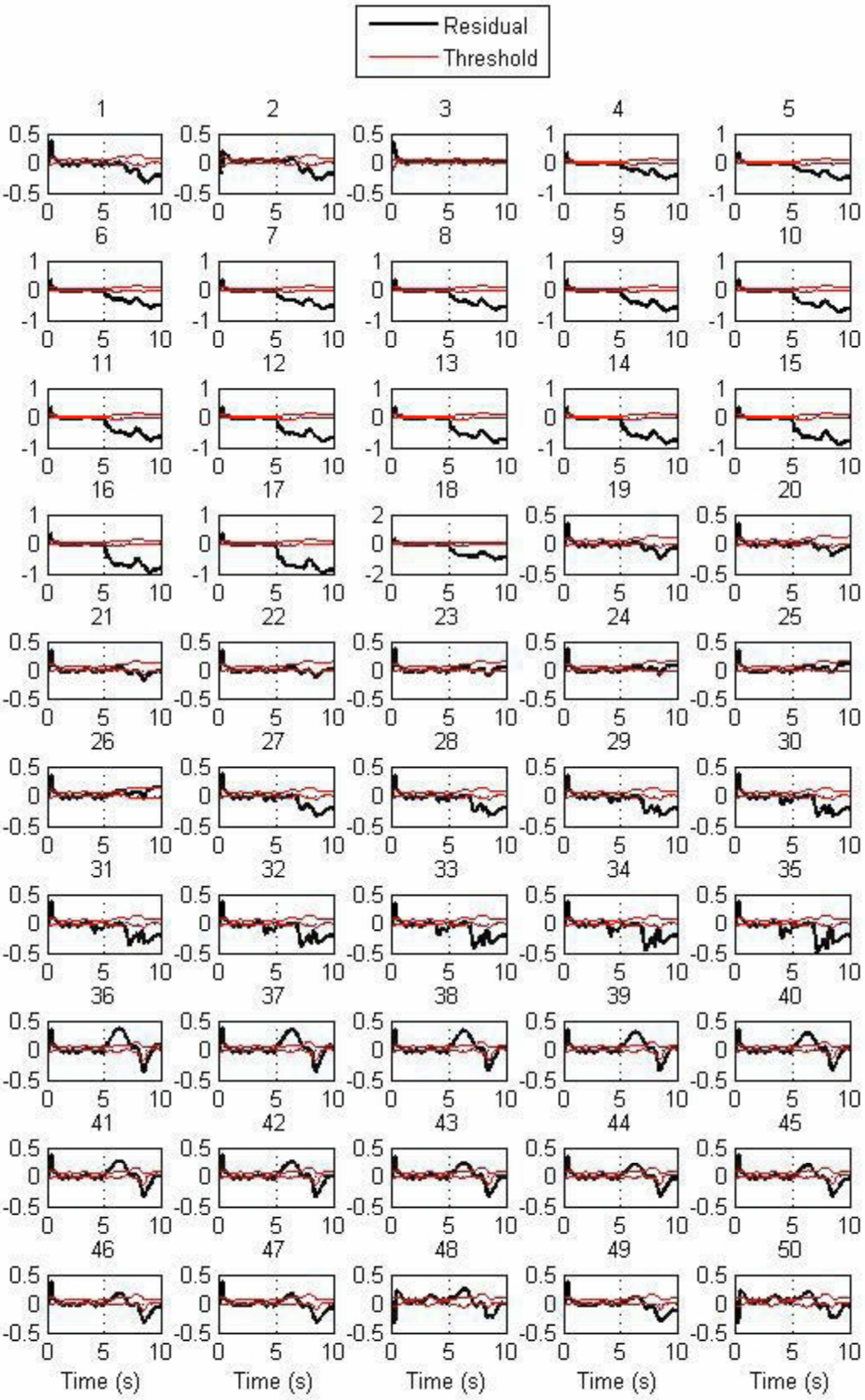

Figure 5.7: Fault identification of the mobile robot by using LLM models: Fifth faulty model performance for all the 50 cases.

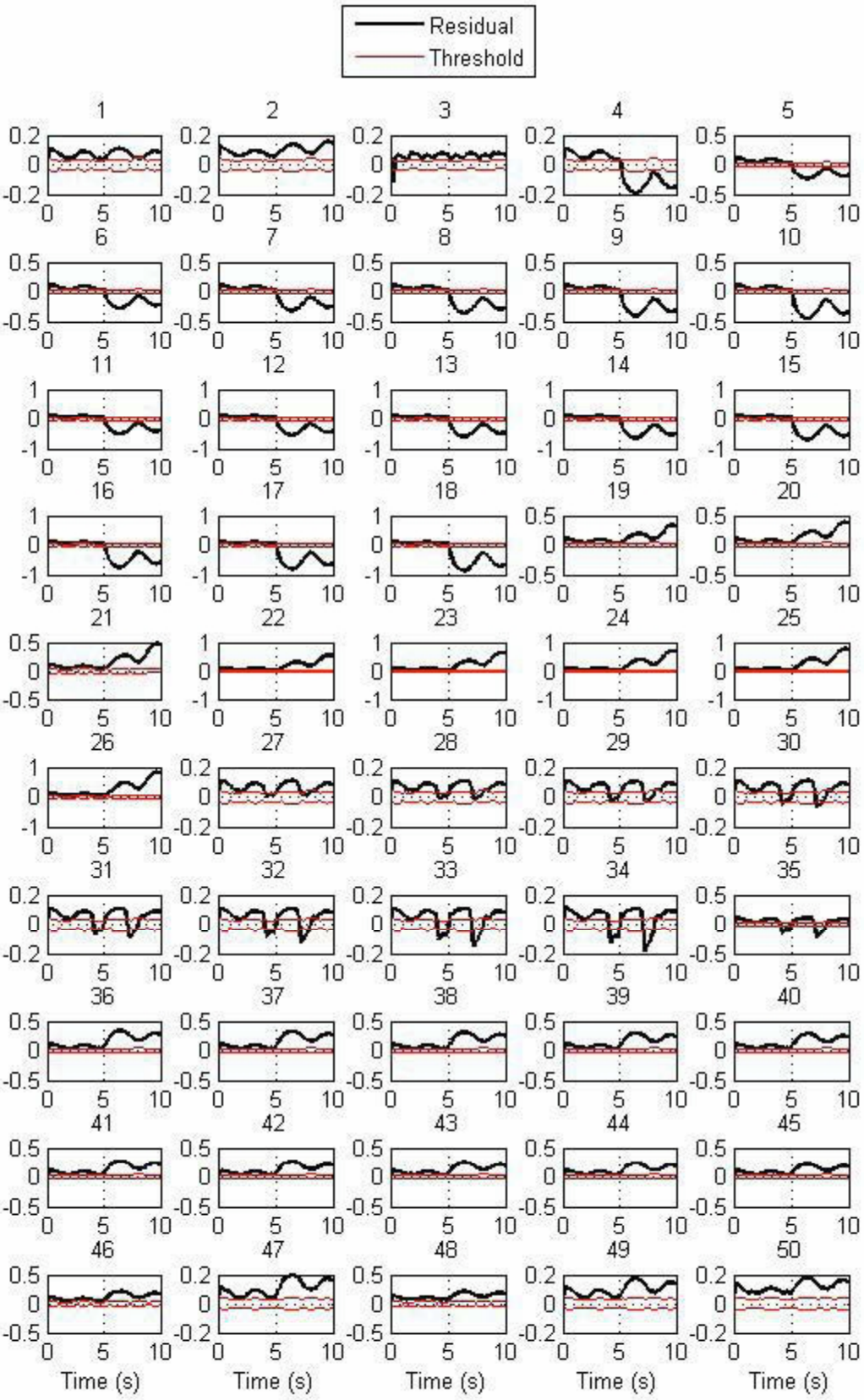

Figure 5.8: Fault identification of the mobile robot by using LLM models: Sixth faulty model performance for all the 50 cases.

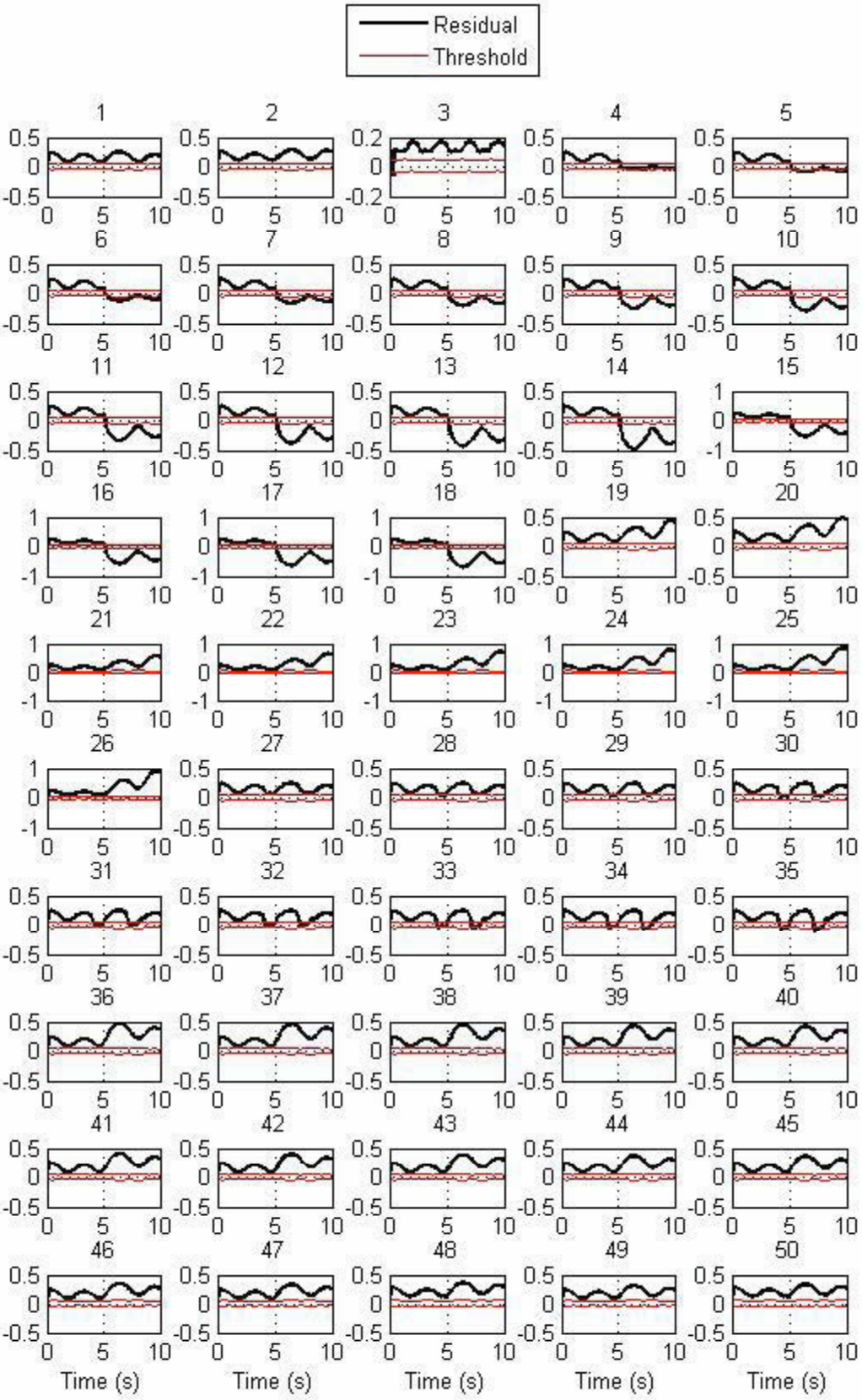

Figure 5.9: Fault identification of the mobile robot by using LLM models: Seventh faulty model performance for all the 50 cases.

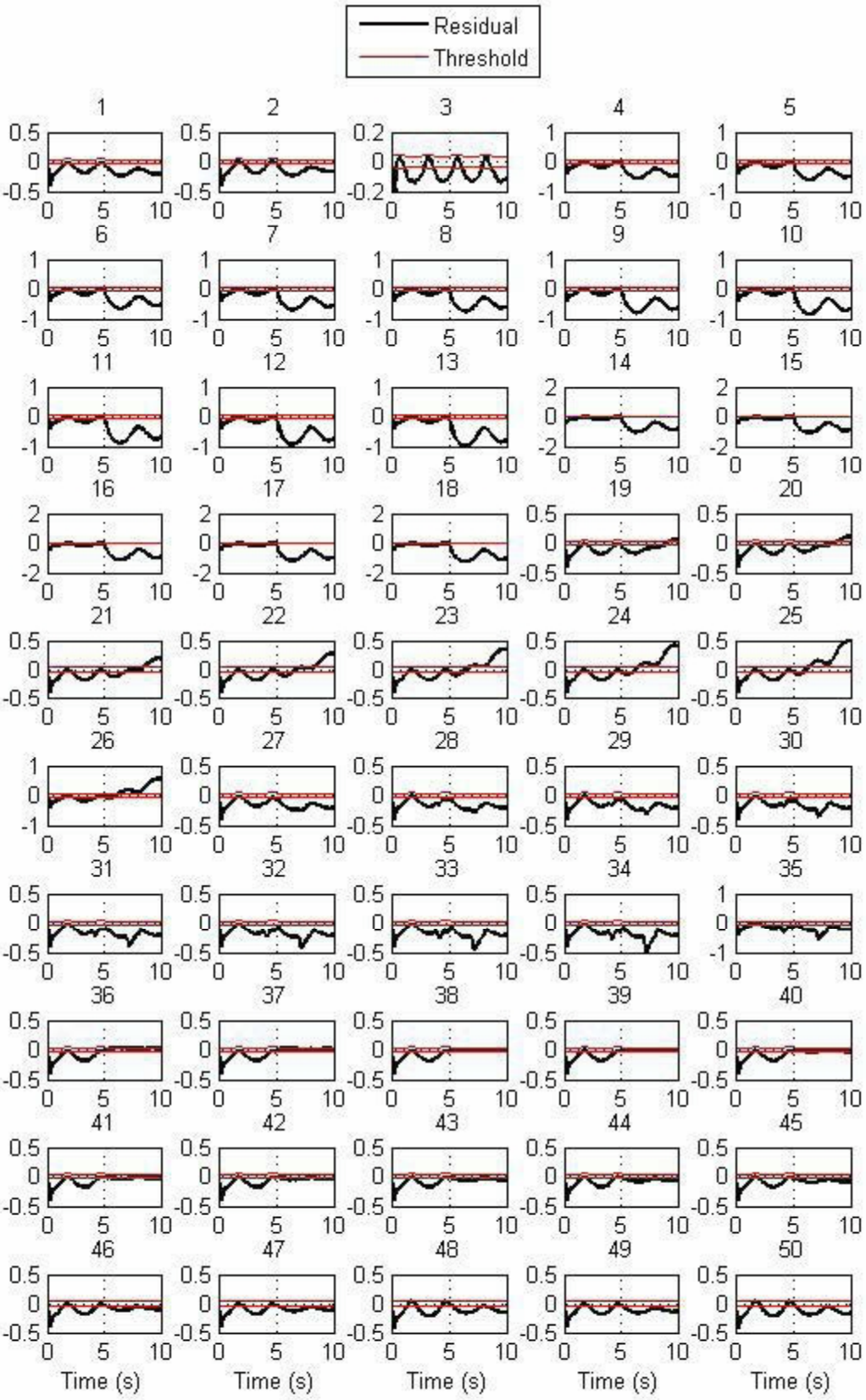

Figure 5.10: Fault identification of the mobile robot by using LLM models: Eighth faulty model performance for all 50 cases.

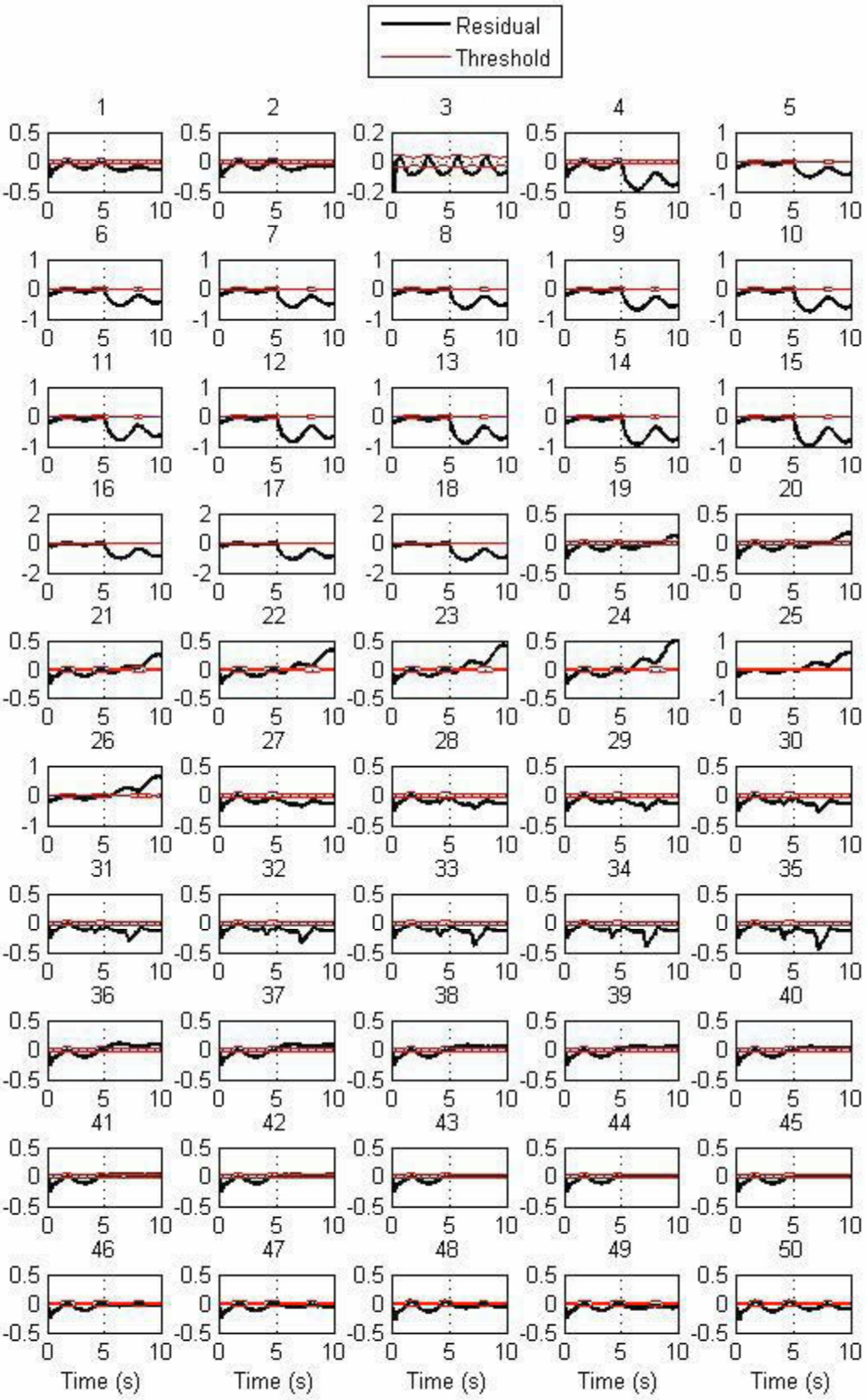

Figure 5.11: Fault identification of the mobile robot by using LLM models: Ninth faulty model performance for all the 50 cases.

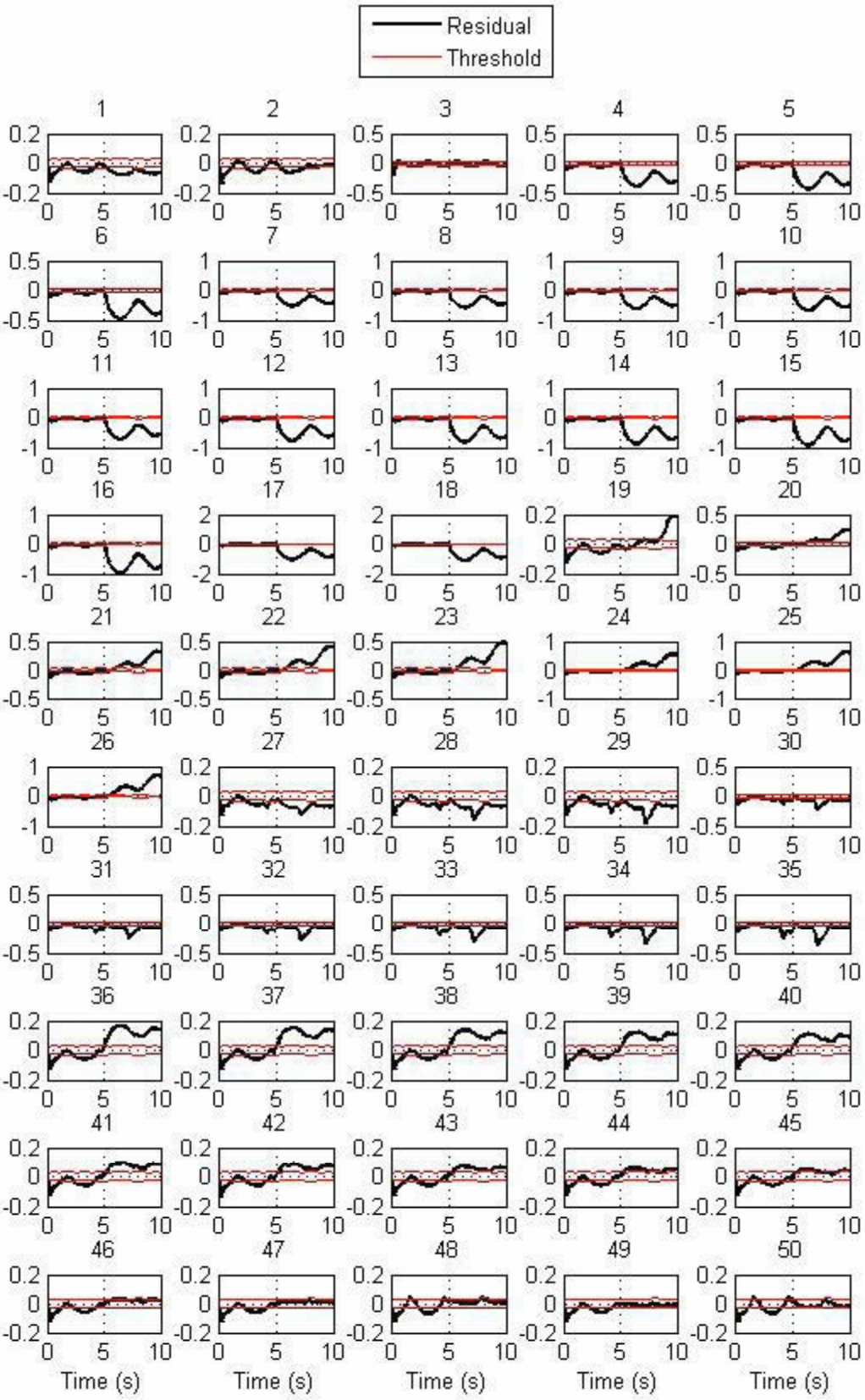

Figure 5.12: Fault identification of the mobile robot by using LLM models: Tenth faulty model performance for all the 50 cases.

Class	0	1	2	3	4	5	6	7	8	9	10	TNFD
0	2929	2	1	0	18	10	5	1	11	13	4	0
1	0	224	26	0	17	0	5	63	0	5	0	250
2	0	27	219	39	1	0	0	3	0	5	0	250
3	0	5	34	235	1	0	0	4	0	5	0	250
4	81	0	0	0	69	18	0	0	41	31	11	200
5	47	0	0	0	5	138	0	0	2	12	0	200
6	27	0	0	0	40	0	42	0	0	0	0	80
7	5	78	0	0	53	4	11	61	0	0	0	100
8	7	0	0	0	71	63	0	0	230	28	0	250
9	20	0	0	0	90	41	0	0	23	225	20	250
10	105	0	0	0	82	15	0	0	12	42	137	250

Table 5.1: The confusion matrix corresponding to the fault identification by using the LLM models and multiple model algorithm. The columns represent the actual classes and the rows represent the estimated classes. TNFD denotes the total number of faulty data in each class.

identified data in each class. The overall fault detection and identification results by using the LLM models are also illustrated in Tables 5.4 and 5.5. The notations for the symbols used in these tables are summarized in Table 5.3.

The performance of the fault identification on all the 50 faults by using the RBF neural networks with adaptive threshold generated by the MEM algorithm are illustrated in Figures 5.13 - 5.23. Each figure shows one model's performance on all the 50 cases by depicting the residuals and the adaptive thresholds.

Figure 5.13 shows the performance corresponding to the label zero, the healthy system in detecting faults. It shows that using the RBF models, data without faults are very well distinguished. Also, faults of the first, the second, the third, the fifth, the seventh, the eighth, the ninth and the tenth classes are very well detected, and faults related to other classes are well detected.

Figure 5.14 shows the performance of the first RBF model, representing the fault class 1, in identifying different faults in the system. It shows that faults of this class are

Class	0	1	2	3	4	5	6	7	8	9	10
0	98.6	0.1	0	0	0.6	0.3	0.2	0	0.4	0.4	0.1
1	0	89.6	10.4	0	6.8	0	2	25.2	0	2	0
2	0	10.8	87.6	15.6	0.4	0	0	1.2	0	2	0
3	0	2	13.6	94	0.4	0	0	1.6	0	2	0
4	40.5	0	0	0	34.5	9	0	0	20.5	15.5	5.5
5	23.5	0	0	0	2	69	0	0	1	6	0
6	33.7	0	0	0	50	0	52.5	0	0	0	0
7	2	78	0	0	53	4	11	61	0	0	0
8	2.8	0	0	0	28.4	25.2	0	0	92	11.2	0
9	8	0	0	0	36	16.4	0	0	9.2	90	8
10	42	0	0	0	32.8	6	0	0	4.8	16.8	54.8

Table 5.2: The confusion matrix shown in percentages corresponding to the fault identification by using the LLM models and multiple model algorithm. The columns represent the actual classes and the rows represent the estimated classes.

Symbol	Percentage	Performance	Interpretation
$\surd$	90-100 %	Very Good	Well Detected/Identified
+	70-90 %	Good	Detected/Identified
$\times$	50-70 %	Acceptable	Barely Detected/Identified
-	$<$50 %	Poor	Poorly Detected/Identified

Table 5.3: Legend.

Class	0	1	2	3	4	5	6	7	8	9	10
Performance	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	+	$\times$	$\surd$	$\surd$	$\surd$	$\times$

Table 5.4: The LLM models detectability.

Class	0	1	2	3	4	5	6	7	8	9	10
Performance	$\surd$	+	+	$\surd$	-	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\times$

Table 5.5: The LLM models identifiability.

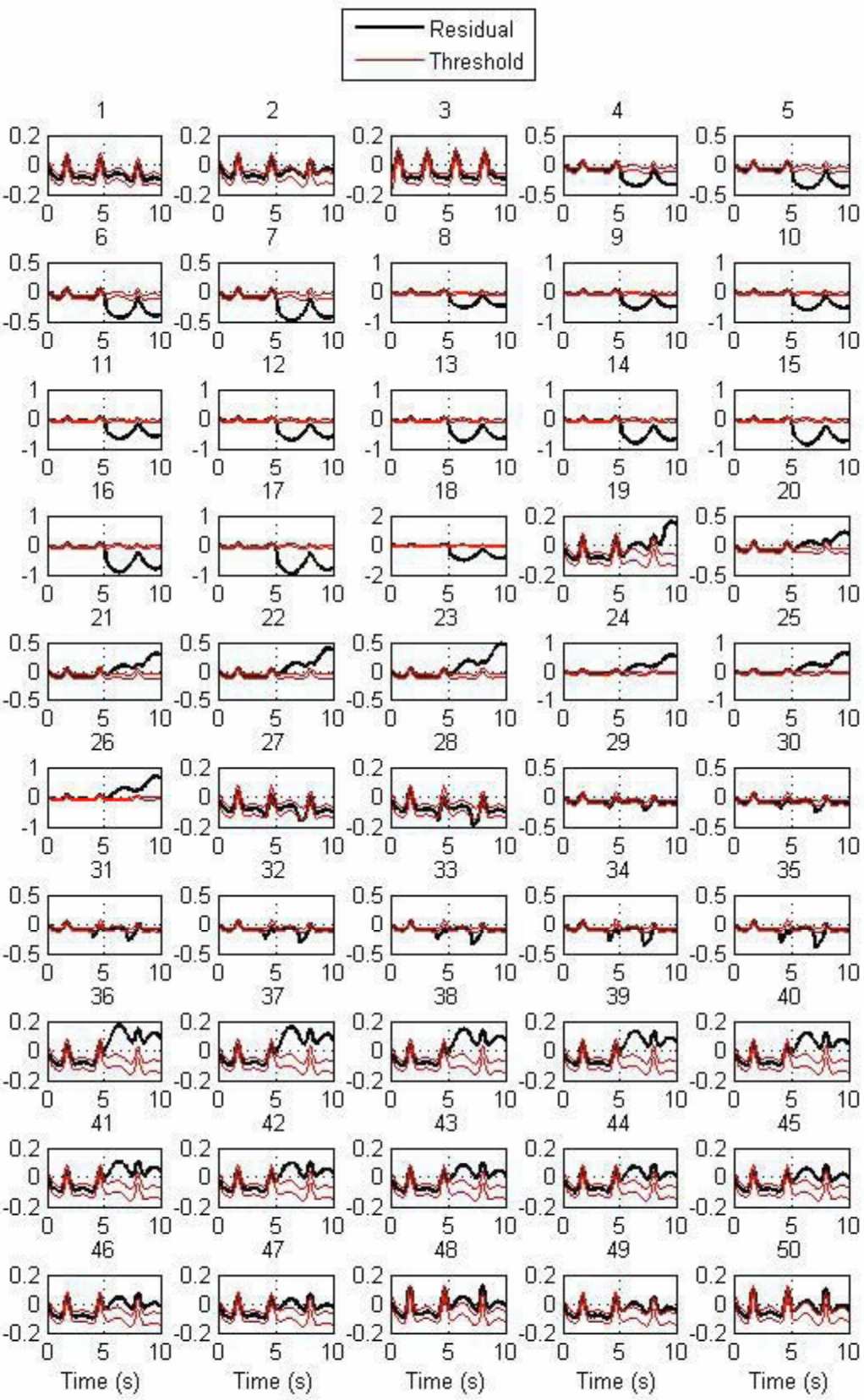

Figure 5.13: Fault detection of all the 50 cases by using the RBF neural networks corresponding to the healthy system

identified with a good performance. A few data points in the second and the third classes are classified in the first class. The worst performance of this model is in detecting faults of the seventh class in the first class. Figures 5.15 and 5.16 show the performance of the second and the third RBF models, representing the fault classes of 2 and 3, in identifying different faults in the system. The good performance of these models in identifying the corresponding fault classes can be clearly concluded.

Figures 5.17 and 5.18 show the performance of the fourth and the fifth RBF models, representing the fault classes of 4 and 5, in identifying different faults in the system. As far as Figures 5.17 and 5.18 and Figure 5.13 are concerned, faults of the fourth class are barely and faults of the fifth class are well identified. Also, certain data of other classes are misclassified by using these models.

Figures 5.19 and 5.20 show the performance of the sixth and the seventh RBF models, representing fault classes of 6 and 7, in identifying different faults in the system. As far as Figures 5.19 and 5.20 and Figure 5.13 are concerned, faults of the sixth class are well and faults of the seventh class are barely identified.

Figures 5.21, 5.22 and 5.23 show the performance of the eighth, the ninth and the tenth RBF models, representing fault classes of 8, 9 and 10, in identifying different faults in the system. As far as Figures 5.21, 5.22 and 5.23 and Figure 5.13 are concerned, faults of theses classes are very well identified. Using these models, some data of the fourth and the fifth faulty classes are misclassified.

The fault identification results by using the RBF models are summarized in Tables 5.6 and 5.7 which show the confusion matrix by using the number and the percentage of the identified data in each class. The overall fault detection and identification results by using the RBF models are also illustrated in Tables 5.8 and 5.9.

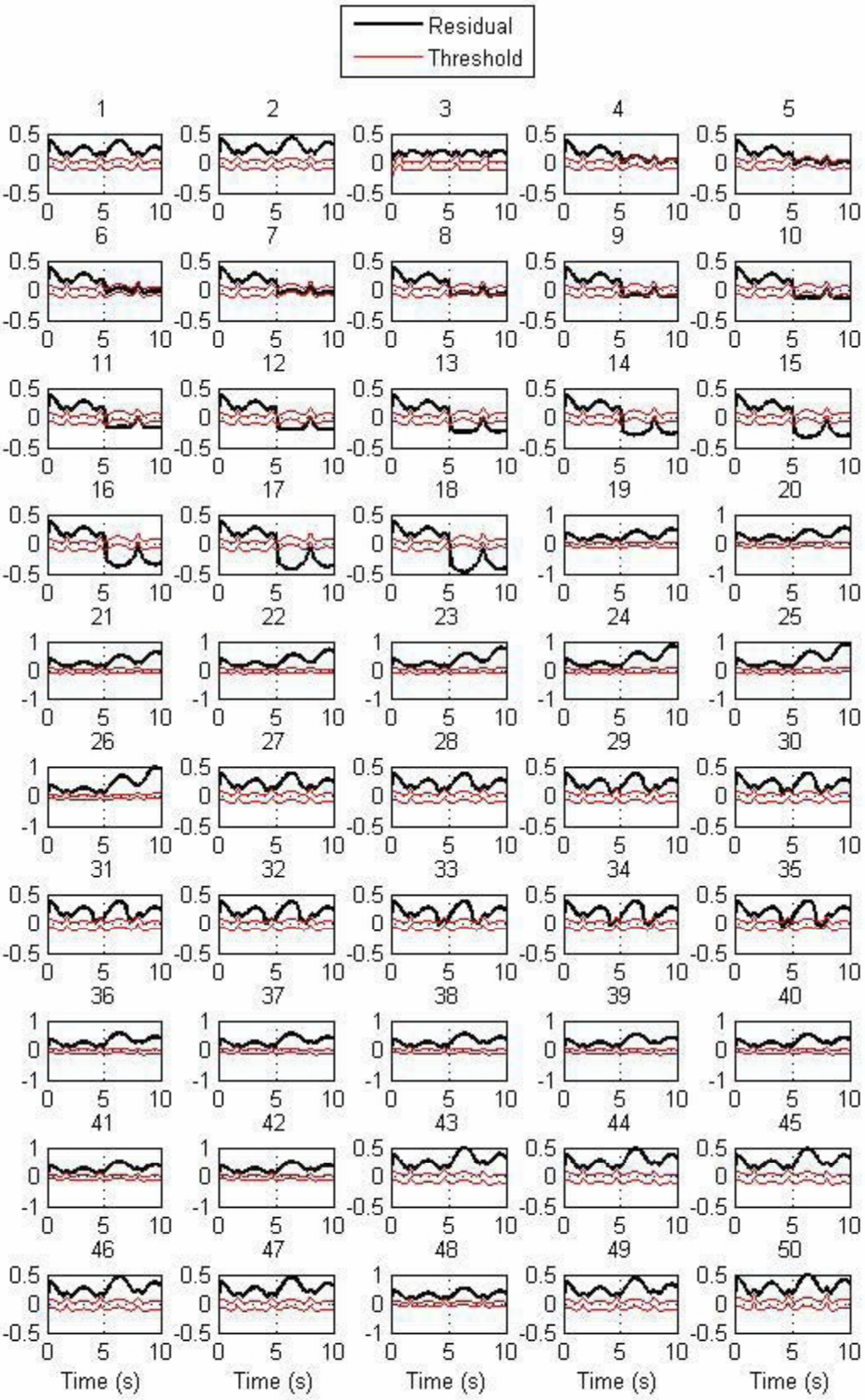

Figure 5.14: Fault identification of the mobile robot by using RBF Neural Networks: First faulty model performance for all the 50 cases.

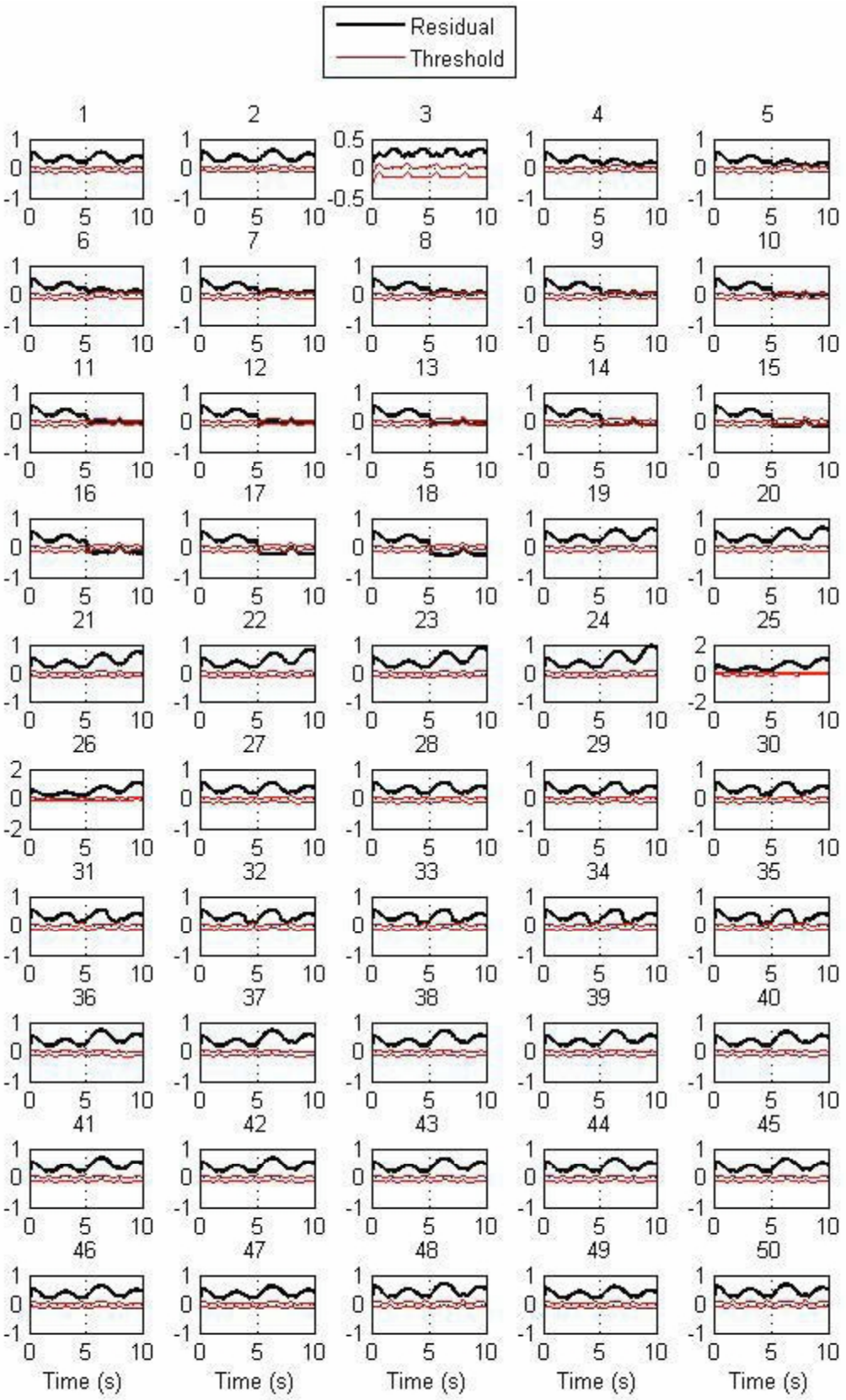

Figure 5.15: Fault identification of the mobile robot by using RBF Neural Networks: Second faulty model performance for all the 50 cases

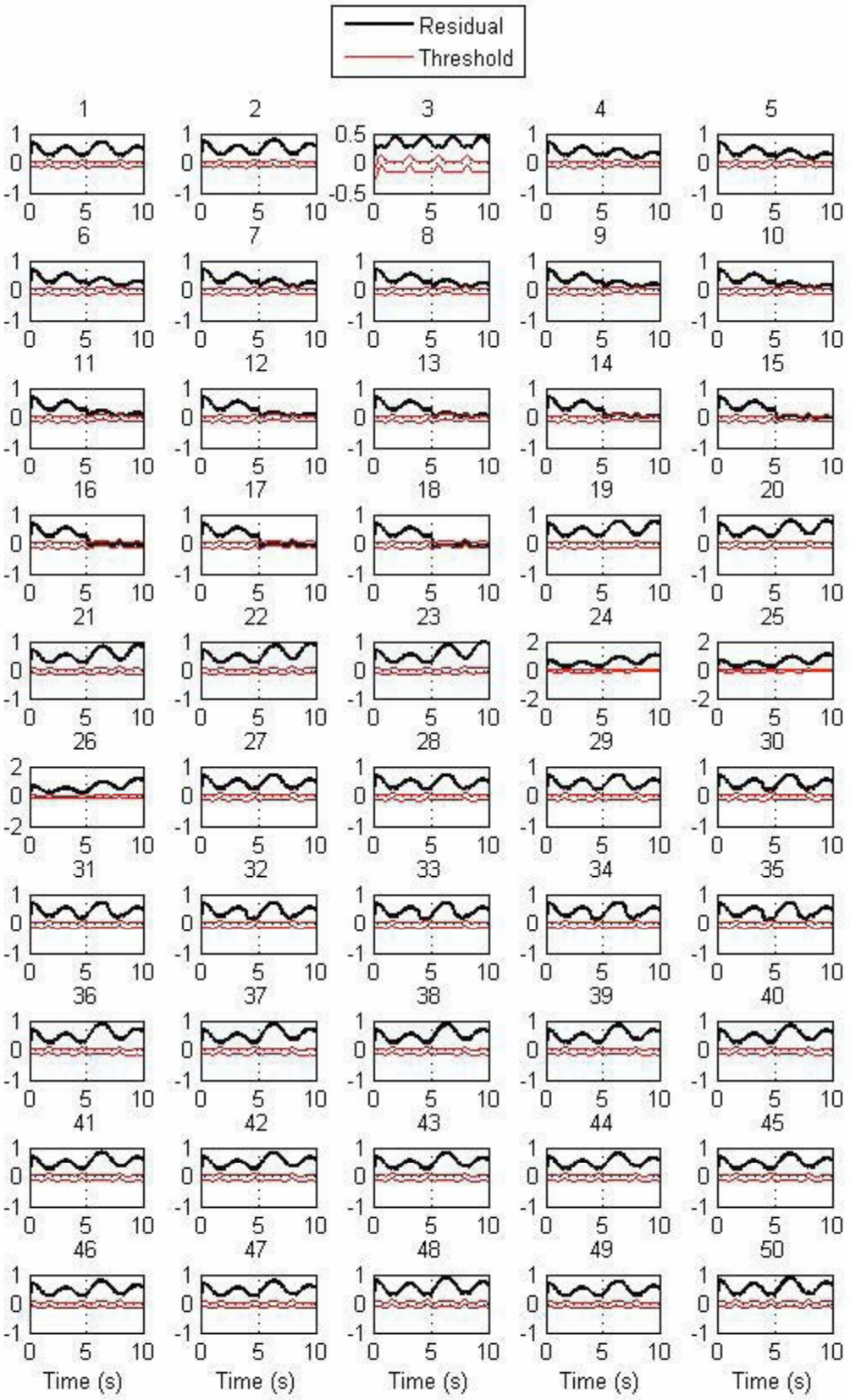

Figure 5.16: Fault identification of the mobile robot by using RBF Neural Networks: Third faulty model performance for all the 50 cases

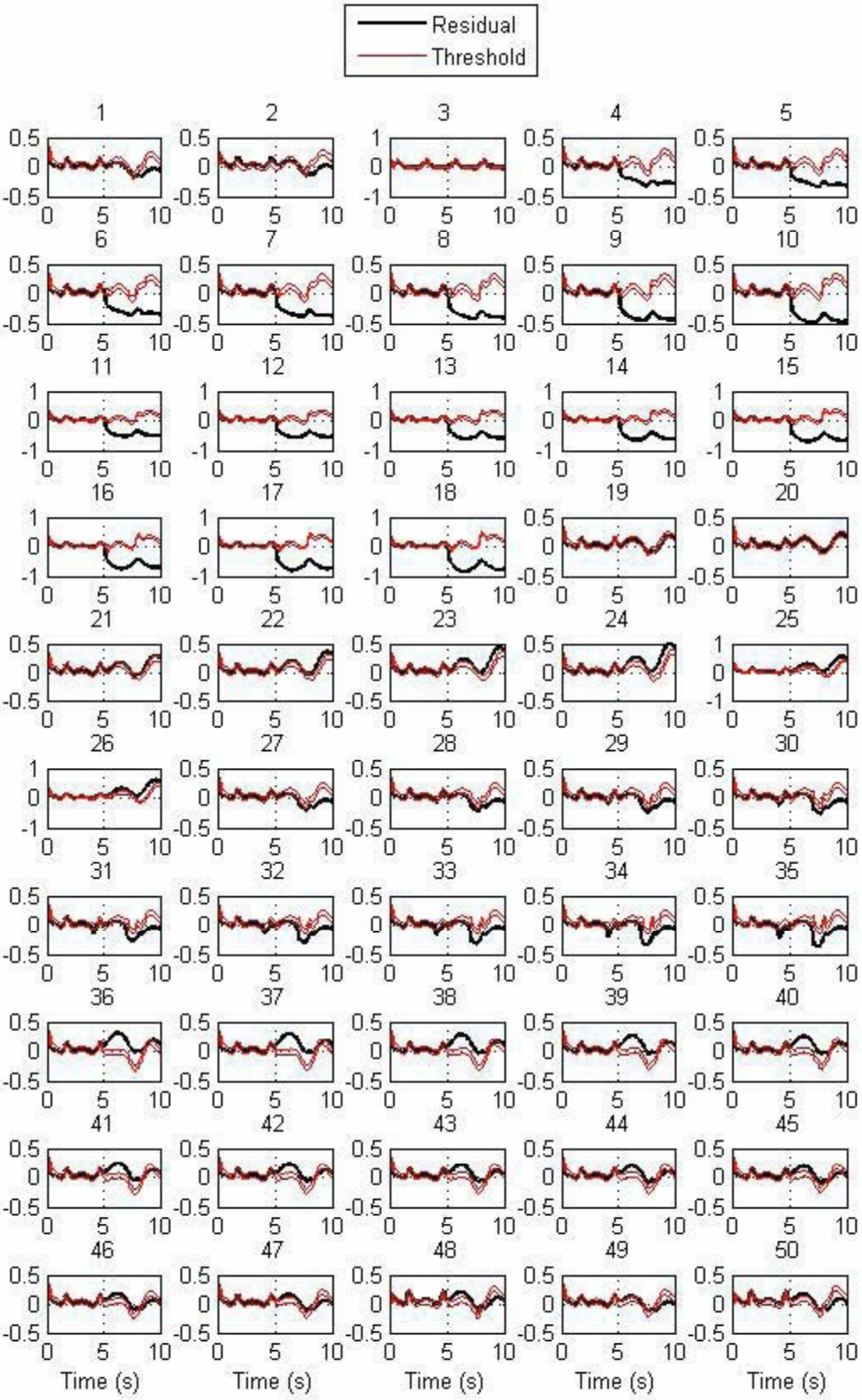

Figure 5.17: Fault identification of the mobile robot by using RBF Neural Networks: Fourth faulty model performance for all the 50 cases

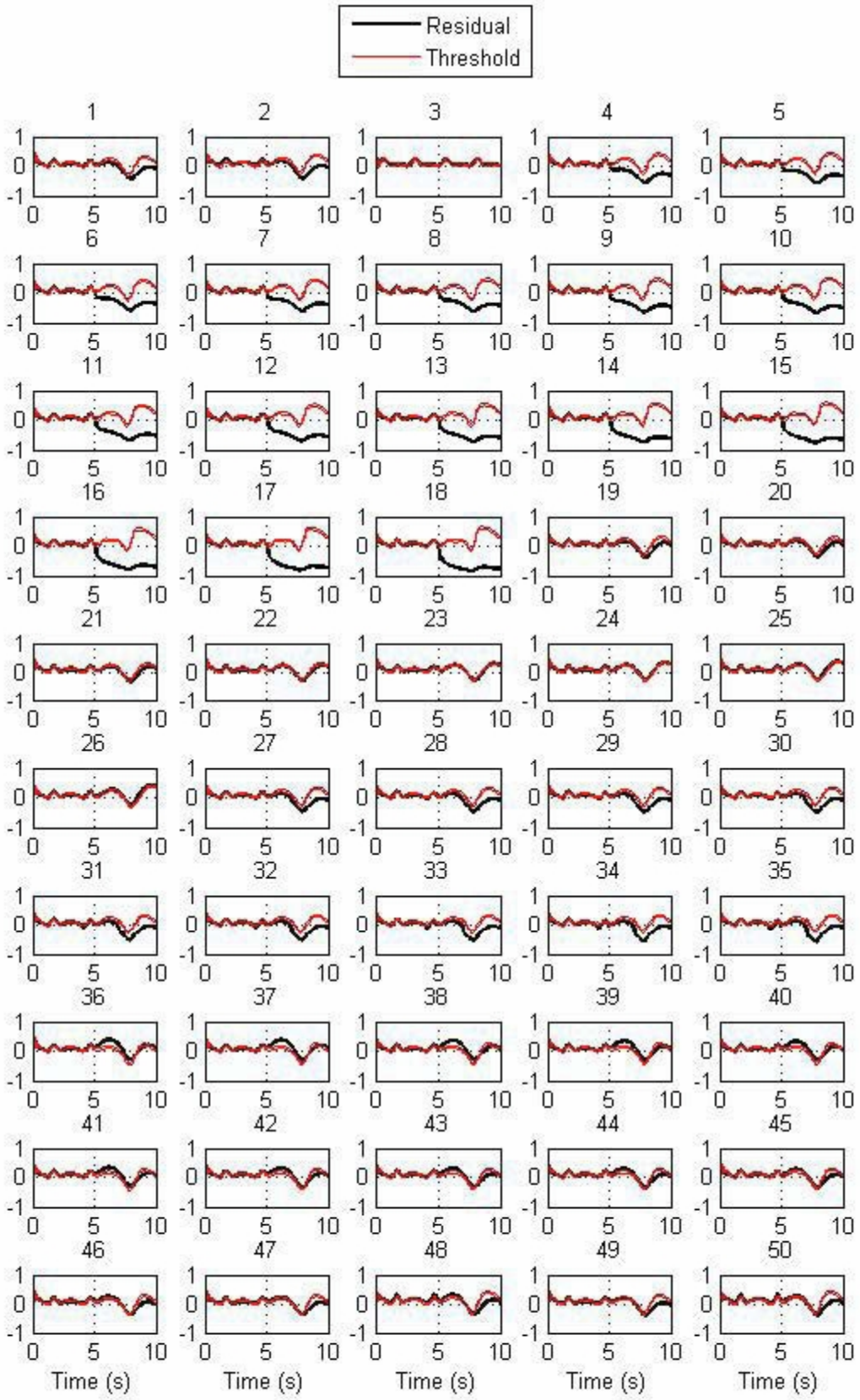

Figure 5.18: Fault identification of the mobile robot by using RBF Neural Networks: Fifth faulty model performance for all the 50 cases

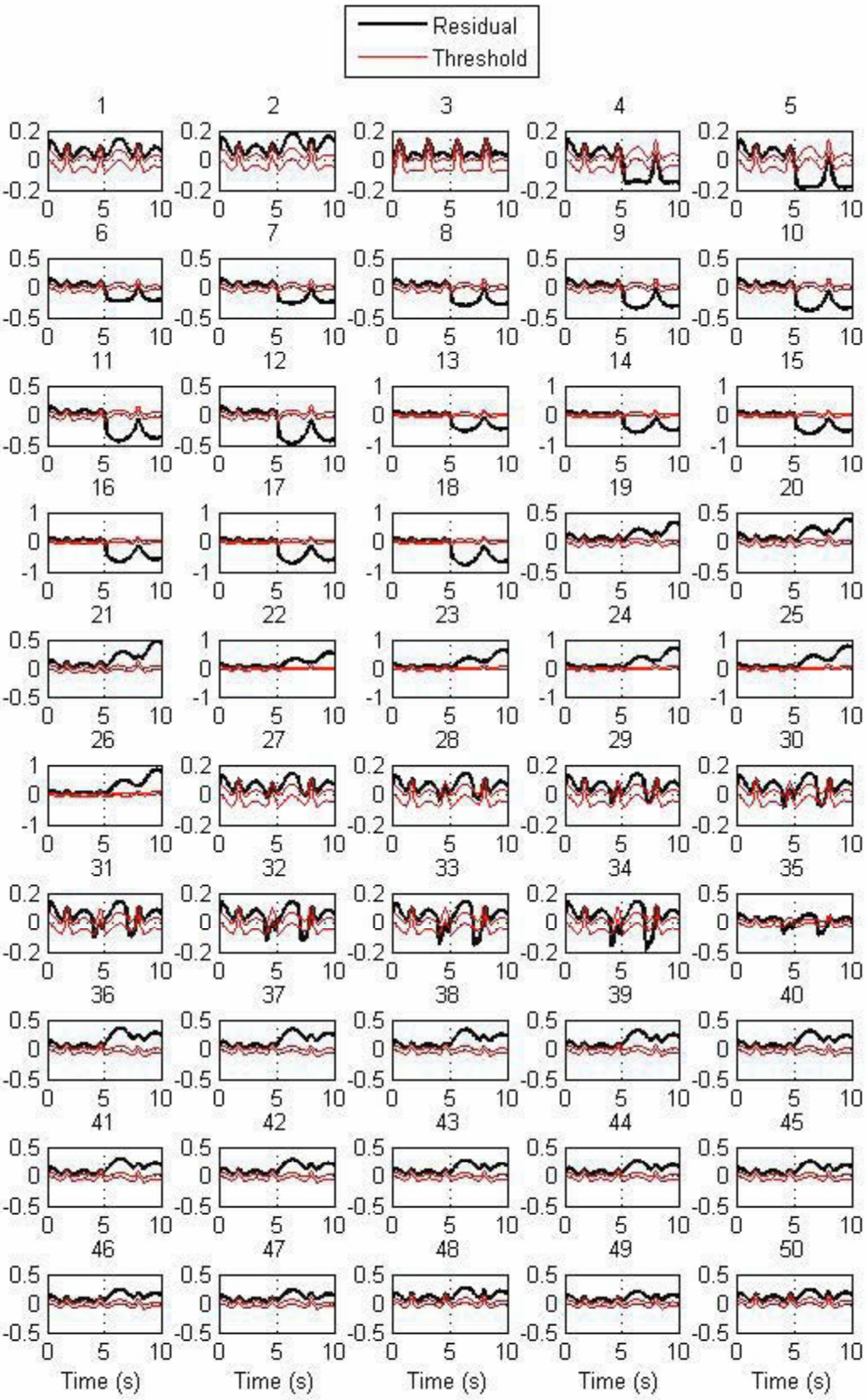

Figure 5.19: Fault identification of the mobile robot by using RBF Neural Networks: Sixth faulty model performance for all the 50 cases

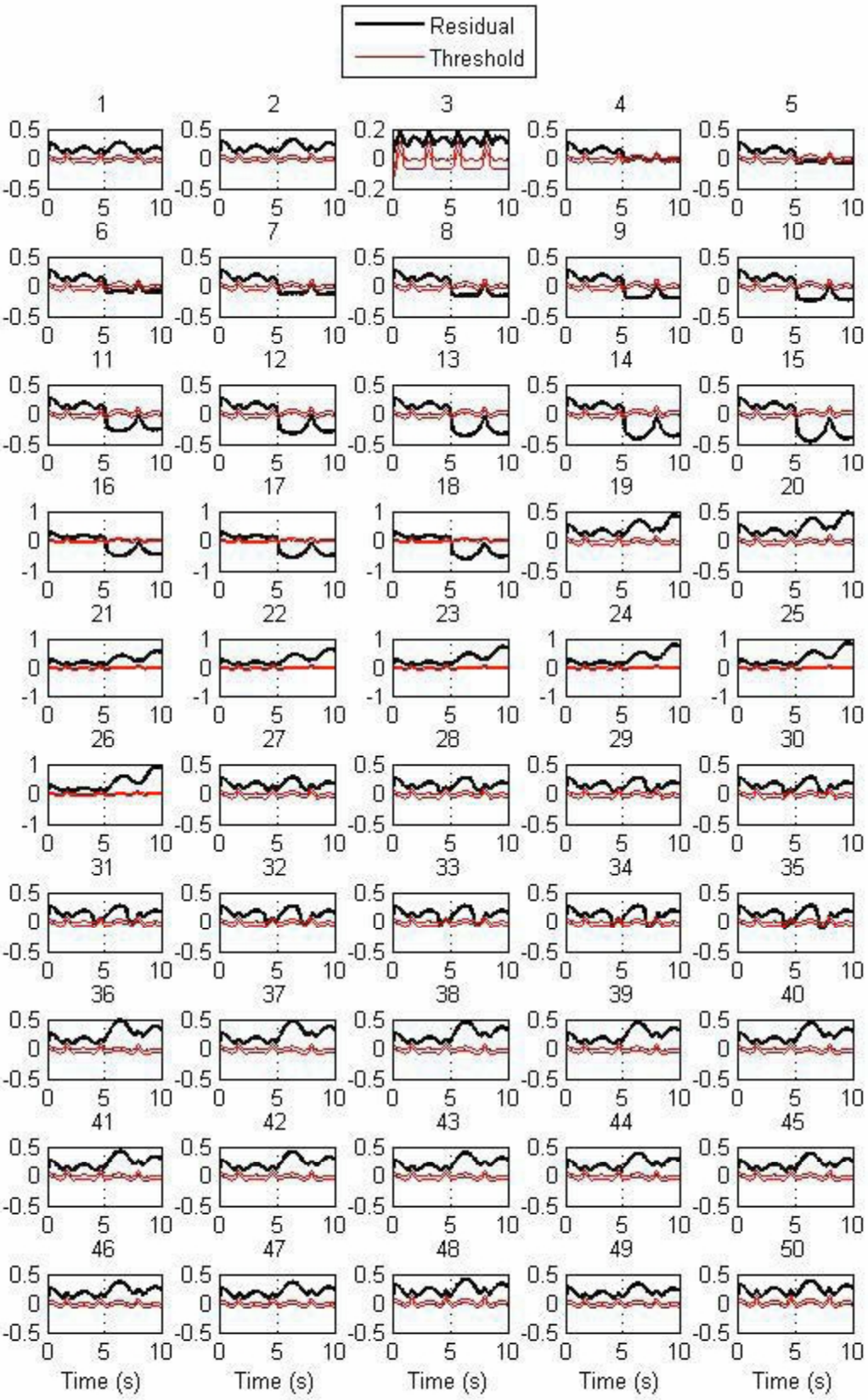

Figure 5.20: Fault identification of the mobile robot by using RBF Neural Networks: Seventh faulty model performance for all the 50 cases

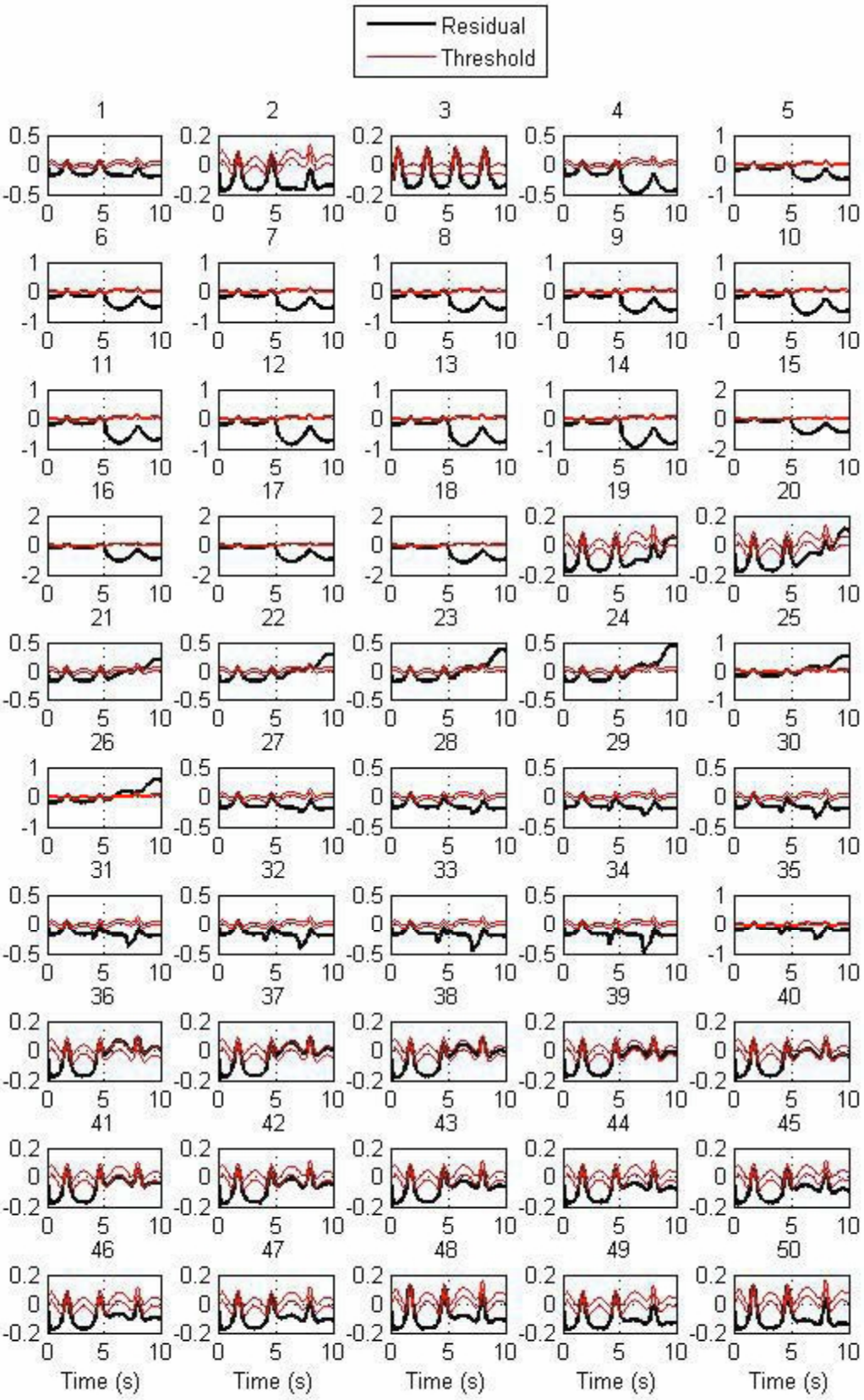

Figure 5.21: Fault identification of the mobile robot by using RBF Neural Networks: Eighth faulty model performance for all the 50 cases.

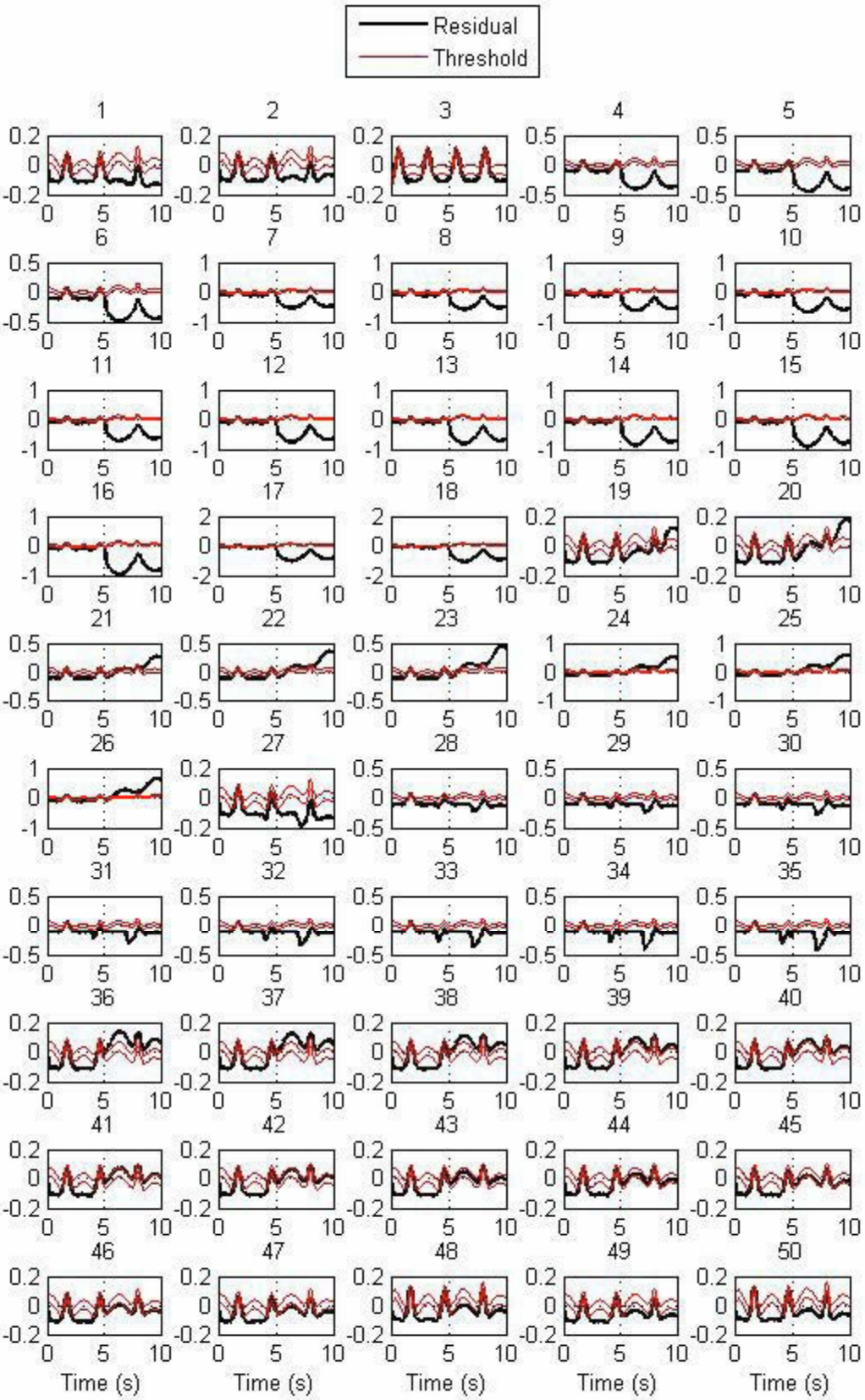

Figure 5.22: Fault identification of the mobile robot by using RBF Neural Networks: Ninth faulty model performance for all the 50 cases

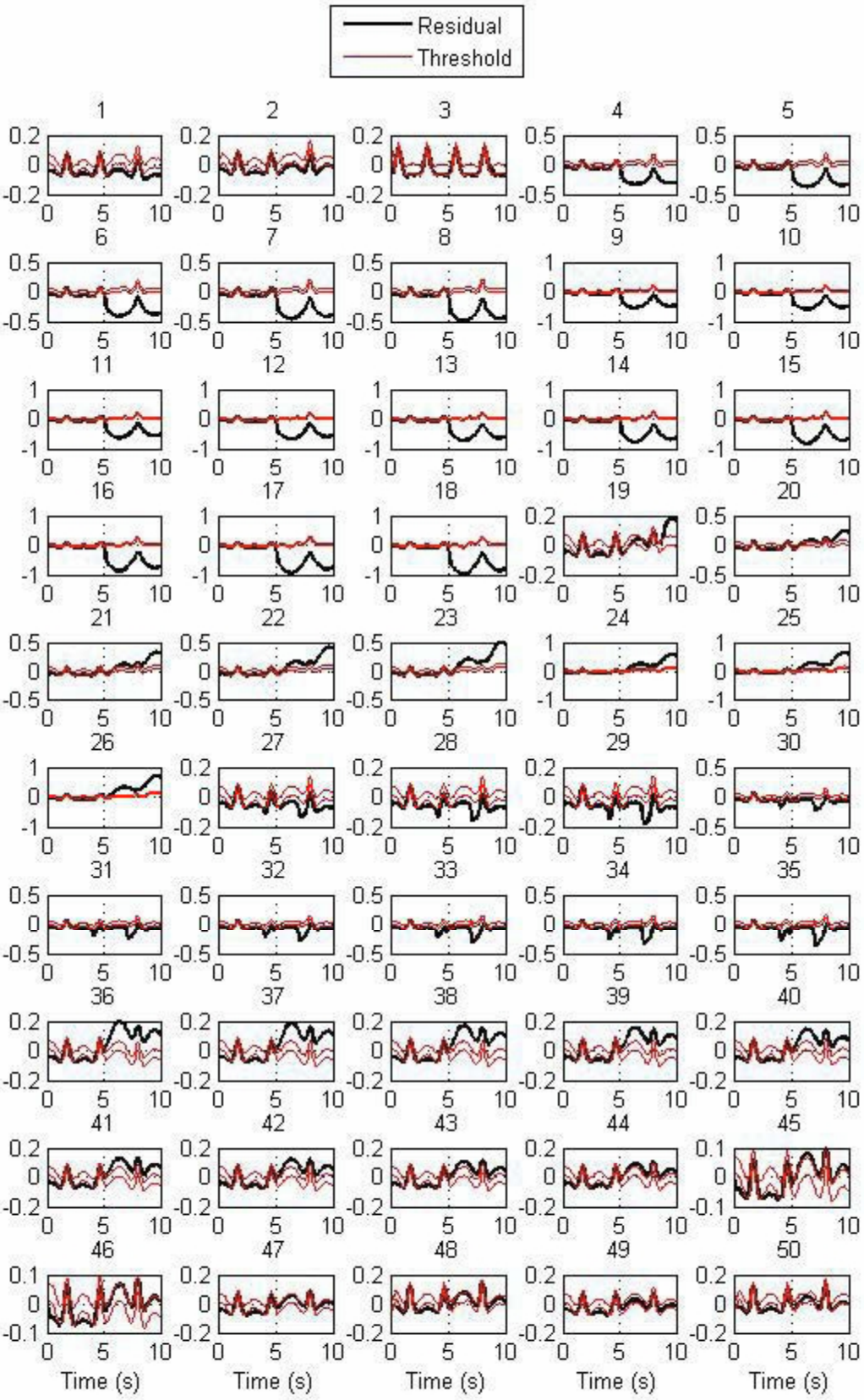

Figure 5.23: Fault identification of the mobile robot by using RBF Neural Networks: Tenth faulty model performance for all the 50 cases.

Class	0	1	2	3	4	5	6	7	8	9	10	TNFD
0	2937	0	0	1	0	1	1	1	3	2	7	0
1	0	222	21	0	0	0	2	33	0	0	0	250
2	0	20	231	34	0	0	0	2	0	0	0	250
3	0	3	24	238	0	0	0	0	0	0	0	250
4	20	0	0	0	131	11	0	0	32	48	53	200
5	10	0	0	0	36	152	0	0	28	15	12	200
6	19	0	0	0	18	18	56	2	0	0	0	80
7	1	79	0	0	9	1	12	69	0	0	0	100
8	0	0	0	0	48	30	0	0	232	17	1	250
9	1	0	0	0	41	38	0	0	19	242	12	250
10	10	0	0	0	39	88	0	0	0	15	229	250

Table 5.6: Confusion Matrix corresponding to the fault identification by using the RBF neural networks and multiple model algorithm. The columns represent the actual classes and the rows represent the estimated classes. TNFD denotes the total number of faulty data in each class.

Class	0	1	2	3	4	5	6	7	8	9	10
0	98.9	0	0	0	0	0	0	0	0.1	0.1	0.2
1	0	88.8	8.4	0	0	0	0.8	13.2	0	0	0
2	0	8	92.4	13.6	0	0	0	0.8	0	0	0
3	0	1.2	9.6	95.2	0	0	0	0	0	0	0
4	10	0	0	0	65.5	5.5	0	0	16	24	26.5
5	5	0	0	0	18	76	0	0	14	7.5	6
6	23.7	0	0	0	22.5	22.5	70	2.5	0	0	0
7	1	79	0	0	9	1	12	69	0	0	0
8	0	0	0	0	19.2	12	0	0	92.8	6.8	0.4
9	0.4	0	0	0	16.4	15.2	0	0	7.6	96.8	4.8
10	4	0	0	0	15.6	35.2	0	0	0	6	91.6

Table 5.7: The confusion matrix shown in percentage corresponding to the fault identification by using the RBF neural networks and multiple model. The columns represent the actual classes and the rows represent the estimated classes.

Class	0	1	2	3	4	5	6	7	8	9	10
Performance	√	√	√	√	+	√	+	√	√	√	√

Table 5.8: The RBF neural networks detectability.

Class	0	1	2	3	4	5	6	7	8	9	10
Performance	√	+	√	√	×	+	+	×	√	√	√

Table 5.9: The RBF neural networks identifiability.

5.4 Conclusions

In this chapter, multiple models of the LLM models and RBF neural networks are used to detect and identify different faults having various severities in the mobile robot platform. Four different fault severities with different sizes are divided into 10 classes. The performance of each method is illustrated by using figures and tables.

To detect and identify faults, linear velocity of the mobile robot is used for representation and modeling the system in different cases. The overall proposed methodology scheme is illustrated in Figure 5.1. Using only one variable to classify the 10 faults can lead to misclassifications, which is mostly visible in the fourth and the fifth classes. Models of these classes are misclassified to other faults and the eighth to the tenth models are misclassified as the fourth and the fifth faulty data.

In conclusion, the RBF models have overall detected and identified faults more effectively. This improved performance is evident in all the faulty classes except the first, the second and the seventh ones.

Chapter 6

Conclusions and Suggestions For Future Work

6.1 Conclusions

This thesis has investigated different methods for representing a mobile robot for the purpose of fault detection and identification. For each method, an adaptive threshold band is proposed to decrease false alarms and improve the fault detection and identification performance. In the considered application, a two wheeled mobile robot is used to track a predefined trajectory in an obstacle free environment.

As the first step, a simulator is developed by using kinematic and dynamic equations of the mobile robot and its actuators. The mobile robot is controlled with an MIMO PD controller to guarantee that the mobile robot is capable of tracking desired trajectories. Four different types of fault containing two actuators and two system faults are implemented in the system abruptly, intermittently and incipiently to be detected and identified by the proposed fault detection and identification scheme.

In order to detect the occurrence of faults in the mobile robot subsystems, two different methods have been proposed. The first method is based on identification of the system

using the local linear models (LLM). In this study, the locally linear model with locally linear model tree (LoLiMoT) learning algorithm is considered because of their ease of interpretability. This property helps to generate adaptive threshold bands by using each local linear model's error which creates locally model thresholds (LMT). The performance of fault detection (FD) by using LMT adaptive threshold bands is compared with the performance of FD using fixed threshold bands and another method of adaptive threshold generation, which uses model error modeling (MEM) technique to identify the residuals. The second approach is based on identification of the system by using radial basis function (RBF) neural networks. In this case, to improve the FD performance and its robustness, the MEM algorithm is used to identify the residuals. Its performance is then compared with the fault detection by using fixed threshold bands.

As described in Chapter 4, the overall performance of the FD scheme by using adaptive threshold bands are much better than the associated fixed ones. It shows that the FD process is more robust when we use adaptive threshold bands. Also, the RBF neural networks with the MEM technique can be considered slightly better than the FD by using the LLM models and corresponding adaptive threshold bands. However, the MEM technique needs another model which can impose certain computational complexity to the FD process.

In Chapter 5, the concept of multiple models is utilized to identify system faults. One healthy and 10 faulty models are constructed to identify 4 different types of faults which are included in 10 fault classes. The best two constructed methods are used to identify the system faults namely the LLM models by using the LMT adaptive threshold bands and the RBF neural networks by using the MEM adaptive threshold bands. The fault identification results are provided in chapter 5 by using the confusion matrices of both methods. Using the confusion matrices, it can be concluded that the fault identification by using the RBF neural networks and the MEM technique has slightly a better performance. However, the number of the trained models in this method is double the number of the trained models

using the LLM models and this fact imposes certain computational burden although better performance is obtained.

To summarize, the main contributions of this thesis can be outlined as follows:

(1) Development of a new simulator for mobile robots containing both the kinematic and the dynamic equations of mobile robot and its actuators. The simulator is capable of implementing different faults in the system.

(2) Development of a new method for residual evaluation by using an adaptive threshold band that is generated by using locally linear models of the system named LMT adaptive threshold band. The improvements of the proposed adaptive threshold bands are shown in simulation results.

(3) Development of two fault detection approaches based on the RBF neural networks and the LLM models for mobile robots using the MEM algorithm to generate adaptive threshold bands.

(4) Development of two fault identification approaches based on the concept of multiple models for a mobile robot.

6.2 Suggestions for Future Studies

Fault diagnosis of nonlinear systems and mobile robots is still an open problem of research. Although this issue has been studied extensively for linear systems, it has remained less investigated for nonlinear systems such as mobile robots.

For future research directions in this field, improving the fault identification performance should be obtained. It would be interesting to determine the cause of overlapping in identification of certain faults and to obtain means to rectify it.

Our proposed methods are tested on a mobile robot simulator. Performing and applying the methods on a real mobile robot is recommended. However, the presented simulator uses the equations of a real robot and the parameters are obtained from a real mobile robot

(Pioneer P3DX) that is available in the lab.

To detect and identify faults in this thesis, only linear velocity of the mobile robot is used to be represented. Modeling other outputs of the system such as the angular velocity should also be studied. Using two outputs, and an appropriate FD technique, the fault detection and identification performance will most likely be improved.

Furthermore, using an on-line gain optimization scheme can improve the modeling performance. In this thesis, model gains are adjusted by using a set of training data so that the model output can follow the system output in a limited range of trajectories with low noise levels. This barrier can be relaxed if an on-line gain optimization scheme is used and developed.

Additionally, a study on fault recovery of mobile robots is suggested. In a complete fault diagnosis system, the fault identification step should lead to further investigations on handling a fault which is known as the recovery phase.

Another issue of interest is fault prognosis. This step which can be added in the overall strategy for handling faults would lead to condition-based maintenance. Consequently, it would be of great practical and economic interest to develop such schemes.

As a matter of fact, there are a large number of open problems in the area of fault diagnosis of nonlinear systems in general and for mobile robots in particular. In this thesis, we have tried to discuss some of the most undeveloped issues and challenges.

Bibliography

[1] R. Isermann, *Fault-Diagnosis Systems: An Introduction From Fault Detection To Fault Tolerance*, Berlin:Springer, 2005.

[2] M. Witczak, *Modeling and Estimation Strategies for Fault Diagnosis of Non-Linear Systems.* Berlin: Springer, 2007.

[3] X. Hu, "Intelligent Fault Diagnosis in Computer Networks," PhD dissertation, Department of Informatics and Mathematical Modeling Building, Technical University of Denmark, Denmark, May 2007.

[4] S. Katipamula and M. R. Brambley, "Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems - A Review, Part II," *American Society of Heating, Refrigerating and Air-Conditioning Engineers,* Inc., Vol. 11, No 2, April 2005.

[5] M. C. Comstock and J. E. Braun, "Literature Review for Applications of Fault Detection and Diagnostic Methods to Vapor Compression Cooling Equipment," Research Project I043-RP, Report 4036-2, December 1999.

[6] S. Poyhonen, "Support Vector Machine Based Classification in Condition Monitoring of Induction Motors," PhD dissertation, Department of Automation and Systems Tech., Helsinki University of Technology Control Engineering Laboratory, Helsinki, Finland, June 2004.

[7] W. G. Zanardelli, E. G. Strangas, H. K. Khalil and J. M. Miller, "Wavelet-Based Methods for the Prognosis of Mechanical and Electrical Failures in Electric Motors, *Mechanical Systems and Signal Processing,* Mar. 2005, pp. 411-426.

[8] H. Zheng, Z. Li and X. Chen, "Gear fault diagnosis based on continuous wavelet transform," *Mechanical Systems and Signal Processing*, vol. 16, 2002, pp. 447-457.

[9] S. I. Roumeliotis, G. S. Sukhatmeyand and G. A. Bekey, "fault detection and isolation in a mobile robot using multiple multiple-model estimation," *In proceeding of the 1988 IEEE International Conference on Robotics and Automation,* vol. 3, Belgium, May 1988, pp. 2223-2228.

[10] M. Luo, D. wang, M. pham, C. B. Low, J. B. Zhang, D. H. Zhang and Y. Z. Zhao, "Model-based fault diagnosis/prognosis for wheeled mobile robots: a review," *32nd Annual Conference of IEEE Industrial Electronics Society,* Nov. 2005, pp. 2267-2272.

[11] J. Carlson and R. R. Murphy, "An Investigation of MML methods for Fault Diagnosis in Mobile Robots," *In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems,* vol.1, 2004, pp. 180-186.

[12] N. Meskin, "Fault detection and isolation in a networked multi-vehicle unmanned system." PhD dissertation, Concordia University, Canada, 2008.

[13] Y. Zhang and J. Jiang, "Bibliographic review on reconfigurable fault-tolerant control systems", *Proc. IFAC Safeprocess,* 2003, pp. 265-276 .

[14] M. M. Tousi, A. G. Aghdam and K. Khorasani, "A hybrid fault diagnosis for a team of unmanned aerial vehicles," *IEEE International Conference on,* June 2009, pp. 1-6.

[15] Y. Zhang, Fault Diagnosis and Fault Tolerant Control Systems, ENGR 691X Lecture Notes, Montreal, Canada: Concordia University, 2010.

[16] Y. Q. Wang, H. Ye and G. Z. Wang, "Fault detection of NCS based on eigendecomposition, adaptive evaluation and adaptive threshold," *International Journal of Control.* vol. 80(12), pp. 1903-1911, 2007.

[17] S.X. Ding, *Model based Fault Diagnosis Techniques*. Berlin: Springer, 2008.

[18] "International Federation of Automatic Control (IFAC)." Internet: http://www.ifac-control.org/, August 1, 2012.

[19] V. R. Venkatasubramanian, R. Rengaswamy, K. Yin and S. N. Kavuri, A review of process fault detection and diagnosis, Part I: Quantitative model-based methods. *Computers in Chemical Engineering,* vol. 27, pp. 293-311, 2003.

[20] A. S. Willsky, "A survey of design methods for failure detection systems." *Automatica,* Vol. 12, pp. 601-611, 1976.

[21] R.Isermann, A. Schwarte and F. Kimmich, "Model based fault detection of a Diesel engine with turbo charger a case study," *Fault Detection, Supervision And Safety Of Technical Processes,* Italy, 2004, pp. 293-306.

[22] J. Chen and R. J. Patton. *Robust Model-based Fault Diagnosis for Dynamic Systems.* Boston: Kluwer Academic Publishers, 1999.

[23] P. M. Frank and S. X. Ding, "Survey of robust residual generation and evaluation methods in observer-based fault detection systems." *Journal of process control,* Vol. 7, No. 6, pp. 403-424, 1997.

[24] J.Gertler, "Fault detection and isolation using parity relations," *Control Engineering Practice,* Vol. 5, No. 5, 1997, pp. 653-661.

[25] M. Omana, "Robust Fault Detection and Isolation Using a Parity Equation Implementation of Directional Residuals." MSc thesis, University of new brunswick, Canada, 2005.

[26] P. M. Frank, S. X. Ding and T. Marcu, "Model-based Fault Diagnosis in Technical Processes" *Transactions of the Institute of Measurement and Control,* vol. 22, no. 1, 2000, pp. 57-101.

[27] C. D. Bocaniala and V. Palade, *Computational Intelligence in Fault Diagnosis Advanced Information and Knowledge Processing.* London: Springer, 2006.

[28] K. Hornik, "Multilayer feedforward networks are universal approximators." *Neural Networks,* vol. 2, pp. 359-366, 1989.

[29] M. Ayoubi, "Fault diagnosis with dynamic neural structure and application to a turbocharger, *Proc. Int. Symp. Fault Detection, Supervision and Safety for technical Process,* 1994, pp. 618-623.

[30] H. Khoshdel Nikkhoo, "Fault Detection in Trajectory Tracking of Wheeled Mobile Robots," M.Sc. thesis, Department of Electrical and Computer Engineering, Concordia University, Canada, 2008.

[31] R. Kohavi and F. Provost, "Glossary of Terms." *Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Machine Learning,* vol. 30, pp. 2-3, 1998.

[32] J. Matthew and X. D. koutsoukos, "Distributed diagnosis in formations of mobile robots," *IEEE Transactions on Robotics*, vol. 23, pp. 353-369, 2007.

[33] G. G. Rigatos, *Modeling and Control for Intelligent Industrial Systems.* Berlin: springer, 2011.

[34] M. G. Perhinschi, M. R. Napolitano, G. Campa, B. Seanor, J. Burken and R. Larson, "An adaptive threshold approach for the design of an actuator failure detection and identification scheme," *IEEE Transactions on Control Systems Technology,* vol. 14, no. 3, May 2006, pp. 519-525.

[35] F. Caccavale, P. cilibrizzi, F. Pierri and L. Villani, "Actuators fault diagnosis for robot manipulators with uncertain model." *Control Engineering Practice,* Vol. 17, pp. 146-157, January 2009.

[36] P. Sundvall and P. Jensfelt, "Fault detection for mobile robots using redundant positioning systems." *IEEE International Conference on Robotics and Automation,* May 2006, pp. 3781-3786.

[37] L. Hongmei, C. Lu, W. Hou and S. Wang, "An adaptive threshold based on support vector machine for fault diagnosis," *8th International Conference on Reliability, Maintainability and Safety,* July 2009, pp. 907-911.

[38] P. M. Frank, "Residual evaluation for fault diagnosis based on adaptive fuzzy thresholds, *IEE Colloquium on Qualitative and Quantitative Modeling Methods for Fault Diagnosis,* 1995, pp. 4/1-4/11.

[39] M. Kowal and J. Korbicz, "Fault Detection under Fuzzy Model Uncertainty" *International Journal of Automation and Computing.* Vol. 4(2), pp. 117-124, 2007.

[40] J. KORBICZ, "Robust fault detection using analytical and soft computing methods," *Bulletin of the polish academy of science,* Vol. 54, No. 1, 2006, pp. 75-88.

[41] J. Yong, W. Hongguang, F. Lijin and Z. Mingyang, "A Combined Logistic and Model Based Approach for Fault Detection and Identification in a Climbing Robot," *IEEE International Conference on Robotics and Biomimetics,* Dec. 2006, pp. 1512-1516.

[42] H. A. Nozari, H. D. Banadaki, M. A. Shoorehdeli and S. Simani, "Model-based Fault Detection and Isolation Using Neural Networks: An Industrial Gas Turbine Case Study," *21st International Conference on Systems Engineering*, Aug. 2011, pp.26-31.

[43] O. Nelles, *Nonlinear System Identification,* Berlin: Springer, 2001.

[44] M. T. Hagan , H. B. Demuth and M. Beale, *Neural network design,* Boston: PWS Publishing Co., 1997.

[45] D. H. Rao, "Neural networks in robotics and control: some perspectives," *International Conference on Industrial Automation and Control,* Jan. 1995, pp. 451-456.

[46] J. Carlson, R. R. Murphy and A. Nelson, "Follow-up analysis of mobile robot failures," *IEEE International Conference on Robotics and Automation,* vol.5, 2004, pp. 4987-4994.

[47] P. Goel, G. Dedeoglu, S. Roumeliotis and G. Sukhatme, "Fault detection and identification in a mobile robot using multiple model estimation and neural network," *IEEE International Conference on Robotics and Automation,* vol. 3, 2000, pp. 2302-2309.

[48] L. A. Zadeh, "Fuzzy Sets." *Information and control* vol. 8, pp. 338-353, 1965.

[49] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision process." *IEEE Transaction on Systems, Man and Cybernetics,* vol.3, No.1, 1973, pp. 28-44.

[50] M. Mamdani, "Advances in the linguistic synthesis of fuzzy controllers." *International Journal of Man-Machine Studies,* vol. 8, pp. 669-678, 1976.

[51] M. Pantoquilho, J. J. Neto, N. Viana, J. Moura-Pires and R. A. Ribeiro, "Online and offline monitoring and diagnosis of spacecraft and space weather status." *Proceedings of the EUROFUSE04- Workshop on Data and Knowledge Engineering,* Poland, 2004.

[52] R. Ribeiro, "Fuzzy Space Monitoring and Fault Detection Applications," *Journal of Decision Systems.* vol. 15(2-3), pp. 267-286, 2006.

[53] E. Ruano, *Intelligent Control Systems Using Computational Intelligence Techniques.* London: Institution of Engineering and Technology, 2005.

[54] V. Palade, R. J. Patton, F. J. Uppal, J. Quevedo and S. Delay, "Fault Diagnosis of An Industrial Gas Turbine Using Neuro-Fuzzy Methods." *In Proceedings of the 15th IFAC World Congress,* pp. 2477-2482, July 2002.

[55] K. Patan, M. Witczak and J. Korbicz, "Towards robustness in neural network based fault diagnosis." *International Journal of Applied Mathematics and Computer Science,* Vol. 18, No. 4, pp. 443-454, 2008.

[56] M. Mokhtare, S. Vahed, Sh. M. Aliyari and A. Fatehi, "Modeling and identification of catalytic reformer unit using locally linear model trees," *19th Iranian Conference on Electrical Engineering (ICEE),* May 2011, pp. 1-6,

[57] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks,* vol. 1, no. 1, 1990, pp. 4-27.

[58] O. Nelles, S. Sinsel and R. Isermann, "Local basis function networks for identification of a turbocharger," *International Conference on Control,* vol.1, Sept. 1996, pp. 7-12.

[59] M. Ayoubi, "Nonlinear dynamic systems identification with dynamic neural networks for fault diagnosis in technical processes," *IEEE International Conference on Systems, Man, and Cybernetics, 1994. Humans, Information and Technology,* vol.3, Oct 1994, pp. 2120-2125.

[60] A. Yazdizadeh and K. Khorasani, "Identification of a class of nonlinear systems using dynamic neural network structures," *International Conference on Neural Networks,* vol. 1, Jun 1997, pp. 194-198.

[61] J. C. Principe and M.A. Motter, "System Identification with Dynamic Neural Networks," *In World Congress on Neural Networks,* 1994, pp. 284-289.

[62] I. Alzyoud and K. Khorasani, "Detection of actuator faults using a dynamic neural network for the attitude control subsystem of a satellite," *Proceeding of International Joint Conference on Neural Networks,* vol. 3, 2005, pp. 1746-1751.

[63] A. Yazdizadeh, "Identification of Nonlinear Systems Using Dynamic Neural Networks." PhD dissertation, Concordia University, Canada, 1997.

[64] J. Hauth, "Grey-Box Modeling for Nonlinear Systems," PhD dissertation, Technischen University Kaiserslautern, Germany, December 2008.

[65] O. Nelles and R. Isermann, "Basis function networks for interpolation of local linear models," *Proceedings of the 35th IEEE on Decision and Control,* vol. 1, Dec. 1996, pp. 470-475.

[66] A. Pedram, M. R. Jamali, T. Pedram, S. M. Fakhraie and C. Lucas, "Local Linear Model Tree Reconfigurable Parallel Hardware," *World Academy of Science, Engineering and Technology,* vol. 13, 2006, pp. 198-201.

[67] H. Nourzadeh, A. Fatehi, B. Labibi and B. N. Araabi, "An Experimental Nonlinear System Identification Based on Local Linear Neuro-Fuzzy Models," *IEEE International Conference on Industrial Technology,* 2006, pp. 2274-2279.

[68] Neural Networks, http://www.statsoft.com/textbook/neural-networks/, (current August 1, 2012).

[69] J. Mark and L. Orr, "Introduction to Radial Basis Function Networks, Centre for Cognitive Science," University of Edinburgh, Scotland, April 1996.

[70] K. M. Bossley, "Neurofuzzy modeling approaches in system identification." PhD thesis, University of Southampton, Southampton, 1997.

[71] A. Gholipour, C. Lucas, B. N. Araabi, M. Mirmomeni and M. Shafiee, "Extracting the main patterns of natural time series for long-term neuro fuzzy prediction," *Neural Computing and Applications,* vol. 16, Aug. 2006, pp. 383-393.

[72] M. Jalili-Kharaajoo, R. Ranji and H. Bagherzadeh, "Predictive control of fossil power plant based on locally linear model tree (LoLiMoT)," *Proc. of IEEE,* vol. 1, 2003, pp. 633-638.

[73] N. Sarkar, X. Yun and V. Kumar, "Control of Mechanical Systems with Rolling Constraints: Application to Dynamic Control of Mobile Robots." *The International Journal of Robotics Research,* vol. 13 (1), pp. 55-69, 1992.

[74] Y. Yamamoto and X.Yun, "Coordinating Locomotion and manipulation of a mobile manipulator," *IEEE Transaction in Automatic Control,* 1994, pp. 1326-1332.

[75] J. Mireles, "Kinematic Models of Mobile Robots." Selected notes from Robotica, Summer 2004.

[76] X. Yan and Y. Yamamoto, "On Feedback Linearization of Mobile Robots," Technical Report, University of Pennsylvania, USA, 1992.

[77] X. Yun and Y. Yamamoto, "Internal dynamics of a wheeled mobile robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 1993, vol. 2, pp. 1288-1294.

[78] J. Choi and B. Kook Kim, "Near minimum-time direct voltage control algorithms for wheeled mobile robots with current and voltage constraints," *Robotica,* vol. 19, pp. 29-39, 2001.

[79] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, 9th Ed., Philadelphia: Prentice-Hall, 2001.

[80] R. Isermann, *Fault-Diagnosis Applications, Model-Based Condition Monitoring: Actuators, Drives, Machinery, Plants, Sensors, and Fault-tolerant Systems,* Berlin: Springer, 2011.

[81] Maplesoft. "DC Motor Model. Internet:http://www.maplesoft.com/applications/view.aspx?SID=1458, May 5, 2008 [August 1, 2012].

[82] L. Huang, "Speed Control of Differentially Driven Wheeled Mobile Robots-Model-Based Adaptive Approach," *Journal of Robotic Systems,* vol. 22, pp. 891-895, 2003.

[83] A. De Luca, G. Oriolo and C. Samson, *Robot motion planning and control*, London: Springer, 1998.

[84] J.E. Bares and D.S. Wettergreen, "Dante ii: Technical description, results and lessons learned." *International Journal of Robotics Research,* vol. 18, pp. 621-664, July 1999.

[85] V. Verma and R. Simmons, "Scalable robot fault detection and identification," *Robotics and Autonomous Systems,* vol. 54, 2006, pp. 184-191.

[86] W. Dixon, I. D. Walker and D. M. Dawson, "Fault Detection for Wheeled Mobile Robots with Parametric Uncertainty," *Proc. of IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM),* vol. 2, pp. 1245-1250, 2001.

[87] M. Hashimoto, T. Ishii and K. Takahashi, "Sensor fault detection and isolation for mobile robots in a multi-robot team," *35th Annual Conference of IEEE Industrial Electronics, IECON '09,* Nov. 2009, pp. 2348-2353.

[88] Z. Li, "Fault diagnosis and fault tolerant control of mobile robot based on neural networks," *International Conference on Machine Learning and Cybernetics,* vol. 2, 2009, pp. 1077-1081.

[89] O. Moseler and R. Isermann, "Model-Based Fault Detection for a Brushless DC Motor Using Parameter Estimation," *Proceedings of the 24th Annual Conference of the IEEE, Industrial Electronics Society, IECON'98,* vol. 4, 1998, pp. 1956-1960.

[90] A. C. Soh and R. Z. Rahman, "Fault Detection and Diagnosis for DC Motor in Robot Movement System using Neural Network," *The Pacific Journal of Science and Technology,* vol. 10, pp. 35-43, 2010.

[91] O. Moseler and R. Isermann, "Application of model-based fault detection to a brushless DC motor," *IEEE Transactions on Industrial Electronics,* 2000, pp. 1015-1020.

[92] Y. Zhang and J. Jiang, "Active fault-tolerant control system against partial actuator failures," *IEE Proceedings on control theory applications,* vol. 149, January 2002, pp. 95-104.

[93] S. Simani, C. Fantuzzi C. and R. J. Patton, *Model- based fault diagnosis in dynamic systems using identification techniques.* London: Springer, 2003.

[94] C. C. Ward and K. Iagnemma, "Model-Based Wheel Slip Detection for Outdoor Mobile Robots," *IEEE International Conference on Robotics and Automation,* 2007, pp. 2724-2729.

[95] R. A. Carrasco, F. Nunez and A. Cipriano, "Fault detection and isolation in cooperative mobile robots using multilayer architecture and dynamic observers," *Robotica,* vol. 29, 2011, pp 555-562.

[96] C. C. Ward and K. Iagnemma, "A Dynamic-Model-Based Wheel Slip Detector for Mobile Robots on Outdoor Terrain," *IEEE Transactions on Robotics,* vol.24, Aug. 2008, pp.821-831.

[97] N. Sidek and N. Sarkar, "Integrating Actuator Fault and Wheel Slippage Detections within FDI Framework," *Proceedings of the 5th WSEAS Int. Conf. on Circuits, Systems, Electronics, Control and Signal Processing,* vol. 6, 2006, pp. 298-303.

[98] E. N. Skoundrianos and S. G. Tzafestas, "Finding fault - fault diagnosis on the wheels of a mobile robot using local model neural networks," *Robotics and Automation Magazine, IEEE,* vol. 11, 2004, pp. 83-90.

[99] W. Reinelt, A. Garulli and L. Ljung, "Comparing different approaches to model error modeling in robust identification," *Automatica,* vol. 38(5), 2002, pp. 787-803.

[100] H. A. Nozari, M. Sh. Aliyari, S. Simani and H. De. Banadaki, "Model-based robust fault detection and isolation of an industrial gas turbine prototype using soft computing techniques." *Neurocomputing,* vol. 91, 2012, pp. 29-47.

[101] K. Duzinkiewicz, "Set membership estimation of parameters and variables in dynamic networks by recursive algorithms with a moving measurement window." *International Journal of Applied Mathematics and Computer Science,* vol. 16(2), pp. 209-217, 2006.

[102] E. Walter and L. Pronzato, *Identification of Parametric Models from Experimental Data.* London: Springer, 1997.

[103] M. Krstic, I. Kanellakopoulos and P. Kokotovic, *Nonlinear and adaptive control design.* New York: Wiley, 1995.