Fault Detection, Isolation and Identification of Autonomous Underwater Vehicles Using Dynamic Neural Networks and Genetic Algorithms

Shaghayegh Shahrokhi Tehrani

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
Concordia University
Montréal, Québec, Canada

September 2015

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Shaghayegh Shahrokhi Tehrani

Entitled: "Fault Detection, Isolation and Identification of Autonomous Underwater Vehicles Using Dynamic Neural Networks and Genetic Algorithms"

and submitted in partial fulfillment 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:

_____ Chair
Dr. R. Raut

_____ Examiner, External
Dr. M. Chen (MIE) To the Program

_____ Examiner
Dr. S. Hashtrudi Zad

_____ Supervisor
Dr. K. Khorasani

_____ Supervisor
Dr. F. Abdollahi

Approved by: _____
Dr. W. E. Lynch, Chair
Department of Electrical and Computer Engineering

_____20_____ _____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Fault Detection, Isolation and Identification of Autonomous Underwater Vehicles Using Dynamic Neural Networks and Genetic Algorithms

Shaghayegh Shahrokhi Tehrani

The main objective of this thesis is to propose and develop a fault detection, isolation and identification scheme based on dynamic neural networks (DNNs) and genetic algorithm (GA) for thrusters of the autonomous underwater vehicles (AUVs) which provide the force for performing the formation missions. In order to achieve the fault detection task, in this thesis two level of fault detection are proposed, I) Agent-level fault detection (ALFD) and II) Formation-level fault detection (FLFD). The proposed agent-level fault detection scheme includes a dynamic neural network which is trained with absolute measurements and states of each thruster in the AUV. The genetic algorithm is used in order to train the DNN. The results from simulations indicate that although the ALFD scheme can detect the high severity faults, for low severity faults the accuracy is not satisfy our expectations. Therefore, a formation-level fault detection scheme is developed. In the proposed formation-level fault detection scheme, a fault detection unit consist of two dynamic neural networks corresponding to its adjacent neighbors, is employed in each AUV to detect the fault in formation. Each DNN of the fault detection unit is trained with one relative and one absolute measurements. Similar to ALFD scheme, these two DNNs are trained with GA. The simulation results and confusion matrix analysis indicate that our proposed FLFD can detect both low severity and high severity faults with high level of accuracy compare to ALFD scheme.

In order to indicate the type and severity of the occurred fault the agent-level and formation-level fault isolation and identification schemes are developed and their performances are compared. In the proposed fault isolation and identification schemes, two neural networks are employed for isolating the type of the fault in the thruster of

the AUV and determining the severity of the occurred fault. In the fist step, a multi layer perceptron (MLP) neural network categorize the type of the fault into thruster blocking, flooded thruster and loss of effectiveness in rotor and in the next step a MLP neural network classify the severity into low, medium and high. The neural networks in fault isolation and identification schemes are trained based on genetic algorithm with various data sets which are obtained through different faulty operating condition of the AUV. The simulation results and the confusion matrix analysis indicate that the proposed formation-level fault isolation and identification schemes have a better performance comparing to agent-level schemes and they are capable of isolating and identifying the faults with high level of accuracy and precision.

Acknowledgments

I would like to express my gratitude to my supervisor Professor Khashayar Khorasani for his sincere guidance and support during my Master studies. It was great part of my academic life to work under his supervision.

I would also like to acknowledge the advice and encouragement of my co-supervisor Professor Farzaneh Abdollahi in the process of preparation and completion of my thesis.

I thank my father Shahrokh Shahrokhi Tehrani, my mother Behdokht Ghazvani, my brother Shervin Shahrokhi Tehrani and my aunt Behnaz Ghazvani for their love and support in these years. I would like to thank Farshid Faal for supporting me during my research.

Finally, there are friends whose encouragement and support are beneficial to my research work. I would like to thank all of them.

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

AUV	Autonomous Underwater Vehicle
UAV	Unmanned Aerial Vehicle
FDII	Fault Detection, Isolation and Identification
ANN	Artificial Neural Network
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
TDNN	Time Delay Neural Network
ATDNN	Adaptive Time Delay Neural Network
RBF	Radial Basis Function
LBL	Long Base Line
FRP	Formation Reference Point
GA	Genetic Algorithm
EA	Evolutionary Algorithm
BP	Back Propagation
ESS	Evolutionary Stable Strategy
NN	Neural Network
MLP	Multi Layer Perceptron
STRE	Short-term Reproduction Expectancy
EANN	Evolutionary Artificial Neural Network
RCGA	Real-coded Genetic Algorithm
HTGA	Hybrid Taguchi-Genetic Algorithm
MLNN	Multi Layer Neural Network
MSE	Mean Squared Error
CPU	Central Processing Unit
FDI	Fault Detection and Isolation
DNN	Dynamic Neural Network
WNN	Wavelet Neural Network
IMU	Inertial Measurement Unit
GNSS	Global Navigation Satellite System
EKF	Extended Kalman Filter
TCM	Thruster Control Matrix
ROV	Remotely Operated Vehicle

DOF	Degree of Freedom
RFDI	Robust Failure Detection and Isolation
IDM	Intelligent Decision Making
FES	Fuzzy Expert System
PCA	Principle Component Analysis
HFDI	Hierarchical Fault Detection and Isolation
UUV	Unmanned Underwater Vehichle
SVM	Support Vector Machine
FDAS	Fault Diagnosis and Accommodation System
FDS	Fault Diagnosis Subsystem
FAS	Fault Accommodation Subsystem
FDU	Fault Detection Unit
PPCA	Partial Principal Component Analysis
QCV	Qualification of Contributing Variables
LQR	Linear Quadratic Regulator
ICMAC	Improved Credit Assignment Cerebellar Model Articulation Controllers
GDP	Grey Dynamic Prediction
FTC	Fault-Tolerant Control
SMC	Sliding Mode Control
FCA-CMAC	Credit Assignment-based Fuzzy Cerebellar Model Articulation Controllers
SFDC	Simultaneous Fault Detection and Control
LMI	Linear Matrix Inequality
SOTSDP	Second-Order Taylor Series Dynamic Prediction
SFDIT	Simultaneous Fault Detection, Isolation and Tracking
FWSVDD	Fuzzy Weighted Support Vector Domain Description
DNM	Dynamic Neural Model
EDBP	Extended Dynamic Back-Propagation
EC	Evolutionary Computation
ES	Evolutionary Strategy
EP	Evolutionary Programming
EA	Evolutionary Algorithm
EANN	Evolutionary Artificial Neural Network
UVMS	Underwater Vehicle-Manipulator System

Chapter 1

Introduction

1.1 Statement of the Problem

Autonomous Underwater Vehicles (AUVs) are complex systems often performing missions in unstructured and hazardous environments. AUVs play an important role in performing underwater tasks, in both civilian and military applications [1, 2].

As the complexities of the missions are increasing, a single AUV could not accomplish the mission tasks individually; therefore applying multiple AUVs has received lots of attention in recent years [3, 4]. The basic idea is to use relatively inexpensive, simple and small AUVs instead of expensive specialized ones to cooperatively fulfill difficult and complex underwater missions. The mentioned approach can increase the overall reliability of the system while decreasing the mission complexity or fulfill the missions that cannot be executed by a single AUV [5, 6]. The degree of autonomy, capability of fault detection, isolation and identification are the crucial factors in these systems to accomplish the tasks of the missions successfully.

Over the last few years, lots of attentions have been paid to the research on the cooperative control and formation control of multiple autonomous agents, especially on the cooperative control of groups of robots, UAVs and AUVs [7 - 14].

Formation of multiple AUVs is a critical technology for underwater missions. In formation of AUVs the precise formation keeping and collision avoidance between agents of the formation while doing the mission tasks are the main issues that must be considered. These issues can be solved with applying the appropriate control laws and precise sensors and actuators in the formation of AUVs. However, the performance of the formation can be affected by the possible faults in any of these components; thus in order to fulfill the mission successfully the fault diagnosis system is a mandatory part that should be considered for this type of missions.

In general, faults are the abnormal conditions that occur when the plant or its instruments deviate from their normal behavior. The system that is capable of detecting, isolating and identifying of the faults is called Fault Detection, Isolation and Identification (FDII) system. The FDII system contains three parts. The first one is the fault detection system that is in charge of making decision about the working condition of the monitored system, which can be the normal (healthy) or abnormal (faulty). The second part is the fault isolation that determines the location of the fault in the monitored system. In the third section the fault identification indicates the type of the occurred fault in the monitored system.

In recent years, FDII systems have been developed for the AUVs. Different approaches have been presented in the literatures on FDII systems, which consider the faults of sensors, actuators and controllers [111 - 144]. However, majority of these studies have been done for a single AUV and not much works have been developed in the literatures on FDII system for multiple AUVs in a formation.

The thrusters are considered as the only actuators in a large number of underwater vehicles; thus they are considered as one of the most common and important sources of the faults in AUVs. Consequently, developing a fault diagnosis system for thrusters of the AUV is essential. The main objective of this work is detecting, isolating and

identifying the faults in thrusters of multiple autonomous underwater vehicles by applying artificial neural networks (ANNs).

1.2 Literature Review

1.2.1 Fault Detection, Isolation and Identification

Fault diagnosis for control systems such as autonomous underwater vehicles have been widely studied in recent years, since sometimes it is critical for the vehicle to perform the mission in the presence of faults in sensors, actuators and plant. The main task of the fault diagnosis is to detect and isolate the occurring faults and indicate their severity in plant, actuators and sensors to avoid the overall failure of the monitored system.

1.2.1.1 Classification of Fault Diagnosis Systems

The fault diagnosis (i.e. fault detection, isolation and identification) methods can be categorized in two classes: I) Model-based fault diagnosis and II) History-based fault diagnosis. Each of these categories are divided into qualitative and quantitative methods [69]. In the model-based fault diagnosis a priori knowledge about the model that is expanded based on some principle understanding of the physics of the process is an essential requirement. In quantitative model-based methods, these understandings are considered as the mathematical functional relationships between the inputs and outputs of the system, while in qualitative model-based techniques these mathematical functional relationships are stated in terms of qualitative functions centered around different units in a process. In opposition to the model-based approaches, in history-based methods the availability of a large amount of historical process data is necessary. These data are transformed and presented throughout a process, named

as feature extraction, as a priori knowledge for the diagnostic system [69 - 71]. The classification of diagnosis systems is depicted in Figure 1.1.

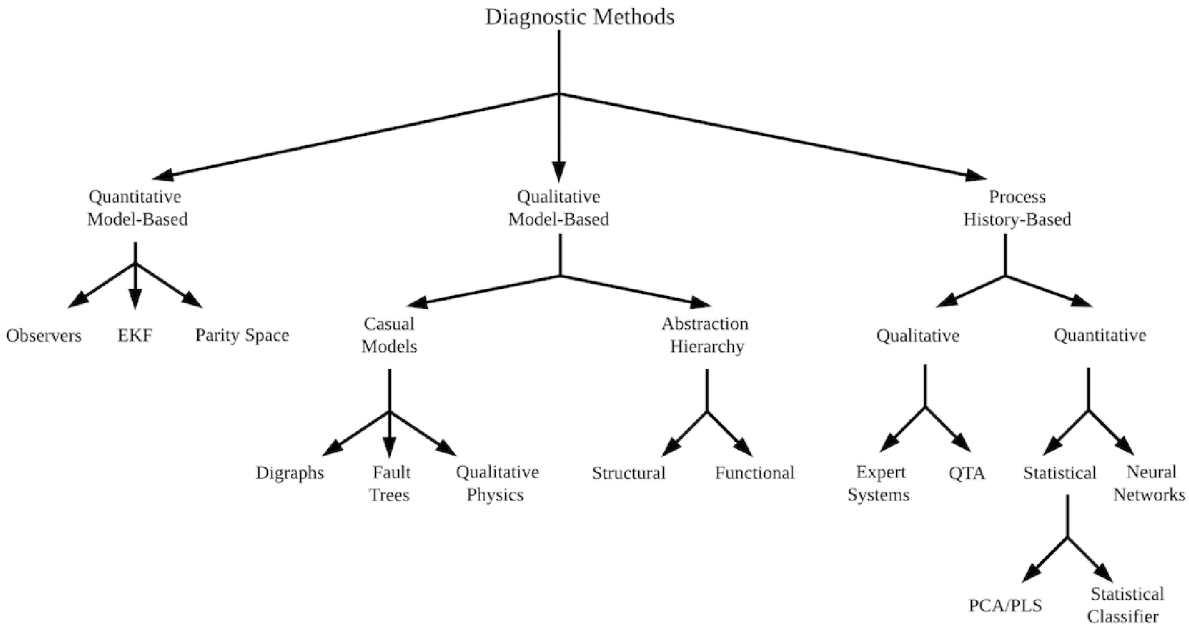


Figure 1.1: Classification of fault diagnosis systems [69].

1.2.1.1.1 Model-based Fault Diagnosis

One of the usual methods in fault diagnosis is the hardware redundancy that applies multiple sensors, actuators, and computers to measure and control the system. The major challenges with the hardware redundancy are related to the extra cost and the additional space that should be provided for the redundant components. In order to deal with these problems, another fault diagnosis approach based on analytical redundancy is proposed. This method utilizes the redundant analytical relationships among system inputs and measured outputs to generate the residual signals. As this scheme uses a mathematical model of the system, it named as the model-based fault diagnosis [72]. The Model-based fault diagnosis is capable of detecting, isolating and identifying the faults on a system by employing the methods that extract features from measured signals and use a priori information on the process available in term

of a mathematical model. The structure of the model-based FDI is illustrated in Figure 1.2. The two fundamental blocks in the model-based FDI structure are: I)

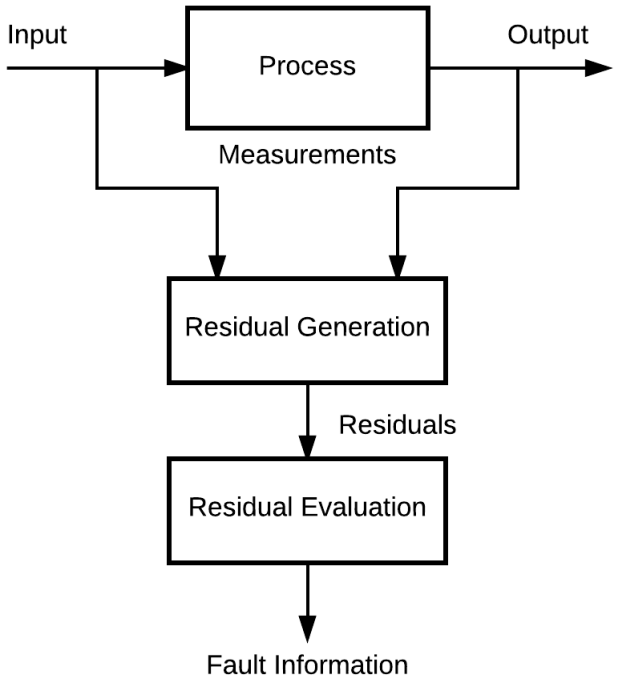


Figure 1.2: Structure of model-based FDI system [72].

Residual generation and II) Residual evaluation. Residual generation produces signals by utilizing the available inputs and outputs from the monitored system. These generated signals that are named as residuals are used to indicate whether or not the fault occurs. The residual evaluation block performs the threshold test on the residuals and makes decision if any faults have occurred.

According to the Figure 1.1 , the model-based FDI is categorized in two groups, quantitative and qualitative approaches. All the approaches in the quantitative category named as observer-based, parameter estimation and parity space, require a mathematical model of the monitored system [73 - 75]. The parameter estimation technique determines the unknown process parameters with measuring the input and

output signals based on the model of the system; therefore the faults can be diagnosed if the estimated parameter value deviates from its nominal value [76, 77]. The idea of the parity space methods is to generate auxiliary signals (residuals or parity vectors) that are independent of system operating conditions and system inputs under nominal operating conditions while carrying fault information [78]. The task of parity space fault diagnosis is to construct a space for analyzing the residuals (parity vectors) [79 - 82]. The observer-based method, state observers or output observers, utilizes the observers that are the dynamical systems in order to estimate the states and consequently the outputs of a process. Thus, the residuals can be calculated as the difference between the estimated output and the measured output. Two observers are applied in the literatures for estimating the output of the system that are named as Luenberger observers [83 - 90] and Kalman filters [91 - 93], which are utilized in deterministic setting and stochastic setting respectively.

1.2.1.1.2 History-based Fault Diagnosis

The main issue in model-based fault detection is related to the accuracy of the mathematical model that describes the behavior of the monitored system. This issue arises when the modeling uncertainties make it impossible to fully understand the monitored system and obtain complete information about it. In other words, finding the precise mathematical model of the monitored system that can be very sensitive to modeling errors, parameter variation, noise and disturbances is challenging. In order to overcome this issue, the history-based FDI is developed. The history-based fault diagnosis approach requires the large amount of the historical process data, which are transformed by diverse techniques to provide a priori knowledge for the diagnostic system. The history-based process is divided in two category: I) Qualitative and II) Quantitative. The major methods in each of these categories are the expert

systems and trend modeling for the qualitative and the non-statistical and the statistical for the quantitative. In past decades, the intelligence and learning theory approaches in history-based such as neural network, fuzzy-logic, neuro-fuzzy systems, and genetic algorithms have been developed and utilized as a fault diagnosis system in different engineering processes. These novel methods provide a way that can get over the above-mentioned problems of the model-based [94, 95]. Neural networks based on their nonlinear function approximation property and their ability to learn and reproduce the system from historical data, are perfect mathematical tools when the system behavior is provided in a form of large quantitative data sets. Powerful nonlinear mapping properties, noise tolerance, self-learning and self-adapting, and parallel processing capabilities are the advantages of the neural networks over the other fault diagnosis methods [23, 24, 26, 96 - 99].

As mentioned previously, model-based fault diagnosis methods depend on the mathematical models of the system to determine the deviations between healthy and faulty operating modes. Developing the mathematical models for complex and nonlinear systems is difficult and time consuming. In addition, in order to validate the model-based approach, lots of experimentation is required. According to the aforementioned drawbacks of the model-based method, in this thesis the fault diagnosis approach based on artificial neural networks is applied.

1.2.1.1.3 Hybrid Methods of Fault Diagnosis

As mentioned in previous sections, no single method can provide all the requirements for a diagnostic system due to their restrictions such as the quality of the information. The fundamental idea in hybrid fault diagnosis is that some of these methods can complement each other resulting in more desirable diagnostic system. In other words, combination of these individual techniques can overcome the limitations of them [26, 100]. In the literature there are some methodologies that utilize the mathematical

model of a system and the approximation and adaption capability of the neural networks at the same time [99].

1.2.1.2 Desirable Characteristics of Fault Diagnosis Systems

Fault diagnostic can fulfill with various approaches, in order to make them comparable, it is useful to identify a set of desirable characteristics that a diagnostic system should possess. According to [69], the fault diagnosis system should posses ideally the following characteristics:

- Quick detection, isolation and identification

  The quick response of the diagnosis system in detection, isolation and identification of the process malfunctions is an important factor in order to avoid any consequences. However, the sensitivity of the fault diagnosis system to noise and disturbances in normal operation condition can result in frequent false alarms.

- Isolability

  Isolability is defined as the ability of the diagnosis system in classification of various failures in the monitored system. This ability of the classifiers in diagnosis system depends to a great extent on the process characteristics. The fault diagnosis system with high degree of isolability has problem with rejection of the modeling uncertainties.

- Robustness

  The ability of the diagnosis system to operate in the presence of the noise, disturbance and modeling uncertainties is called robustness. In other words, the performance of the robust fault diagnosis system degrades gracefully instead of failing totally and abruptly.

- Novelty identifiability

  It refers to the ability of the FDI system in making decisions about the monitored system that whether it operates in a normal or abnormal condition and if abnormal, whether the cause is a known fault, unknown fault or novel fault. According to this characteristic, it is expected that the fault diagnosis system can recognize the occurrence of the novel faults and not misclassify them into the other groups.

- Classification error estimate

  Providing a prior estimate on classification error that can occur in monitored system is an important practical requirement for the fault diagnosis system. Such error estimations would be useful to give the reliability level of the fault diagnosis system.

- Adaptability

  The monitored process could change due to structural changes, disturbances and changing in the environmental conditions, therefore the fault diagnosis system should be adaptable to these different changes.

- Explanation facility

  The fault diagnosis system not only should identify the source of the faults but also it should provide the information and explanation about the origination of them. Achieving this ability requires the capability of reasoning on causes and effects relationships in a process.

- Modeling requirements

  The modeling effort for development of the diagnosis classifier that capable to perform in real-time should be minimal as possible.

- Storage and computational requirements

  The implementation of the algorithms that are no too complex computationally and have the high capability of the information storing are the two competing requirements for a fault diagnosis system.

- Multiple fault identifiability

  The fault diagnosis system should be able to detect and isolate multiple faults, which is a difficult task due to the interacting nature of most faults.

1.2.2 Previous Work on Fault Diagnosis of AUVs

Underwater vehicles are liable to faults or failures through underwater tasks to prevent annulling the missions. In spite of the fact that researchers attempt to design a model to minimize the occurrence of the faults and failures, the probability of the fault occurrence exists. Identification that such situations do occur empowers the designers to minimize the effect of them. In the literature numerous applications of fault diagnosis are reported for aeronautical and aerospace systems, automotive and traffic systems, chemical processes, electrical and electronic systems, nuclear plants, power systems and transportation systems [113]. Recently a significant attention has been dedicated to fault diagnosis for unmanned underwater vehicles. The integration of fault-tolerant capabilities within the frameworks of the various control architectures for unmanned underwater vehicles is still an open problem.

In [111, 114] a model-based fault detection scheme was proposed for isolating actuators faults in the horizontal motions. The presented algorithm was based on a bank of Extended Kalman Filters (EKFs) that the outputs of them were checked in order to diagnose a performance of the system, which was different from dynamic model. Designing three EKFs simulates the behaviors of the two horizontal thrusters and horizontal motion. These behaviors were categorized in three types: I) Nominal

II) Left thruster fault II) Right thruster fault. In [115], the similar method was studied while instead of using EKFs a sliding-mode observer was applied. The effectiveness of this approach was validated experimentally.

The study in [104, 116] focused on the thruster failure detection through observing the motor current and the propeller's revolution rate. The non-linear nominal characteristic has been identified in experiment; therefore if the measured couple current propeller's rate was out of a specific bound, then a fault was occurred. Mapping of the i-o axis made it feasible to specify the possible cause and informed the remote human operator. Two types of thruster failures, thruster flooding and rotor failure, could be isolated as they fell in different axis regions.

The failure in a thruster was also considered in [107] by utilizing the hall-effect sensor mounting on each thruster. The controller and the thruster control matrix (TCM) computed the desired voltage, input, when the hall-effect sensor measured the output voltage. The difference between the measured and the predicated values for the voltage indicated that a fault occurred.

In [117], a fault manager subsystem was applied for an AUV, which was considered as a high-level fault detection approach. This method was based on layered control concepts that in the first step divided the mission into different phases (phase i.e. series of maneuvers between way points) and secondly when a failure occurred in each of the phases, it activated a related behavior.

In [118] a model-based fault detection scheme for thrusters and sensors was presented. It had been developed with respect to the identified model of the 6-thruster remotely operated vehicle (ROV) and it consisted of a bank of single-output Luenberger observers. Its effectiveness was verified by simulations.

In [119] a robust failure detection and isolation (RFDI) method was applied to the open loop pitch dynamic of an underwater vehicle for detecting the failure during

maneuvers and also during straight and level cruise. The proposed technique is sensitive to a fault occurrence while remaining robust to failure mode, noise and plant model uncertainties. This algorithm was the general model of likelihood ration test for failure detection and isolation in dynamic systems.

The authors in [120] proposed an intelligent decision making (IDM) to detect, isolate and control faults in control surfaces and sensors of an AUV. The proposed system was the combination of the learning capability of an ANN and the decision making ability of a fuzzy expert system (FES). The ANN was trained to detect the faults and the FES was utilized to suggest alternate measures to recover from the failure and to identify the stability of the system in the faulty situation.

In [121, 122], the Hotelling's T^2 statics and the principle component analysis (PCA) were applied to detect and isolate the faults in underwater vehicles. The stern plane jams and rudder jams were considered as the fault scenarios in this paper. The simulation showed that the presented technique was reliable in detecting and isolating the fault.

A control reconfiguration based on hierarchical FDI (HFDI) for rudder and sensor failures of an unmanned underwater vehicles (UUVs) was investigated in [123]. The authors developed a control reconfiguration technique with multiple sliding mode controller associated with each of the hypothesized failures modes. A probability-weighted average of all control signals was used to reconfigure the control signal. The results demonstrated that the proposed method compensated the steering track and heading angle properly.

The authors in [124] presented an observer-based fault diagnosis to detect, isolate and identify the actuator faults of the AUV. The support vector machine (SVM) was utilized as the diagnostic observer which was trained off line. In addition, for detecting and estimating the unknown actuator faults, a RBF network was embedded

into the observer. The effectiveness of the proposed method has been validated by performing simulation for the NPS AUV II vehicle (PHOENIX).

A novel thruster fault diagnosis and accommodation system (FDAS) for open-frame underwater vehicles was presented in [125]. The presented FDAS composed of two subsystems named as fault diagnosis subsystem (FDS) and fault accommodation subsystem (FAS). The FDS utilized fault detector units (FDUs) for monitoring the thruster states and reporting any internal or external faulty situations. The proposed FDUs were based on integration of self-organizing maps and fuzzy logic clustering methods. The FAS accommodated the faults and performed a proper control reallocation with respect to the information providing by the FDS. The weighted pseudo-inverse was used in the FAS to find a solution for control allocation problem. The performance of the proposed FDAS in various faulty situations was evaluated by a ROV simulator.

The authors in [126] presented a statistical method for detecting and isolating the faults in the actuator of an AUV. The Hotelling's T^2 statics and the partial PCA (PPCA) were utilized in this paper. In PPCA technique the data set was divided into some subsets and the PCA algorithm was applied on each subset.

The authors in [105] applied a model-based observer to generate residuals between real behavior of the system that was measured by sensors and the predicated one. In this approach a fuzzy interface system was used to isolate the source of fault when the residual is larger than a given threshold.

The authors in [127, 128] presented a fault tolerant control using qualification of contributing variables (QCV) for AUVs in the presence of stern faults (i.e. stern plane jams). In this paper the FDI unit was implemented by linear quadratic regulator (LQR) method which precalculated the gains for different faults and different speeds.

The authors in [129] presented a new fault diagnosis and accommodation system

for open-frame underwater vehicles. For the online FDI unit an improved credit assignment cerebellar model articulation controllers (ICMAC) neural network was utilized which has a faster training process in comparison with the traditional neural networks (i.e. MLP). In the fault accommodation unit a weighted pseudo-inverse based controller was applied.

Wang *et al.* presented a fault detection and identification scheme for sensors of AUV based on the improved PCA model [130]. This novel method was named as cumulative percent variance based on average eigenvalue that was applied to select the principal component scores. In this work the PCA, Hotelling's T^2 test and Q-statics were used.

The authors proposed a model-based thruster fault detection scheme for an AUV in [131]. Practically, a motor controller provides either, or both, electric current load and shaft angular velocity as feedback. Accordingly, for a vehicle equipped with servo motor based marine thrusters, the velocity and current feedback from the motor controllers can be used to derive two independent thrust approximations. The difference between the two models revealed the presence of fault conditions, and computed the error of the output thrust relative to the desired reference. Moreover, the faults were inherently isolated because each thruster had its dedicated motor controller with independently monitored feedback data.

A grey dynamic prediction (GDP) based sensor fault diagnosis for AUV was presented in [132]. The proposed method was applied to diagnose the faults in depth sensors of the AUV. The results showed that this method was fast and accurate in diagnosing the faults but it was not sensitive for slow faults.

Liang *et al.* proposed a least disturbance wavelet neural network to produce the dynamic model of an AUV [133]. The residuals were generated by comparing the outputs of the dynamical models and the real state values to detect the thruster

faults of the AUV.

The authors in [134] presented a fault diagnosis system for thruster of an AUV. In the proposed FDI system, a bank of nonlinear diagnostic observers were applied to generate the residuals. The generated residuals were the difference between the behavior of the reference model and the behavior of the AUV. The decision about the occurrence of the fault in the thruster was made by evaluating the residuals.

Corradini *et al.* presented an actuator fault-tolerant control (FTC) scheme for a class of nonlinear system which was applied for an underwater remotely operated vehicle (ROV) [135]. The proposed method consisted of three modules named as, detection, isolation and accommodation. The fault detection was performed by a nonlinear observer, while the isolation and the control was based on the sliding mode control (SMC) where each sliding surface was affected by a single thruster. Finally, when the faulty actuator was determined the control reconfiguration was done through utilizing the redundant healthy actuator. Similarly the authors proposed a failure tolerant robust control scheme for an actuator of the underwater ROV in [136]. In this scheme in the first step a reduced order observer has been designed to estimate the ROV velocities. Then the measured ROV positions and the estimated velocities has been used to develop the SMC algorithm and detect the thruster failures. In the next step, isolation was performed through exploiting the ROV structure and finally when the faulty thruster was identified, it was replaced by a healthy redundant actuator.

A quantitative/qualitative hybrid fault diagnosis method for simultaneous faults of thrusters and sensors of the AUV was proposed in [137]. The presented method was the combination of neural network technology with the dynamic trend analysis technology. In the first step of this method a fault detection observer model was applied to achieve the estimated state and the fault vector of AUV, then the dynamic trend analysis was proposed to locate the fault. The effectiveness of this method was

shown by experiment results on "Beaver" AUV.

Jianguo *et al.* presented a fault diagnosis scheme for AUV based on wavelet neural networks (WNN) [138]. The WNN was used to build the model of the AUV, and the residuals were generated by comparing the sensor outputs and the WNN outputs. The fault diagnosis was accomplished by analyzing the residuals based on the rules that were achieved through the simulation.

The authors in [139] presented a new fault diagnosis and accommodation technique for unmanned underwater vehicles. The proposed fault identification subsystem was named as credit assignment-based fuzzy cerebellar model articulation controllers (FCA-CMAC) neural network which was applied for identification of multi-uncertain abrupt thruster faults of the unmanned underwater vehicles. A reconstruction algorithm based on weighted pseudo-inverse was utilized for finding the possible solution of the allocation problem.

The authors in [140] proposed an H_∞ formulation of the simultaneous fault detection and control (SFDC) problem for an AUV. In this method dynamic observer detectors and state feed-back controller for linear continuous-time model of AUVs were used. An LMI-based (linear matrix inequality) solution to the SFDC was presented in this work which stabilized the closed-loop system, and ensured achieving control and fault detection.

A fault diagnosis method based on grey correlation analysis for sensor of AUV was presented in [141]. The sensor fault was detected and isolated through analyzing correlation coefficient between fault pattern vector and testing vector. The proposed technique had some advantages such as, it required no system model. The simulation results proved that the sensor faults were diagnosed fast and accurately by using this method. The authors in [142] presented a sensor fault diagnosis method based on second-order Taylor series dynamic prediction (SOTSDP) for AUV. Generally,

the SOTSDP is utilized to build the model of the system when the information is incomplete. The simulation results showed that the proposed method can diagnose the sensor faults fast and precisely but it was not sensitive to slow change faults. This technique improved the prediction accuracy in comparison with grey dynamic prediction approach.

The authors in [143] developed an H_∞/H_- formulation of the simultaneous fault detection, isolation and tracking (SFDIT) problem for linear continuous-time systems using a dynamic observer. In the proposed method, each element of the residual vector was sensitive to a specific fault, for this reason the occurrence of simultaneous faults was manageable. An LMI-based approach to the SFDIT was proposed in this work which stabilized the closed-loop system, and ensured achieving control and fault detection. This method was applied to the linearized longitudinal mode of the Subzero II AUV which the results illustrated its effectiveness.

An AUV multi-fault mode classifier was established in [144] by using fuzzy weighted support vector domain description (FWSVDD) method based on positive and negative class samples. In this paper, the multi-fault mode classification technique based on a hierarchical strategy was presented to improve the training speed and the accuracy of the fault diagnosis. In the proposed method fault contain detection surface was added for each thruster and sensor to isolate fault components during fault diagnosis. In addition, a relative judgment method was developed considering the class ownership and judgment problem for fuzzy sample points that are in the overlapping area of hyper spheres or that do not belong to any hyper-sphere in the process of fault classification. The effectiveness of the presented multi-fault diagnosis approach was validated through water tank experiments with an experimental AUV prototype.

1.2.3 Dynamic Neural Networks

In recent years, dynamic neural networks (DNNs) have been significantly applied for modeling nonlinear systems. The first type of a dynamic neural network applies the internal feedback in its neurons for modeling the dynamical behavior of the nonlinear system. This network utilizes the FIR (Finite Impulse Response) or IIR (Infinite Impulse Response) filters in its neurons structure along with the activation function [15 - 22].

The second type of dynamic neural networks is called time-delay neural networks, which adds a delay associated with the weights of the network in order to modify the static neurons. This modification lead to a network that is capable of generating dynamical behavior of a nonlinear system [23 - 25].

Another form of the dynamic neural networks is developed with applying the external feedbacks to the structure of the static neural network [26, 27]. In [27], the tapped delay lines were utilized along a static network as the external feedbacks to present the dynamical behavior. The other method in external feedback was proposed in [26], where the extensive feedback between the neurons of different layers generates the dynamics in the structure of the neural network. A fault diagnosis system for the reaction wheel of the satellites based on a recurrent Elman network was presented in [28]. Similarly, in [29] a fault diagnosis scheme based on recurrent neural network was applied for detecting and isolating the fault in the actuator and thruster of the satellite.

A neuro-dynamic structure was developed in [15], which utilized an internal feedback between the input and output of the neurons. This method employed an IIR filter that lead to local memory characteristics for the neural network. In this approach the filter was placed before the activation function while in [24] the filter was applied after the activation function of the neuron that result in a simpler relation between

the input and output of the network. In the other approach in [23], a delayed sample of the output was considered as an input to the neural network for modeling the dynamical systems.

As mentioned earlier the dynamic neural networks are capable of learning the dynamics of the nonlinear systems, thus recently they have applied in fault diagnosis systems. In [19], a dynamic neural model was proposed in order to detect actuator faults in the attitude control subsystem of a satellite. The authors in [22] presented a dynamic neural network for fault detection of real sugar evaporation processes. A simulation perturbation stochastic approximation was developed for updating the network parameters. In addition, for isolating the fault in the system a multiple model including healthy and faulty modes was applied. The other fault diagnosis system based on dynamic neural networks was developed for thrusters in the formation flying of satellites [18]. The network had a series-parallel architecture, which means it utilized a delayed feedback of the actual output for the training phase and a feedback of the network for the recall phase. A dynamic neural network was developed in [30] for process modeling and fault diagnosis of a two-tank process. A modified form of dynamic neural networks for identification of nonlinear systems was presented in [24], where the application of it for fault detection of the aircraft jet engines have been shown in [17]. In [17], for each parameter of the engine a separate neural network was developed and for the residual generation the series-parallel method was applied.

Time delay neural network (TDNN) is another structure for the dynamic neural networks that primarily was developed by Wiabel [25] for phoneme recognition. This network contains a delay along with each weight to provide a dynamic neuron. The delays in TDNN are applied to all the layers of the network, while in taped delay network the delays only are considered for the input layer. These two networks are utilized for recognition of spatio-temporal patterns and the back propagation

algorithm is used for training them.

In order to improve the performance of the conventional time delay neural networks that utilizes the fixed delays during the training phase, the adaptive time delay neural network (ATDNN) was presented in [23]. In the ATDNN the weights and the delays were updated in the training mode. The proposed neural networks in [23] have been applied for identification of four different classes of nonlinear systems. This method was developed for identification of a two-link flexible robot [31]. The development of the TDNN can be seen in fault diagnosis systems of automobile transmissions gears which was combined with radial basis function (RBF) network [32] and also damage detection of railway bridges [33]. The TDNN was applied in other areas such as modeling industrial systems, speech recognition and image sequence analysis [34].

The authors in [166] presented a fault detection and isolation scheme based on neural networks in order to detect and isolate the component faults in a dual spool turbo fan engine. In the proposed method, multiple dynamic neural networks (DNNs) were used where each of the networks corresponded to different operation modes of the healthy and faulty engine conditions. The performance of this method was investigated through various faulty scenarios.

1.2.4 Formation Control of AUVs

During the last two decades, the "Formation Control" is received lots of attentions from researchers, since it can reduce the system cost and increase the robustness and efficiency of the system. Formation control is considered as one of the most fundamental problems in control of Multi-AUVs, which means each AUV keeps a desired formation configuration while performing the mission. The formation of small AUVs is capable of performing the tasks of a large AUV, since the tasks are distributed

among them.

Formation control strategies are divided into two main category namely as, centralized and decentralized. A centralized coordination scheme depends on the assumption that each vehicle in the team has the ability to communicate to a central vehicle. Therefore, the coordination of all vehicles can be achieved if the central command and control has the capability to process the information and inform each individual vehicle the desired localization or command frequently enough. In decentralized coordination scheme a group of vehicles reaches the desired group behavior via local interaction [178]. In this scheme there is no central vehicle and the feedback is only the relative states of each vehicle to its neighbor vehicles.

The centralized approach requires the stable communication among the vehicles but is vulnerable because of inevitable disturbances, limited bandwidth, and unreliable communication channels [178]. Another disadvantage of the centralized scheme is the failure of the overall system due to its single point of failure [164]. The centralized formation control is considered as a good strategy for the small teams which are implemented with a single computer and a single sensor to monitor and control the entire team. However, for the teams with a large number of agents that require greater computational capacity and a large communication bandwidth, utilizing the decentralized formation control strategy is preferred because the computational load is equally distribute through the formation. Another advantage of the decentralized method in comparison with the centralized approach is its robustness to a single-point failure [177] .

In addition, the formation architectures are roughly categorized as leader-follower, behavioral and virtual structure. These main categories are fully explained in following.

- Leader-Follower approach

  In this method one of the agents is considered as a leader and the remaining agents is designated as followers. The leader tracks the pre-specified trajectory while the others should track the states that transforms by the leader to them. The advantage of applying this technique is that the group behavior can be determined through specifying the leader's behavior. Although, it can result in dependency of the formation failure on a single points and nonexistence of the obvious feedback to the formation.

- Behavioral approach

  In this method the various desired behaviors are determined for each of the agents that leads to the controlling each of the agents based on the weighted average of the control for each behavior. The behaviors can be named as: formation keeping, goal seeking, collision avoidance and obstacle avoidance. The pros. and cons. of this approach are listed below: Advantages: 1) The implementation is decentralized that can compensate the problem of centralized approach, which is the failure of the formation depends on one agent. 2) As each agent reacts to the position of its neighbors, the explicit feedback to the formation is existed. 3) Deriving control strategies naturally while the agents have multiple competing objectives. Disadvantages: 1) An explicit group behavior cannot be defined in this technique. 2) At the point of mathematics it is difficult to analyze and warrant the stability of the group.

- Virtual Structure

  In this method the formation acts as a single structure, which is called "Virtual Structure". It is considered as a rigid body with a specified direction and orientation that should keep the geometric pattern among agents. The main

advantage of this methodology is that describing the coordination behavior of the group is easy. The disadvantage of it is that the virtual structure results in the limitation of the area that this technique can be used.

The proposed framework in [35] for keeping the fixed geometrical formation while performing the mission in multiple AUVs formation used the leader-follower algorithm. In the presented scheme each follower stayed in the predetermined position by use of measurement of its inertial position and the leader vehicle position, which obtained by using acoustic Long Base Line (LBL). In addition, if the follower cannot complete the mission it can be done in two ways: 1) Each follower use its independent navigation that is gained through inertial LBL information or 2) Substituting one of the followers as a new leader.

In [36] the authors presented a leader-follower formation control of underactuated AUVs that a virtual vehicle was designed based on the position measurements from the leader. The trajectory of the virtual vehicle converged to the reference trajectory of the follower. The Lyapunov and backstepping synthesis were applied as a position tracking control for the followers to track the virtual vehicle. A variable structure control law for formation tracking of multiple AUVs was applied in [37]. The AUVs tracked along the desired trajectory by minimizing the cross track error that was calculated from the line-of-sight angle and the leader-follower formation control was established, which was based on the feedback linearization.

In [38] a leader-follower formation control for multiple autonomous underwater vehicles in spatial motion was presented while incorporating the uncertainties of hydrodynamic parameters into the controller design was considered. In this work the leader kept its desired trajectory by applying adaptive inverse dynamics algorithm and it periodically broadcasted its position to the followers, since they maintained in a fixed configuration with use of this information and applying the adaptive control

algorithm.

The study in [39] presented an adaptive distributed control for a group AUVs, which considered the hydrodynamic parameters uncertainties of the vehicle and the communication constraints in underwater environment. In this work the inter-vehicle communication in formation control was eliminated by use of a local controller for each AUV that produced control signal based on the measured formation functions, which broadcasted to the whole system. In this work, an adaptive inverse dynamics algorithm was applied to deal with uncertainties in the hydrodynamic parameters.

The authors in [40] developed a formation control scheme for large-scale multiple autonomous underwater vehicles, which was the combination of the leader-follower and the virtual structure. In this method the AUVs were categorized into some clusters based on their relative position in space. The leaders in all clusters track their desired trajectories according to the virtual structure and back stepping control algorithm, and the followers of all clusters followed their corresponding leader as the desired tracking trajectory. In addition, this hierarchical virtual-leader-follower structure was developed to overcome the constraints of the acoustic communication that the AUVs deal with them.

The formation control of multiple AUVs presenting in [41] was a control method under fixed interconnection topology, which means the AUV exchange information with the fixed AUVs (neighbors) during the mission. This method included two parts: 1) Path Following and 2) Formation Keeping. In the formation keeping the formation reference point (FRP) was considered that follows the given parameterized path with a desired formation speed and each AUV followed the point with a predetermined distance to FRP. The authors designed an adaptive sliding variable structure control law for AUVs, since there was parameter uncertainty and environmental disturbances.

1.2.5 Genetic Algorithms and Neural Networks

Genetic algorithms (GAs) are considered as the search techniques that are very efficient in optimization problems. The GAs search the enormous and complicated spaces in order to find the global optima. Since the neural network weight selection problem includes a search space, the GA method can be a possible solution for finding the weights.

According to the literature, one of the main challenges in neural networks is spreading more local optima over the space when the number of examples and complexity of the network increases. The gradient-based approaches deal with some difficulties to pass the local optima and find the global optima. Thus, the GAs can be a possible choice when the complexity of the search space increases and the gradient-based techniques such as back-propagation are not able to find the global optima.

Another advantage of the GAs is their generality which means by applying minor changes into the algorithm, it can be utilized to train various types of networks. They can choose the weights for the networks with the closed paths in their topologies such as recurrent networks. They can be used to train networks with different kind of transfer functions like Gaussians and step transfer functions that are discontinuous and cannot be trained by the traditional gradient-based methods. In addition, genetic algorithms can optimize any networks with various combinations of weights, biases, topologies, and transfer functions. Genetic algorithms have been used together with neural networks in different modes that are expressed as follows [42]:

- Applying genetic algorithms in order to process neural network data, generally GAs are used to do feature selection.

- Selecting the topology of the neural networks via using genetic algorithms. In

other words, the GA determines the number of the hidden units that is required in the network and how the nodes should be connected.

- Selecting the weights of the neural networks by means of genetic algorithms.
- Using genetic algorithms in order to learn the neural network learning algorithm.

The evolution of weights in the networks with the static structure can be considered as a replacement of the traditional training algorithms. The traditional gradient-descent methods such as back-propagation (BP) [43], can be trapped in the local minima and they require that the activation function be differentiable, while the evolutionary approaches can overcome these difficulties. Therefore, instead of adapting the weights locally, evolutionary algorithms (EAs) are applied to evolve the weights with respect to the fitness of the whole network.

Different approaches have been presented in the literature for optimizing the neural network weights that utilize genetic algorithms. The methods in the literatures are explained fully in following.

In [44] the biases and the weights of the neural network were encoded with real numbers and were initialized based on a random probability distribution function. Different type of genetic operators named as, mutations, crossovers and gradients were applied in this study. Mutation employed perturbation to some of the entries in the chromosomes at random for generating its offspring. Crossover generated the two offsprings that containing the genetic material from their parents. The gradient operator determined the child of the selected individual through adding the gradient value regarding the evaluation function to its entries. The main objective of this work was selecting a set of good operators for the problem based on their performance in different situations. The results proved that the evolutionary training is faster that the BP approach.

Whitley *et al.* proposed a genetic approach based on binary encoding of the weights that is called GENITOR [45]. In the proposed algorithm each string was evaluated and the population was arranged with respect to ranking strings based on their evaluations. The random selection was applied to select parents for recombination. Only one of the generated offsprings by recombination of the selected parents remained and it was replaced in the population based on its ranked string. Whitley *et al.* compared their algorithm with the Montana and Davis method in [46]. The main difference of these two techniques was the implementation of encoding. Whitley *et al.* presented a modified version of the GENITOR in [47, 48] where the real value encoding, a small population and high mutation rates were considered in the algorithm. This new method provided better results in comparison with the back-propagation technique.

One of the difficulties in applying GAs is the premature convergence that occurs when the population looses its diversity [49]. Yang *et al.* proposed a genetic algorithm based on evolutionary stable strategy (ESSGA) in [50] to overcome these problems and manage convergence speed and the population diversity during the evolution procedure. This objective was achieved through applying a mutation operator in conjunction with a controller stable factor. Smith in [51] presented the ESS where a controller was determined to maintain the quantity of preponderant individuals to stable quantity of the population dimension in each generation. The authors showed that by using this approach the premature convergence can be prevented and there was no increase in the running time. In other words, restricting the over reproduction of preponderant individuals result in a diverse population and a large search space. This method applied for the XOR-problem and it increased the speed and the accuracy.

The authors in [52] studied an evolutionary neural network for gait analysis of

human motion. In this work the real numbers were selected for the weight evolution of the network. The multi-point crossover and the mutations were the operators that were implemented. In mutation procedure the mutation rate and the step size were varied. The multi point crossover was carried out on the binary chromosomes and they were decoded to generate the offsprings with real number of chromosomes. The results proved that the mutation and crossover provided a system with the better classification ability in comparison with the multi layer perceptron (MLP) network. In addition, the simulation results showed that the BP approach was over-fitting the training data and had more error comparing to the evolutionary method.

Seiffert in [53] presented a genetic algorithm in the training phase of the MLPs which can be completely be replaced by the traditional gradient descent approach. In this work the architecture of the neural network was determined in advance and stayed fixed after the initialization. The chromosomes of the genetic algorithms contained the weight values and did not include any information about the topology or the structure of the network. Selection, reproduction and mutations were the operators that were applied in this work. The author implemented this method on various problems to compare the results with BP solutions. The outcome proved that in the complex problem the BP algorithm failed and the GA can be considered as an alternative.

A genetic inheritance operator named as, short-term reproduction expectancy (STRE) was applied in [54] to determine the weights of an EANN that is a multi-layer feedforward network with a predetermined fixed topology. In the STRE method a set of best fit parent chromosomes were selected and their life expectancy is extended for at least one generation whereas the remaining chromosomes were the best fit offspring. In this algorithm no mutation operator was applied and a two-point crossover is used for the reproduction. The chromosomes were coded in decimal with values between

0 and 9. The inverse of the root mean squared error in the training phase was chosen as the fitness value for the chromosomes. This algorithm was applied for prediction of uplift capacity in the field of geotechnical engineering. The result showed that the STRE converged faster, decreased the learning error and provided better values compared to other approaches.

In [55], a genetic algorithm was applied to optimize the weights of a three-layer back propagation network to minimize the error. This neural network was used to anticipate punch radius regarding to the air-bending of sheet metal experiments. The presented algorithm contained two steps: in the first stage the GA was applied in order to find the optimal connection weights and the threshold for the network, following the learning rules of the back-propagation was used for adjusting the final weights. The simulation results showed the effectiveness of the proposed algorithm in accurate prediction of punch radius with less time and fast convergence.

The authors in [56] presented a novel real-coded genetic approach called "RCGA-ELM" in order to find the optimal number of hidden neurons, input weights and bias values. Selecting these optimal numbers is critical for performance of the ELM network. Two new genetic operators were applied in this work namely as "network based operator" and "weight based operator". The network based operator for the crossover used heuristic operation to generate the weights of the L^{th} hidden neuron and for the mutation operator added or deleted a hidden neuron in the selected parent. The weight based operator used an averaging operation to generate the values of the selected connections in the children for crossover and randomly selected M weight values in mutation. The results showed that the proposed method was effective.

In [57] the hybrid taguchi-genetic algorithm (HTGA) was implemented to find the optimal parameters of the ANN including weights of links and biases. This HTGA based ANN have been applied to predict the transfusions requirements of the RD-PC

and the SD-PC on AML patients. The taguchi method applied two major tools named as, signal-to-noise ration and orthogonal arrays to select better genes for crossover. The experimental results proved the prediction accuracy of this proposed algorithm.

The combination of the adaptive genetic algorithm and modified Newton method was proposed in [58] to train a feedforward neural network. The genetic algorithm and its operators searched for the initial weights and bias, whereas the Newton algorithm increased the performance of the network. The effectiveness of this method was shown by experiments of system identification and suppression.

Karegowda *et al.* proposed a hybrid model that combined genetic algorithm and BP where the GA was applied to initialize and optimize the connection weights of neural network [59]. In the presented method the chromosomes were encoded through real value approach, the fitness value was calculated through mean squared error and the individuals with the best fit were replaced the worst fit individuals. A new type of crossover called mixed crossover was implemented where for the 60% of the generation the multipoint crossover were used, following by next 20% of generation utilized two point crossover and the remained population used the one point crossover. In the last stage of the algorithm the mutation was applied. This algorithm had been experimented for classification of the PIMA data set and the experimental results showed its accuracy in classification comparing to the BP network.

As in the RBF neural networks accomplishing the learning ability and deciding about the network architecture is hard the authors in [60] proposed an algorithm where the RBF neural network learning was optimized based on the genetic algorithm. This method implemented hybrid encoding which encoded the network through binary encoding and encoded the weights by real encoding. The network architecture was self-adapted while the weights of the network were learned. Finally the network

was adjusted by pseudo inverse method or least mean square algorithm. The results showed that by applying this method the network had a better architecture and classification ability and the time for constructing the network decreased.

A method was proposed in [61] for optimizing the BP algorithm by using genetic algorithm for training the BP network for overcoming disadvantages of the BP named as, getting stuck in local minima and not having a good rate of convergence. This algorithm was validated by the UCI data set which proved that the combination of the GA and BP had a better generalization ability and had a good stabilization performance.

Elveren and Yumuşak in [62] proposed a genetic algorithm for training a multilayer neural network with two hidden layers. This method was applied for classification of the tuberculosis data set and the experimental results proved that it had a better performance in comparison with a traditional multi layer neural network (MLNN).

Back propagation and genetic algorithm results in adjusting the feedforward neural networks weights were compared in [63]. The comparison showed that the back propagation method had a faster speed while it had the disadvantage of overtraining where the GA did not. In addition, the required central processing unit (CPU) time was less in back propagation comparing to GA.

A hybrid genetic algorithm-neural network (GA-ANN) is presented in [64] which combined the ability of the gradient-based back propagation in local searching and the ability of the genetic algorithm in global searching. The genetic algorithm was applied to set the initial weights of the gradient decent method. The developed ANN learning process contained two steps: in the first step the GA was employed to find the approximate optimal weights and threshold for the network, then the back propagation was applied to adjust the final weight values. The experimental results

indicated that the presented method had a better performance in comparison with back propagation.

Ahmadi *et al.* presented a feedforward artificial neural network where the optimization of the network was done by means of hybrid genetic algorithm and particle swarm optimization method [65]. The proposed genetic algorithm was applied for initial weighting the parameters of the neural network. This hybrid algorithm was applied to anticipate the permeability distribution in an oil field case. The experimental results indicated that the proposed algorithm was more reliable in comparison with traditional methods; it was a strong optimization tool in problems when the objective function has lots of local minima; it provided a precise prediction of the permeability and it had lower mean squared error (MSE).

The authors in [66] studied the effectiveness of the evolutionary training of the feedforward neural network in cancer detection and prediction of its recurrence. The GA was particularly implemented to optimize the MLP weights. The encoding approach was based on the real value encoding. Five different crossovers were implemented separately named as, total arithmetic recombination operator, blend crossover, wright's heuristic crossover, linear BGA crossover and uniform crossover; the effectiveness of each crossover approach compared to others. The experimental showed that the presented algorithm provided the optimal weights and had the ability of using different data sets with keeping the accuracy comparing to other methods.

The authors in [67] presented a wavelet neural network (WNN) which was optimized using genetic algorithm to predict the air traffic flow. This presented method overcame the difficulties of the local minima and the oscillation effect of the wavelet neural network that uses only the traditional gradient descent method. The experimental results proved that the proposed technique had a high precision feature and can find the globally optimal parameters of the network.

Kalderstam *et al.* in [68] developed a prognostic model by applying the ANNs which were trained with genetic algorithm. The results of the proposed algorithm compared with Cox model showed that the ANN model has a greater performance on nonlinear data.

The authors in [168], presented an adaptive time delay neural network training based on parallel genetic algorithm in order to predict time-series. In the proposed training method the main population was divided into sub-populations which evolved separately for a certain number of generation and after a certain time a number of individuals is distributed between the sub-populations. The results indicated the efficiency of the proposed method.

1.3 Objective of the Thesis

The main objective of this thesis is to present a fault detection, isolation and identification (FDII) scheme for thruster in formation of multiple AUVs. In order to accomplish this goal, the dynamics of the AUV, thrusters and the formation control architectures are presented and described fully. Two different fault detection schemes based on dynamic neural networks are presented, named as agent-level fault detection (ALFD) and formation-level fault detection (FLFD). The mentioned approaches are developed, described and valued for formation of AUVs by performing extensive simulation and presenting their results. For the fault isolation and identification schemes, a multiple neural network-based method is proposed. Two different fault isolation and identification schemes are presented namely as, agent-level and formation-level. In the proposed schemes two multi layer perceptron neural networks (MLPNNs) are employed to classify the type of the occurred fault and indicate its severity respectively. The performance of the proposed agent-level and formation-level fault isolation and identification schemes are evaluated through simulation.

1.4 Contributions of the Thesis

The main contributions of this thesis are presented as follows:

- A novel fault detection scheme for thruster of the AUV in a formation is proposed by employing dynamic neural networks (DNNs) which are trained with genetic algorithm (GA). The fault detection scheme is developed in two different approaches namely as, agent-level fault detection (ALFD) and formation-level fault detection (FLFD) scheme. The proposed schemes are capable of successfully detecting commonly occurring thruster faults in an AUV. The results that are obtained through simulations indicate that both agent-level fault detection and formation-level fault detection schemes can detect medium severity and high severity faults in the thruster, while the formation-level fault detection is capable of detecting low severity faults with a high level of accuracy and precision as well. In addition, the results obtained through a large number of simulation scenarios demonstrate a high level of accuracy and precision. The capability of combining the continues genetic algorithm and neural networks schemes in the fault diagnosis problem is successfully demonstrated.

- The capabilities and advantages of our proposed training algorithm (i.e. genetic algorithm) for dynamic neural networks is compared with the extended dynamic back-propagation (EDBP) algorithm. The results indicates that our proposed training method converge faster and has a less root mean squared error in comparison with the EDBP algorithm.

- A fault isolation and identification scheme based on multiple neural networks is proposed to indicate the type of the fault in the thruster of the AUVs and determining the severity of the occurred fault. The fault isolation and identification

schemes are developed through two different approaches namely as, agent-level and formation-level. Two MLPNNs are employed in the fault isolation and identification schemes respectively to categorize the faults into thruster blocking, flooded thruster and loss of effectiveness in rotor and classify their severity into low, medium and high. The performance of the proposed approaches are investigated through various faulty scenarios and extensive simulations. The confusion matrix analysis indicate that the formation-level fault isolation and identification schemes have a better performance comparing to agent-level and it is capable of isolating and identifying the faults with acceptable accuracy and precision.

1.5 Outline of the Thesis

The organization of this thesis is expresses as following:

- In Chapter 2, firstly the model and the architecture of the dynamic neural network (DNN) is explained. Afterward the extended dynamic back-propagation (EDBP) which is a training algorithm for DNN is described. Secondly the evolutionary computation, evolutionary algorithms and its principles are presented fully. Afterward the genetic algorithm and its operators are explained in details. Thirdly, the model of an AUV is fully explained, which includes the kinematics and nonlinear dynamics. In addition, the concept of AUVs formation and the controller designs for a single AUV and formation of multiple AUVs are described in details. Finally, the mathematical model of the propulsion system of the AUV and the common faults in AUVs are explained.

- In Chapter 3, the agent-level and formation-level fault detection schemes in the formation of multiple AUVs are proposed and developed. Different fault

scenarios for thruster are considered and simulated and the accuracy and the precision of the proposed fault detection schemes are evaluated and compared to each other. The results corresponding to the training phase of the dynamic neural network (DNN) based on extended back-propagation (EDBP) and genetic algorithm (GA) are compared.

- In Chapter 4, the agent-level and formation-level fault isolation and identification schemes based on multiple neural networks are proposed and developed. In the proposed fault isolation and identification schemes, at first a MLPNN is applied to indicate the type of the occurred fault and secondly, a MLPNN is used to determine its severity respectively. The generated residuals from the agent-level and formation-level are processed and are used as the inputs for these networks. In order to evaluate the performance of these schemes different faulty scenarios are considered.

- In Chapter 5, the conclusions and the contributions of this thesis are provided and explained. Finally, the suggestions for future work are presented.

Chapter 2

Background Information

2.1 Neural Networks (NNs)

Neural networks are considered as nonlinear statistical models [145] that are inspired by the biological nervous system. Several references exist that present the neural networks with combination of neuron and synaptic connections which have the capability of transmitting data through multiple layers [146 - 147]. These networks are capable of solving various problems such as pattern recognition and classification. One of the important features of the artificial neural networks (ANNs) is their adaptive nature that makes them capable of learning through examples. This feature is beneficial in problem solving. In addition, ANNs constantly improve their performance through the learning process.

In ANN design several parameters should be set that can be affect the finding solution procedure. The number of layers and nodes that are part of the network architecture and the connection weights are some of these parameters. Moreover, the training data set and testing data set are considered as the critical factors in finding solution process.

The back-propagation (BP) method is the common technique that is used in

training of the neural networks to adapt weights. The BP is based on the gradient descent algorithm which can get stuck in local minimum and not able to find the global minimum when the error function is multimodal or non-differentiable. In addition, the sensitivity of the BP to the initial conditions can make it slow. Therefore, the evolutionary neural networks can be considered as an alternative to this traditional method.

2.1.1 Dynamic Neural Network (DNN) Model

The proposed neuro-dynamic structure in this thesis is based on the Ayoubi model [15], which is named as dynamic neuron model (DNM). Adding the internal dynamics to the structure of the dynamic neuron model makes the neuron's activity dependent on its internal states. This network contains dynamic neurons in its structure that these dynamic neurons can be produced by adding an infinite impulse response (IIR) filter in the structure of the standard static perceptron. The structure of a dynamic neuron model is depicted in Figure 2.1.

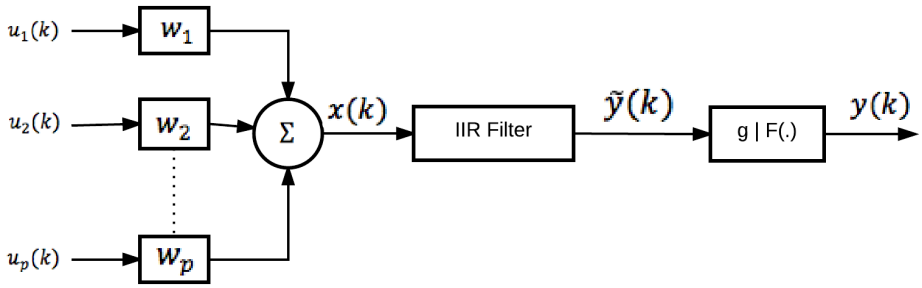


Figure 2.1: General structure of the DNM with p inputs [20].

According to the Figure 2.1, in the first stage of the dynamic neuron structure

the weighted sum of the inputs is calculated through the following equation:

$$x(k) = w^T u(k) = \sum_{p=1}^{P} w_p u_p(k) \tag{2.1.1}$$

where $w = [w_1, w_2, \cdots, w_p]^T$ is the weight vector and $u = [u_1(k), u_2(k), \cdots, u_p(k)]^T$ denotes the input vector.

The output of this step is passed through the IIR filter. The transfer function and the output of the n^{th} order filter are expressed as follows:

$$\widetilde{y}(k) = \sum_{i=1}^{n} b_i x(k-i) - \sum_{i=1}^{n} a_i \widetilde{y}(k-i) \tag{2.1.2}$$

$$H(q^{-1}) = \frac{b_0 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_n q^{-n}}{1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_n q^{-n}} \tag{2.1.3}$$

where n denotes the order of filter, q is the time shift operator, $x(k)$ denotes the filter input, $\widetilde{y}(k)$ denotes the filter output, $a = [a_1, a_2, \cdots, a_n]^T$ and $b = [b_1, b_2, \cdots, b_n]^T$ are feedback weight vector and feedforward weight vector respectively. As a result, the neuron output is stated as:

$$y(k) = F(g.\widetilde{y}(k)) \tag{2.1.4}$$

where g is the slope parameter of the activation function and $F(.)$ is the nonlinear activation function.

2.1.2 Dynamic Neural Network Architecture

Figure 2.2 illustrates an L-layered network that uses the dynamic neuron in its structure. A differentiable activation function, $F(.)$, describes the dynamic neurons.

In Figure 2.2 N_l is the number of neurons in the $l-th$ layer, $O_n^l(k)$ denotes the

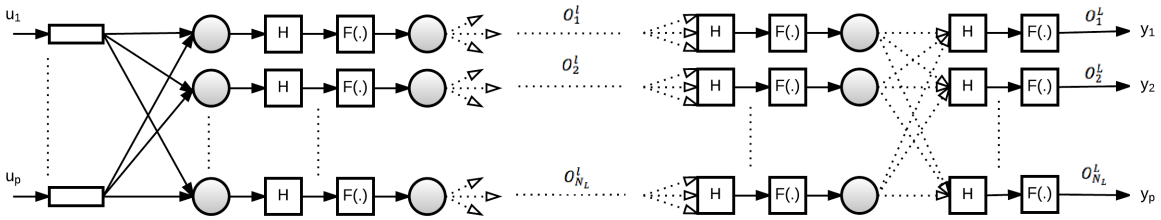


Figure 2.2: Dynamic neural network architecture

output of the $n-th$ neuron of the $l-th$ layer, and $u_p^l(k)$ denotes the input of the $l-th$ layer, generated from the $p-th$ neuron of the previous layer at discrete times k ($l = 1, \cdots, L; n = 1, \cdots, N_l$). The output of the $n-th$ neuron in the $l-th$ layer is defined as follows [30]:

$$O_n^l(k) = F[g_n^l(\sum_{d=0}^{D} b_{dn}^l \sum_{p=1}^{N_{l-1}} w_{np}^l u_p^l(k-d) - \sum_{d=1}^{D} a_{dn}^l \widetilde{y}_n^l(k-d))] \tag{2.1.5}$$

In equation (2.1.5), $w = \left[w_{np}^l\right]$ is the weight matrix, $a = \left[a_{dn}^l\right]$ and $b = \left[b_{dn}^l\right]$ are the feedback and feedforward filter parameters matrices respectively, $g = \left[g_n^l\right]$ is the slope parameter matrix and D denotes the order of the filter. ($l = 1, \cdots, L$; $n = 1, \cdots, N_L$; $d = 1, \cdots, D$)

It can be noticed from the equation (2.1.5) that the network outputs depend on the past outputs $\widetilde{y}(k-1), \widetilde{y}(k-2), \cdots, \widetilde{y}(k-n)$. Since the activation function, $F(.)$, is an invertible function (e.g. Tangent hyperbolic), therefore the network outputs will also depend on past outputs $y(k-1), y(k-2), \cdots, y(k-n)$. Thus the last layer outputs is expressed in the equation (2.1.6).

$$O_n^L(k) = \Gamma[y(k-1), \cdots, y(k-m_s), u(k), u(k-1), \cdots, u(k-n_s)] \tag{2.1.6}$$

where $\Gamma(.)$ is a nonlinear function presenting the overall network map. This equation shows that the network outputs are nonlinear functions of the inputs and their delays as well as the previous output samples.

2.1.3 Extended Dynamic Back-Propagation (EDBP) Algorithm

The main objective in both static and dynamic neural network is determining an algorithm to adjust the parameters of the network. The extended dynamic back-propagation (EDBP) algorithm which is the modified form of the static back-propagation algorithm is applied to adjust the parameters of the dynamic neural network [30]. In the learning process all the unknown network parameters are identified and adjusted by using the training set of input-output pairs. The back-propagation error method is applied extensively for training the static networks. The objective of the EDBP algorithm is to modify all the parameters of the dynamical neural network vector, $\gamma = [w, A, B, g]$, to minimize the performance index J which is expressed as follows:

$$J = \frac{1}{2}\sum_{i=1}^{N}(e_i(k))^2 = \frac{1}{2}\sum_{i=1}^{N}(y_i^d(k) - y_i(k))^2 \tag{2.1.7}$$

where $e_i(k)$ denotes the error of the i^{th} output which is the difference between the desired response $y_i^d(k)$ and the actual response $y_i(k)$, and N denotes the number of outputs.

Based on the EDBP algorithm the parameters related to the $n-th$ neuron of the $l-th$ layer are adjusted based on the following rule:

$$\gamma_n^l(k+1) = \gamma_n^l(k) + \eta\delta_n^l(k)S_{\lambda n}^l \tag{2.1.8}$$

where γ is the unknown generalized parameter vector, η denotes the learning rate,

S represents the sensitivity function for the elements of the parameter vector γ, and δ is the generalized output error which is defined for the output layer and hidden layer in the following equations:

- Hidden layers generalized output error

$$\delta_n^l = \sum_{z=1}^{N_{l+1}} (\delta_n^{l+1}(k) g_z^{l+1} b_{0z}^{l+1} w_{zn}^{l+1}) F'(\widetilde{y}_{1n}^l) \tag{2.1.9}$$

- Output layer generalized output error

$$\delta_n^L = e_n(k) F'(\widetilde{y}_{1n}^L) \tag{2.1.10}$$

The sensitivity function is given as below:

- Sensitivity with respect to the weight parameters

$$S_{w_{pn}}^l(k) = g_n^l [\sum_{i=0}^{m} b_{in}^l u_p^l(k-i) - \sum_{i=1}^{m} a_{in}^l S_{w_{pn}}^l(k-i)] \tag{2.1.11}$$

- Sensitivity with respect to the feedback parameters

$$S_{a_{in}}^l(k) = -g_n^l \widetilde{y}_n^k(k-i) \tag{2.1.12}$$

- Sensitivity with respect to the feed-forward parameters

$$S_{b_{in}}^l(k) = g_n^l x_n^l(k-i) \tag{2.1.13}$$

- Sensitivity with respect to the slope parameters

$$S_{g_n}^l(k) = \widetilde{y}_n^l(k) \tag{2.1.14}$$

Based on the aforementioned equations the updating laws for each of the network parameters can be expressed as following:

- Hidden layers parameters
  - Weight parameters

$$w_{np}^{l}(k+1) = w_{np}^{l}(k) + \eta[\sum_{z=1}^{N_{l+1}}(\delta_{n}^{l}(k)g_{z}^{l+1}b_{0z}^{l+1}w_{zn}^{l+1})F'(\widetilde{y}_{1n}^{l}(k))] \\ g_{n}^{l}[\sum_{i=0}^{m} b_{in}^{l}u_{p}^{l}(k-i) - \sum_{i=1}^{m} a_{in}^{l}S_{w_{pn}}^{l}(k-i)] \tag{2.1.15}$$

  - Filter feedback parameters

$$a_{n}^{l}(k+1) = a_{n}^{l}(k) - \eta[\sum_{z=1}^{N_{l+1}}(\delta_{n}^{l}(k)g_{z}^{l+1}b_{0z}^{l+1}w_{zn}^{l+1})F'(\widetilde{y}_{1n}^{l}(k))]g_{n}^{l}\widetilde{y}_{n}^{l}(k-i) \tag{2.1.16}$$

  - Filter feed-forward parameters

$$b_{n}^{l}(k+1) = b_{n}^{l}(k) - \eta[\sum_{z=1}^{N_{l+1}}(\delta_{n}^{l}(k)g_{z}^{l+1}b_{0z}^{l+1}w_{zn}^{l+1})F'(\widetilde{y}_{1n}^{l}(k))]g_{n}^{l}x_{n}^{l}(k-i) \tag{2.1.17}$$

  - Slope parameters

$$g_{n}^{l}(k+1) = g_{n}^{l}(k) - \eta[\sum_{z=1}^{N_{l+1}}(\delta_{n}^{l}(k)g_{z}^{l+1}b_{0z}^{l+1}w_{zn}^{l+1})F'(\widetilde{y}_{1n}^{l}(k))]g_{n}^{l}\widetilde{y}_{n}^{l}(k) \tag{2.1.18}$$

- Output layer parameters
  - Weight parameters

$$w_{np}^{l}(k+1) = w_{np}^{l}(k) + \eta[e_{n}(k)F'(\widetilde{y}_{1n}^{l}(k))]g_{n}^{l}[\sum_{i=0}^{m} b_{in}^{l}u_{p}^{l}(k-i) - \sum_{i=1}^{m} a_{in}^{l}S_{w_{pn}}^{l}(k-i)] \tag{2.1.19}$$

– Filter feedback parameters

$$a_n^l(k+1) = a_n^l(k) - \eta[e_n(k)F'(\widetilde{y}_{1n}^l(k))]g_n^l\widetilde{y}_n^l(k-i) \tag{2.1.20}$$

– Filter feed-forward parameters

$$b_n^l(k+1) = b_n^l(k) - \eta[e_n(k)F'(\widetilde{y}_{1n}^l(k))]g_n^l x_n^l(k-i) \tag{2.1.21}$$

– Slope parameters

$$g_n^l(k+1) = g_n^l(k) - \eta[e_n(k)F'(\widetilde{y}_{1n}^l(k))]\widetilde{y}_n^l(k) \tag{2.1.22}$$

2.2 Evolutionary Computation

Evolutionary computation (EC) is a recent field of research that is based on the concept of evolution and adaption [145, 149 - 152]. The principal objective of the evolutionary computation is to design highly robust, flexible and efficient algorithm to solve real-world problem where the conventional computing methods deal with lots of difficulties [153]. The main advantages of the evolutionary computations in comparison with conventional techniques are presented in the literature [154], which are the simplicity in concepts and computations, applicable to apply in extensive types of problems, acceptable for real world problem, ability of the self-optimization, etc.

According these advantages, the evolutionary computations approaches can be used in the problems where the environment changes dynamically and multi-objective optimization requirements. Traditional optimization applications may fail in processing inaccurate, noisy and complex data and they can be replaced by the evolutionary

computations techniques. The evolutionary computation contains three main techniques namely as, genetic algorithm (GA), evolutionary strategies (ES) and evolutionary programming (EP) which go back to 1960s [155].

2.2.1 Evolutionary Algorithms (EAs)

Recently, the evolutionary algorithm term has been used for the algorithms which implement the evolutionary computation. In complex optimization problems where the number of parameters is large and finding the analytical solutions are challenging, applying evolutionary algorithms are beneficial. EAs are capable of finding the optimal solutions globally over a domain. Evolutionary algorithms have been implemented for different applications such as combinational optimization problem [156]. The other applications are related to the design problems such as designing the artificial neural network topologies and finding the set of optimum weights by using EC techniques.

2.2.1.1 Principles of Evolutionary Algorithms

Evolutionary algorithms are based on three fundamental properties that make them different from other search algorithms. These primary properties are explained in following:

- Evolutionary algorithms rely on the population and they utilize the collective learning procedure of a population of individuals. Each evolutionary algorithm applies different methods to update the whole population in each of the iterations where the population contains the possible solutions of the problem. The initial population can be generated randomly from the solution space or the solutions that are provided by the local search procedures. These algorithms try to find the globally optimal solutions to the problem.

- A population is evolved by applying stochastic operators named as, mutation, recombination and selection. Mutation is applied to avoid from replication of individuals; the characteristics of the parents are passed to the offspring through recombination, and selection process choose the better individuals to reproduce in the next generations.

- The quality of each individuals is measured in their environment. The fitness of individuals is compared to each other and the selection procedure is fulfilled based on these quality measurements.

All of these major properties are same for the entire evolutionary algorithms the differences between algorithms are related to different representations of individuals and schemes to achieve fitness evaluation, selection and search operators are adopted differently in each of them.

2.2.2 Evolutionary Artificial Neural Networks (EANNs)

As mentioned in previous sections the evolutionary algorithms are applied in the problems with complexity and large number of parameters where finding the analytical solutions is challenging. These types of methods are valuable in finding the optimal solutions. One of these evolving systems is called neuro-genetic systems that are a main topic of research in evolutionary computation.

Recently, evolutionary artificial neural networks (EANNs) are received significant attention in the field of ANN design. One of the noticeable characteristics of EANNs is their adaptability to dynamic environment. This feature includes two forms of the adaptations namely as, evolution and learning [157] that makes EANNs capable of adapting to the environment and it changes more efficiently. EANNs are considered as the adaptive systems which are able to change their learning rules and architectures without human interference.

ANNs and EAs are evolved in different ways in order to achieve the objectives such as connection weight training, architecture design, learning rule adaption, input feature selection, connection weight initialization, etc. The three main evolving ANNs and EAs approaches are expressed in the following:

- Connection weights: The evolution of connection weights is a weight optimization method where the network architecture must be static. The evolution of connection weights is considered as an adaptive and global training technique that is valuable when the gradient-based training deals with difficulties.

- Learning rules: In this approach the adaption of the learning rules in ANN are achieved by means of evolution. It can also be considered as an adaptive process of finding the learning rules automatically.

- Architecture: In this method the ANNs are designed automatically without human intervention. The ANNs adjust their topologies including connection weights and structures through evolving.

Feature selection and the evolution of the transfer function of a neural network are the other methods that are employed in conjunction with the three aforementioned techniques to achieve more desirable results.

As previously explained, the evolutionary learning are applied in ANNs to prevent the difficulties related to the traditional gradient descent methods such as back-propagation that can result in trapping in local minima. EAs are less sensitive to the initial condition of the training. They explore to find the globally optimal solution, while a gradient descent method can only obtain the locally optimal solution in the neighborhood of the initial solution. EANNs can overcome these aforementioned difficulties. One of the beneficial aspects in designing procedure of the evolutionary artificial neural networks is that a near-optimal neural network with both structure

and weights can be evolved automatically and there is no need to use the trial and error.

Several research direction considered the design of the ANN as an optimization problem. Tettamanzi *et al.* in [145] explained the evolutionary systems and their interaction with neural and fuzzy systems. Different studies are done in the area of designing neural network based on evolutionary algorithms. Some EAs are applied in the weight optimization that can be considered as an alternative to training algorithm in the networks with static structures. The other works are focused on the topology, optimal learning parameters and neural network transfer functions.

2.3 Genetic Algorithms (GAs)

Genetic algorithm is the search and optimization algorithm that its concepts are based on genetics and natural selection. Genetic algorithms originally attributed to John Holland and his students in 1970s [158]. GA provides some advantages in comparison with other methods, mentioning as follows [159]:

- GA can perform the optimization with both continuous and discrete variables.
- GA does not need the derivative information and has the capability to avoid the local minima.
- GA can be applied for optimization problems with highly complex cost surfaces.
- GA can provide a list of optimal solutions and not only one solution.
- GA can efficiently deal with different objective functions such as, discontinuous multimodal, and noisy functions.

2.3.1 Genetic Algorithm Encoding

GA contains a pool of possible solutions that are encoded as chromosomes. The encoding can be fulfilled in different ways. The two common encoding approaches are called, binary encoding and real-valued encoding. The binary GA is used in the cases that the variables are naturally quantized while the real-valued GA is applied when the variables are continuous. The continuous genetic algorithm has two advantages in comparison with the binary GA. First of all, since in continuous GA the floating point numbers presents the variables the less storage is required. Secondly, for the reason that the chromosomes do not have to be decoded before the evaluation of the cost function, the continuous GA is faster than the binary GA. In this work, as the neural network parameters are continuous values the real-valued GA is applied.

2.3.2 Genetic Algorithm Procedure

The genetic algorithm procedure starts with generating an initial population where the members of this population can be selected at random or regarding some rules. In the next step, a fitness value is determined for each of the chromosomes in the population based on the defined fitness function. The fitness function measures the optimality of each of the possible solutions. According to the fitness value the best individuals are chosen to create the mating pool. After generating the mating pool, the next generation is created by using crossover. In the crossover process, two parents are selected randomly from the mating pool and they exchange their genetic material and produce offsprings. In the next step the mutation operator is applied on the offspring pool and randomly change part of the offspring's genetic. Finally, the new generated offsprings are compared with the chromosomes based on their fitness value to decide which chromosomes should be survived to the next generation. These steps

should be repeated till the termination criterion is accomplished.

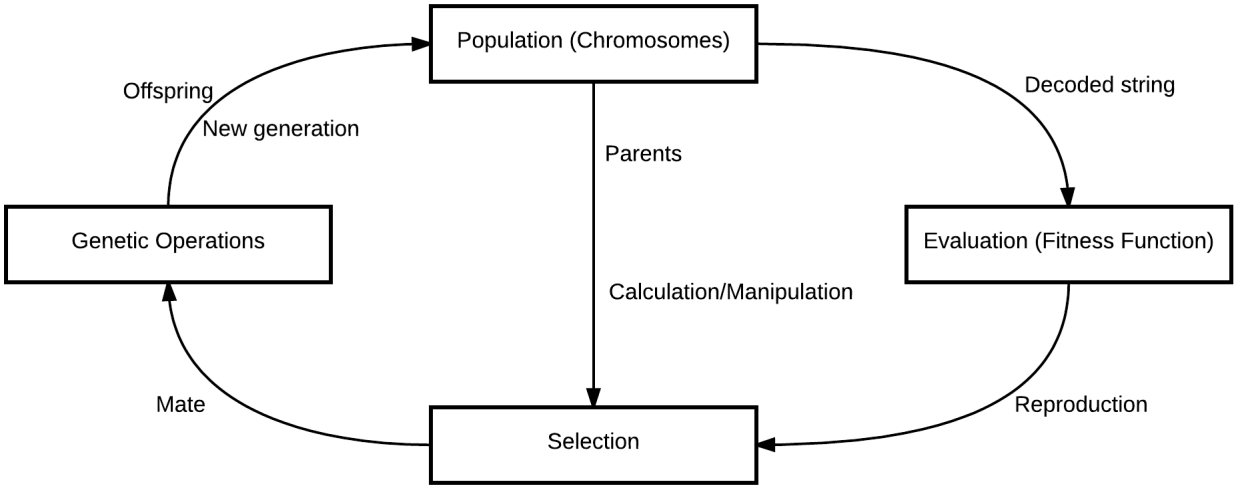


Figure 2.3: Genetic algorithm cycle [160]

The genetic algorithm process is illustrated through the GA cycle in Figure 2.3. In this figure, reproduction is the procedure that two or more parent is combined to obtain one or more offspring. In fitness evaluation step, the individual's quality is calculated. In mutation process the original genetic material has been randomly changed in one individual to produce a new version of it. Selection proceeding helps to decide which individuals should be chosen for reproduction and mutation in order to generate new search points.

2.3.3 Genetic Algorithm Operators

2.3.3.1 Selection

Selection is a critical operator in improving the performance of the GA. During the selection process the better individuals are allowed to pass their genetic materials to the next generation. The fitness of each individual determines the goodness of each chromosomes. The selection can be implemented in different approaches. The

common selection methods are explained as follows:

1) Roulette Wheel Selection: In this selection technique, the chromosomes are placed on the roulette wheel regarding their fitness value. Each segment of the roulette wheel is related to each of the chromosomes that the size of the segments are proportional to the fitness values of the individuals. In other words, the larger segment is for the chromosome with higher value of the fitness. The roulette wheel is spinned until it stops. The individual corresponding to the place that the roulette wheel stopped, is selected. This process is repeated until the required number of chromosomes is chosen. Individuals with higher fitness have the more chance to be selected. The disadvantage of this method is that some good individuals may not be survive to the next generation.

2) Rank Selection: In this method, the individuals are ranked based on their fitness values. The chromosomes with higher fitness values will be ranked higher and those with lower fitness values will have the lower ranks. The chromosomes will be selected with a probability that is linearly proportional to the rank of the individuals in the population.

3) Natural Selection: This method inspired by natural selection where fitness values associated with the chromosomes are ranked from highest values to lowest values. Then the chromosomes with the most fitness survive and the least fit chromosomes are eliminated and replaced by the new generated offsprings.

2.3.3.2 Crossover

In crossover process, pairs of parents exchange their materials to generate new offspring. The crossover procedure can be divided in two stages. In the first stage, pairs of chromosomes are mated randomly to produce of two new offsprings. Secondly, a point is selected in the chromosomes to exchange their genetic material. The most

common crossover methods are described as follows:

1) Single point crossover

In single point crossover, in the first step a crossover point is chosen randomly. Secondly, all genes beyond the selected point are exchanged between two parents. The process of the single point crossover is expressed in following equations:

$$\begin{aligned} Parent_1 &= \left[p_{m_1}, p_{m_2}\star, p_{m_3}, p_{m_4}, p_{m_5}, p_{m_6}, \cdots, p_{mN_{var}}\right] \\ Parent_2 &= \left[p_{d_1}, p_{d_2}\star, p_{d_3}, p_{d_4}, p_{d_5}, p_{d_6}, \cdots, p_{dN_{var}}\right] \end{aligned} \tag{2.3.1}$$

$$\begin{aligned} Offspring_1 &= \left[p_{m_1}, p_{m_2}\star, p_{d_3}, p_{d_4}, p_{d_5}, p_{d_6}, \cdots, p_{dN_{var}}\right] \\ Offspring_2 &= \left[p_{d_1}, p_{d_2}\star, p_{m_3}, p_{m_4}, p_{m_5}, p_{m_6}, \cdots, p_{mN_{var}}\right] \end{aligned} \tag{2.3.2}$$

In equation (2.3.1), two parents are mated where the $\star$ is the crossover point that is selected at random. In equation (2.3.2), the generated offsprings after applying crossover on parents are shown.

2) Two point crossover

The two point crossover is very much alike to single point crossover excepting the number of crossover points. In this crossover approach two crossover points are selected at random. The two point crossover method is shown in following equations:

$$\begin{aligned} Parent_1 &= \left[p_{m_1}, p_{m_2}\star, p_{m_3}, p_{m_4}, p_{m_5}\star, p_{m_6}, \cdots, p_{mN_{var}}\right] \\ Parent_2 &= \left[p_{d_1}, p_{d_2}\star, p_{d_3}, p_{d_4}, p_{d_5}\star, p_{d_6}, \cdots, p_{dN_{var}}\right] \end{aligned} \tag{2.3.3}$$

$$\begin{aligned} Offspring_1 &= \left[p_{m_1}, p_{m_2}\star, p_{d_3}, p_{d_4}, p_{d_5}\star, p_{m_6}, \cdots, p_{mN_{var}}\right] \\ Offspring_2 &= \left[p_{d_1}, p_{d_2}\star, p_{m_3}, p_{m_4}, p_{m_5}\star, p_{d_6}, \cdots, p_{dN_{var}}\right] \end{aligned} \tag{2.3.4}$$

In equation (2.3.3), two parents are mated where the two $\star$ is the crossover points

that are selected at random. In equation (2.3.4), the generated offsprings after applying crossover on parents are shown.

3) Uniform crossover

In uniform crossover a gene value of the first parent is assigned to the first offspring and the value of the second parent's gene is assigned to the second offspring with probability of p_c which is called as mixing ratio. The crossover operator decides which gene values of the parents contributes to the offsprings chromosomes. An exemplar of an uniform crossover is mentioned in following equations:

$$\begin{aligned} Parent_1 &= \left[p_{m_1}, p_{m_2}, p_{m_3}, p_{m_4}, p_{m_5}, p_{m_6}, \cdots, p_{mN_{var}}\right] \\ Parent_2 &= \left[p_{d_1}, p_{d_2}, p_{d_3}, p_{d_4}, p_{d_5}, p_{d_6}, \cdots, p_{dN_{var}}\right] \end{aligned} \tag{2.3.5}$$

$$\begin{aligned} Offspring_1 &= \left[p_{m_1}, p_{m_2}, p_{d_3}, p_{d_4}, p_{m_5}, p_{d_6}, \cdots, p_{mN_{var}}\right] \\ Offspring_2 &= \left[p_{m_1}, p_{d_2}, p_{d_3}, p_{d_4}, p_{m_5}, p_{d_6}, \cdots, p_{dN_{var}}\right] \end{aligned} \tag{2.3.6}$$

In the aforementioned example the mixing ratio for the uniform crossover is set to p_c, since the following genes values ($p_{m_3}, p_{m_4}, p_{m_6} > p_c$ and $p_{d_1}, p_{d_5} > p_c$) in two parents are bigger than the probability they are swapped with the value of the other parent's gene.

The main disadvantage of applying crossover is that it decreases the diversity of the population. This decrement in diversity leads to a population with identical chromosomes that are not able to generate new chromosomes. In order to avoid this problem and keep the diversity of the population the mutation is applied.

2.3.3.3 Mutation

Mutation procedure is implemented differently for each type of chromosomes. The mutation for a binary chromosome is performed through flipping some bits in the

chromosome while for a real valued chromosome, the gene to be mutated is changed with a random values which is selected between it given ranges. The mutation improves the performance of the GA by avoiding fixed values of genes. On the other hand, mutation can negatively affect the good combinations that found. In order to reduce the consequences of this negative effect the mutation rate should be set to a small number. The mutation rate should be selected based on the population size. In smaller size population the probability that a specific bit in all the chromosomes be stuck at the same value augments. In order to keep the diversity in the population the larger mutation rate should be chosen for the smaller populations.

2.4 Underwater Vehicle

Unmanned underwater vehicles (UUVs) are all type of underwater robots which are operated with minimum or without intervention of human operator. These vehicles can be divided into two groups named as, remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs). The term ROV denotes an underwater vehicle that physically linked, via the tether, to an operator that can be on a submarine or on a surface ship. AUV, on the other side, is supposed to be completely autonomous, thus relying to onboard power system and intelligence [101]. In case of missions that require interaction with the environment, the vehicle can be equipped with one or more manipulators; in this case the system is usually called underwater vehicle-manipulator system (UVMS). Currently, there are about more than 100 prototypes in the laboratories all over the world. For instance, r2D4 developed at URA laboratory of the University of Tokyo (Tokyo, Japan), ABE of the Deep Submerge Laboratory of the Woods Hole Oceanographic Institution (Massachusetts, USA), Odissey IId belonging to the AUV Laboratory of the Massachusetts Institute

of Technology (Massachusetts, USA), ODIN III designed at the Autonomous Systems Laboratory of the University of Hawaii (Hawaii, USA), Phoenix and ARIES, torpedo-like vehicles developed at the Naval Postgraduate School (California, USA) and Girona500 belonging to the University of Girona (Girona, Spain) [101].

2.5 Model of AUV

Modeling of the AUV is divided into two main categories that are named as Kinematics and Dynamics, which analyze the geometrical aspects of the motion and the forces causing the motion respectively. The AUV as a rigid body has six degree of freedom, which determines the position and orientation of the AUV. The 6 DOF are defined in table 2.1. In this table the first three coordinates and their time derivatives determine the position and translational motion of the AUV along the x, y, and z axes, while the last three coordinates and their time derivatives describe the orientation and rotational motion.

Table 2.1: 6 DOFs of the AUV

DOF	Motion	Force and Moment	Velocity	Position
1	Motion in x-direction (Surge)	X	u	x
2	Motion in y-direction (Sway)	Y	v	y
3	Motion in z-direction (Heave)	Z	w	z
4	Rotation about x-axis (Roll)	K	p	ϕ
5	Rotation about y-axis (Pitch)	M	q	θ
6	Rotation about z-axis (Yaw)	N	r	ψ

2.5.1 Kinematics of the AUV

In order to present the motion of the AUV in 6 DOFs, two coordinate frames are applied. I) Earth-Fixed (Inertial) Frame II) Body-Fixed Frame. The second coordinate frame calling body-fixed reference frame, the moving reference frame, which is fixed to the vehicle. The origin of the body-fixed frame coincides with the Center of the Gravity (CG) and the body axes X_o, Y_o and Z_o coincide with principal axes of inertia. The following definitions are used for these axes [1]:

- X: Longitudinal axis (Directed from aft to fore)
- Y: Transverse axis (Directed to starboard)
- Z: Normal axis (Directed from top to bottom)

The coordinate frames of the AUV are illustrated in Figure 2.4.

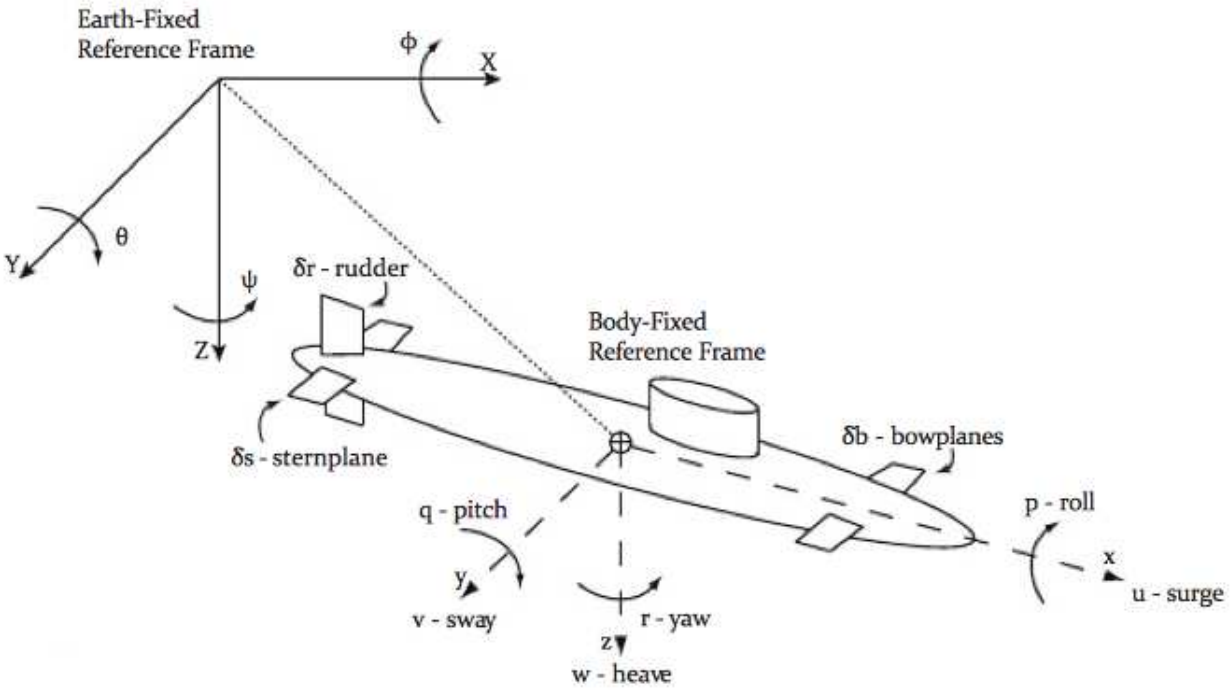


Figure 2.4: Coordinate frames of AUV [161].

Motion of the body-fixed frame is described relative to inertial frame, therefore the linear and angular velocities of the vehicle should be expressed in body-fixed frame while position and orientation should be described with respect to inertial frame. The following vectors describe the motion of an underwater vehicle in 6 DOF:

$$\eta = [\eta_1^T \quad \eta_2^T]^T \quad \eta_1 = [x \quad y \quad z]^T \quad \eta_2 = [\phi \quad \theta \quad \psi]^T \tag{2.5.1}$$

$$v = [v_1^T \quad v_2^T]^T \quad v_1 = [u \quad v \quad w]^T \quad v_2 = [p \quad q \quad r]^T \tag{2.5.2}$$

$$\tau = [\tau_1^T \quad \tau_2^T]^T \quad \tau_1 = [X \quad Y \quad Z]^T \quad \tau_2 = [K \quad M \quad N]^T \tag{2.5.3}$$

where η describes the position and orientation of the vehicle with respect to the earth-fixed reference frame, v denotes the translational and rotational velocities with respect to the body-fixed reference frame, and τ is the total forces and moments acting on the vehicle with respect to the body-fixed reference frame. Vehicle's path relative to the earth-fixed coordinate system is determined by a velocity transformation in equation (2.5.4).

$$\dot{\eta}_1 = J_1(\eta_2)v_1 \tag{2.5.4}$$

where, $J_1(\eta_2)$ is a transformation matrix that is calculated as follows:

$$J_1(\eta_2) = \begin{bmatrix} cos\psi cos\theta & -sin\psi cos\phi + cos\psi sin\theta sin\phi & sin\psi cos\phi + cos\psi sin\theta cos\phi \\ sin\psi cos\theta & cos\psi cos\phi + sin\psi sin\theta sin\phi & -cos\psi sin\phi + sin\psi sin\theta cos\phi \\ -sin\theta & cos\theta sin\phi & cos\theta cos\phi \end{bmatrix} \tag{2.5.5}$$

The body-fixed angular velocity vector v_2 and the Euler rate vector $\dot{\eta}_2$ have the relation through the transformation matrix $J_2(\eta_2)$ as bellow:

$$\dot{\eta}_2 = J_2(\eta_2)v_2 \tag{2.5.6}$$

Where the transformation matrix is calculated in the equation (2.5.7).

$$J_2(\eta_2) = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & \frac{sin\phi}{cos\theta} & \frac{cos\phi}{cos\theta} \end{bmatrix} \tag{2.5.7}$$

2.5.2 Nonlinear Dynamics of the AUV

The 6 DOF nonlinear dynamic equation motion of the underwater vehicle is expressed as follows [1]:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \tag{2.5.8}$$

where,

- M, is the inertia matrix for the rigid body and added mass.
- $C(v)$, is the Coriolis and centripetal matrix for the rigid body and added mass.
- $D(v)$, is the damping matrix.
- $g(\eta)$, is the gravitational forces and moments vector.
- τ, is the external force and torque input vector.
- v, is the velocity state vector.

Dynamics of the AUV are categorized into Translational and Rotational Motion. The equations of these two motions are respectively mentioning as follows:

$$m(\dot{v_o} + \omega \times v_o + \dot{\omega} \times r_G + \omega \times (\omega \times r_G)) = f_o \tag{2.5.9}$$

$$I_o\dot{w} + \omega \times (I_o\omega) + mr_G \times (\dot{v_o} + \omega \times v_o) = m_o \tag{2.5.10}$$

where,

- $f_o = \tau_1 = [X \quad Y \quad Z]^T \rightarrow$ External Forces
- $m_o = \tau_2 = [K \quad M \quad N]^T \rightarrow$ Moment of External Forces
- $v_o = V_1 = [u \quad v \quad w]^T \rightarrow$ Linear Velocity
- $\omega = V_2 = [p \quad q \quad r]^T \rightarrow$ Angular Velocity
- $r_G = [x_G \quad y_G \quad z_G]^T \rightarrow$ Center of Gravity

The rigid body equations of motion for an underwater vehicle can be expressed as:

$$m(\dot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})) = X \tag{2.5.11}$$

$$m(\dot{v} - wp + ur - y_G(r^2 + p^2) + z_G(qr - \dot{p}) + x_G(qp + \dot{r})) = Y \tag{2.5.12}$$

$$m(\dot{w} - uq + vp - z_G(p^2 + q^2) + x_G(rp - \dot{q}) + y_G(rq + \dot{p})) = Z \tag{2.5.13}$$

$$I_x\dot{p}+(I_z-I_y)qr-(\dot{r}+pq)I_{xz}+(r^2-q^2)I_{yz}+(pr-\dot{q})I_{xy}+m[y_G(\dot{w}-uq+vq)-z_G(\dot{v}-wp+ur) = K \tag{2.5.14}$$

$$I_y\dot{q}+(I_x-I_z)rp-(\dot{p}+qr)I_{xy}+(p^2-r^2)I_{zx}+(pq-\dot{r})I_{yz}+m[z_G(\dot{u}-vr+wq)-x_G(\dot{w}-uq+vp) = M \tag{2.5.15}$$

$$I_z\dot{r}+(I_y-I_x)pq-(\dot{q}+rp)I_{yz}+(q^2-p^2)I_{xy}+(rq-\dot{p})I_{zx}+m[x_G(\dot{v}-wp+ur)-y_G(\dot{u}-vr+wp) = N \tag{2.5.16}$$

2.5.2.1 Mass and Inertia Matrix

The M contains two parts: I) Rigid body mass and inertia (M_{RB}) II) Hydrodynamic added mass (M_A), which is expressed as [1]:

$$M = M_{RB} + M_A \tag{2.5.17}$$

The effect of the added mass is expressed by matrix M_A. The elements of this matrix depend on the shape of the vehicle, which have constant values when the AUV fully submerged. This matrix is written as following:

$$M_A = \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{2.5.18}$$

The rigid body mass, M_{RB}, is expressed as:

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_G & -my_G \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & my_G & -mx_G & 0 \\ 0 & -mx_G & my_G & I_{xx} & -I_{xy} & -I_{xz} \\ mz_G & 0 & -mx_G & -I_{yx} & I_{yy} & -I_{yz} \\ -my_G & mx_G & 0 & -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{2.5.19}$$

In equation (2.5.19) the m denotes the mass of the AUV and the I terms are the inertial tensors.

In this thesis, it is assumed that the AUV is symmetric in all planes and the origin of the body-fixed frame is located at the center of the gravity of the AUV (i.e. $r_G = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$), therefore the M_A and M_{RB} are simplified into equations (2.5.20)

and (2.5.21) respectively.

$$M_A = \begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{2.5.20}$$

The rigid body mass, M_{RB}, is expressed as:

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \tag{2.5.21}$$

2.5.2.2 Coriolis and Centripetal Matrix

$C(v)$ is the Coriolis and Centripetal matrix that is shown as [1]:

$$C(v) = C_{RB}(v) + C_A(v) \tag{2.5.22}$$

C_A is called Coriolis-like matrix, that is written as:

$$C_A = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix} \tag{2.5.23}$$

where,

- $a_1 = X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{w}}w + X_{\dot{p}}p + X_{\dot{q}}q + X_{\dot{r}}r$
- $a_2 = X_{\dot{v}}u + Y_{\dot{v}}v + Y_{\dot{w}}w + Y_{\dot{p}}p + Y_{\dot{q}}q + Y_{\dot{r}}r$
- $a_3 = X_{\dot{w}}u + Y_{\dot{w}}v + Z_{\dot{w}}w + Z_{\dot{p}}p + Z_{\dot{q}}q + Z_{\dot{r}}r$
- $b_1 = X_{\dot{p}}u + Y_{\dot{p}}v + Z_{\dot{p}}w + K_{\dot{p}}p + K_{\dot{q}}q + K_{\dot{r}}r$
- $b_2 = X_{\dot{q}}u + Y_{\dot{q}}v + Z_{\dot{q}}w + K_{\dot{q}}p + M_{\dot{q}}q + M_{\dot{r}}r$
- $b_3 = X_{\dot{r}}u + Y_{\dot{r}}v + Z_{\dot{r}}w + K_{\dot{r}}p + M_{\dot{r}}q + N_{\dot{r}}r$

According to this assumption that the AUV is symmetric in all planes and the origin of the body-fixed frame is located at the center of the gravity of the AUV, the centripetal matrix ,C_{RB}, is defined as:

$$C_{RB} = \begin{bmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & mw & -mv & 0 & I_{zz}r & -I_{yy}q \\ -mw & 0 & mu & -I_{zz}r & 0 & I_{xx}p \\ mv & -mu & 0 & I_{yy}q & -I_{xx}p & 0 \end{bmatrix} \tag{2.5.24}$$

2.5.2.3 Hydrodynamic Damping Matrix

In underwater vehicles the hydrodynamic damping includes the drag and the lift forces. According to this fact the AUV only operates at a low speed, the lift forces can be neglected in comparison with the drag forces. The drag forces can be divided into a linear and quadratic term as follows [1]:

$$D(v) \triangleq D_q(v) + D_l(v) \tag{2.5.25}$$

where, $D_q(v)$ denotes to the quadratic term and $D_l(v)$ is the linear drag. With respect to this assumption that the AUV is symmetric in all planes then the linear and quadratic drag term can be expressed as follows:

$$D_l(v) = \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_w & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r \end{bmatrix} \tag{2.5.26}$$

$$D_q(v) = \begin{bmatrix} X_{u|u|}\,|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{v|v|}\,|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{w|w|}\,|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{p|p|}\,|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{q|q|}\,|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{r|r|}\,|r| \end{bmatrix} \tag{2.5.27}$$

2.5.2.4 Hydro static Forces and Moments (Restoring Forces and Moments)

The gravitational and buoyant forces are considered as the restoring forces for the underwater vehicles. Assume that the m be the mass of the vehicle including water in free-floating spaces,$\triangledown$ be the volume of fluid displaced by the vehicle, g be the acceleration of gravity (positive downwards), and ρ be the fluid density, the buoyancy force and gravity force are expressed as follows [1]:

- $W = mg \rightarrow$ Weight Force
- $B = \rho g \triangledown \rightarrow$ Buoyancy Force

The hydro static forces and moments are described as follows:

$$g(\eta) = \begin{bmatrix} (W-B)sin\theta \\ -(W-B)cos\theta sin\phi \\ -(W-B)cos\theta cos\phi \\ -(y_G W - y_B)cos\theta cos\phi + (z_G W - z_B B)cos\theta sin\phi \\ (z_G W - z_B B)sin\theta + (x_G W - x_B B)cos\theta sin\phi \\ -(x_G W - x_B B)cos\theta sin\phi + (y_G W - y_B B)sin\theta \end{bmatrix} \tag{2.5.28}$$

In equation (2.5.28), the $r_G = \begin{bmatrix} x_G, & y_G, & z_G \end{bmatrix}$ and the $r_B = \begin{bmatrix} x_B, & y_B, & z_B \end{bmatrix}$ are the center of gravity and the center of buoyancy of the AUV respectively.

2.5.2.5 Physical Parameters of the AUV

The physical parameters of the of the proposed autonomous underwater vehicle in this work, are expressed in Table 2.2 [175].

Table 2.2: The physical parameters of the AUV [175].

Parameters	Numerical values
Mass $(m)[kg]$	1089.8142
Volume m^3	0.97
$X_{\dot{u}}$ $[kg]$	-26.2096
$Y_{\dot{v}}$ $[kg]$	-1043.5908
$Z_{\dot{w}}$ $[kg]$	-1043.5908
$K_{\dot{p}}$ $[kg.m^2]$	0
$M_{\dot{q}}$ $[kg.m^2]$	-1907.0841
$N_{\dot{r}}$ $[kg.m^2]$	-1907.0841
I_{xx} $[kgm^2]$	36.6777
I_{yy} $[kgm^2]$	2154.3075
I_{zz} $[kgm^2]$	2154.3075
$[X_G, Y_G, Z_G]$ $[m]$	$[0, 0, 0]$
$[X_B, Y_B, Z_B]$ $[m]$	$[0, 0, 0]$
X_u $[kg.m^2]$	-3×10^{-3}
Y_v $[kg.m^2]$	-1×10^{-1}
Z_w $[kg.m^2]$	-3×10^{-1}
K_p $[kg.m^2]$	1.1×10^{-2}
M_q $[kg.m^2]$	-1.6×10^{-3}
N_r $[kg.m^2]$	1.6×10^{-2}
$X_{u\|u\|}\|u\|$ $[kg/m]$	-25.5028
$Y_{v\|v\|}\|v\|$ $[kg/m]$	-920.1417
$Z_{w\|w\|}\|w\|$ $[kg/m]$	-920.1417
$K_{p\|p\|}\|p\|$ $[kg.m^2]$	-0.3114
$M_{q\|q\|}\|q\|$ $[kg.m^2]$	-2850.1982
$N_{r\|r\|}\|r\|$ $[kg.m^2]$	-2850.1982

2.5.2.6 Environmental Disturbances

Ocean currents and waves are considered as the environmental disturbances for the AUV. In this thesis it is assumed that the AUVs are deeply submerged, therefore the wave induced currents are neglected. Ocean currents are horizontal and vertical circulating systems of ocean waters produced by gravity, wind friction and water density variation in different parts of the ocean [167]. The ocean currents can be modeled as a Gauss-Markov process as following [1]:

$$M\dot{V}_c(t) + \mu_c V_c(t) = w_c(t) \tag{2.5.29}$$

where $w_c(t)$ is Gaussian white noise, and μ_c is a constant that in many cases is chosen to be zero ($\mu_c = 0$). In integration process a saturation element is applied to limit the ocean current speed that is expressed as following:

$$V_{min} \leq V_c(t) \leq V_{max} \tag{2.5.30}$$

Considering this assumption that the fluid is irrotational, the the earth-fixed current velocity vector is denoted by $\left[u_c^E,\, v_c^E,\, w_c^E\right]$. The vertical ocean current components are generally negligible in comparison to lateral currents [180], therefore the earth-fixed current velocity is considered as: $\left[u_c^E,\, v_c^E,\, 0\right]$.

where,

$$V_c = \sqrt{(u_c^E)^2) + (v_c^E)^2)} \tag{2.5.31}$$

and,

$$\begin{aligned} u_c^E &= V_c cos(\psi_c) cos(\theta_c) \\ v_c^E &= V_c sin(\psi_c) cos(\theta_c) \end{aligned} \tag{2.5.32}$$

where ψ_c and θ_c denote horizontal and vertical current angel respectively.

The current velocity in the the body-fixed can be computed as follows:

$$\left[u_c^B,\, v_c^B,\, 0_{4\times1}\right] = J_1^T(\eta_2)\left[u_c^E,\, v_c^E,\, 0_{4\times1}\right]^T \tag{2.5.33}$$

Therefore, the dynamical model of AUV with relative velocity, $V_r = \left[u - u_c^B,\, v - v_c^B,\, w, p, q, r\right]$, can be written as:

$$\begin{aligned} M\dot{V}_r + C(V_r)V_r + D(V_r)V_r + g(\eta) &= \tau \\ \dot{\eta} &= J(\eta)V_r + V_c^E \end{aligned} \tag{2.5.34}$$

where $J(\eta)$ is transformation matrix and is defined as follow:

$$\begin{bmatrix} J_1(\eta_2) & 0_{3\times3} \\ 0_{3\times3} & J_2(\eta_2) \end{bmatrix} \tag{2.5.35}$$

2.6 Control of Single AUV

The computed torque control method is proposed in this section in order to control the AUV. This method has the advantage of making a nonlinear system appear linear. Computed torque control is a common method that is applied in robotics. This technique allows for the feedback linearization of a nonlinear system which is very useful in the case of an AUV since its dynamic model is highly nonlinear. According to this fact that the proposed technique makes a nonlinear system appear linear, therefore the linear control methods like PD control can be used in controlling the AUV.

The authors in [162], presented the computed torque control method for robots which is explained as following. The dynamic model of a robot can be written as:

$$M(\theta)\ddot{\theta} + N(\theta, \dot{\theta}) = \tau \tag{2.6.1}$$

where θ is a position state vector, M is a mass and inertia matrix, N is a combined matrix representing the Coriolis, gravitational and friction forces, and τ represents the joint forces and torque vector. The computed torque control law for equation (2.6.1) is expressed as:

$$\tau = \alpha\tau' + \beta \tag{2.6.2}$$

where α is chosen to be $M(\theta)$ and β as $N(\theta, \dot{\theta})$. Then, an appropriate linear controller for τ' is chosen.

In the next section the aforementioned method is applied for the AUV.

2.6.1 Applying Computed Torque Control to the AUV

The dynamic model of an AUV can be expressed as:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \tag{2.6.3}$$

Considering the similarity between the dynamic model of the AUV and the equation (2.6.1), the computed torque control technique can be applied to the AUV. The computed torque control law for the AUV's dynamic model is expressed as [170]:

$$\tau = \alpha\tau' + \beta \tag{2.6.4}$$

where α is chosen to be M and β is chosen as $D(v)v + C(v)v + g(\eta)$.

By choosing τ' as [170],

$$\tau' = \ddot{v}_d + k_v\dot{\epsilon} + k_p\epsilon \tag{2.6.5}$$

the computed torque controller is extended to a PD tracking controller where $\ddot{v_d}$ represents the desired acceleration vector, while $\dot{\epsilon}$ and ϵ denote the tracking error

vectors for velocity and position respectively. Then the k_p and k_v can be chosen with respect to the desirable system characteristics. The error vector for the position state vector is written as:

$$\epsilon = P_d - P \tag{2.6.6}$$

where P_d denotes the desired position vector and P is the current position vector. The position vector contains spatial coordinates and the attitude of the AUV.

The error vector for the velocity state vector is expressed as:

$$\dot{\epsilon} = V_d - V \tag{2.6.7}$$

where V_d denotes the desired velocity vector and V the current velocity vector. The velocity vector includes both linear and angular velocities.

2.7 Formation Control of Multiple AUVs

In this section the formation control of multiple AUVs is presented. The formation control of multiple AUVs means that each vehicle follows a predefined path, and maintain in a geometric configuration with its neighbors while performing its tasks during the mission. The proposed method in this section assumes that the interconnection topology of AUVs is fixed, that means each AUV exchange information with a fixed vehicles during the mission. In this approach the nonlinear model of the AUV is applied and the environmental disturbances are considered.

2.7.1 Formation Controller Design

In this thesis a decentralized architecture via the virtual structure approach is developed to control the formation of the four AUVs [164]. In the presented approach, four

coordinate reference frames are used in formation of AUVs. The inertial frame F_O, the formation frame F_F which is fixed at the virtual center of the formation, body frame F_b and reference frame F_b^d which denotes the desired configuration for each AUV. In the virtual structure approach, the entire desired formation is considered as a single structure with a formation frame F_F located at its virtual center of mass to represent its configuration. The virtual structure then has position $r_F \in \mathbb{R}^3$, velocity $v_F \in \mathbb{R}^3$, attitude $q_F \in \mathbb{R}^4$ and angular velocity $\omega_F \in \mathbb{R}^3$ relative to inertial frame F_O.

Let $r_i \in \mathbb{R}^3$, $v_i \in \mathbb{R}^3$, $q_i \in \mathbb{R}^4$ and $\omega_i \in \mathbb{R}^3$ represent the position, velocity, attitude and angular velocity of the i^{th} AUV relative to the inertial frame F_O. Similarly, let r_{iF}, v_{iF}, q_{iF} and ω_{iF} represent position, velocity, attitude and angular velocity of the i^{th} AUV relative to formation frame F_F and a superscript "d" represent the desired state of each AUV relative to F_O or F_F.

The actual states of the i^{th} place holder represent the desired states of the i^{th} AUV, hence these states are denoted by q_{iF}^d and ω_{iF}^d.

The state of the virtual structure is defined as:

$$\xi = [r_F^T, v_F^T, q_F^T, \omega_F^T]^T \tag{2.7.1}$$

If each AUV has knowledge of ξ and of its own desired position and orientation with respect to the virtual structure, then formation keeping is transformed into an individual tracking problem. Therefore, the vector ξ represents the minimum amount of information needed by each AUV to coordinate its motion with the group. Given r_F, v_F, q_F and ω_F, the desired states for the i^{th} AUV are expressed as:

$$[r_i^d]_o = [r_F]_o \tag{2.7.2}$$

$$[v_i^d]_o = [v_F]_o \tag{2.7.3}$$

$$[q_i^d]_o = [q_F]_o[q_{iF}^d]_F \tag{2.7.4}$$

$$[\omega_i^d]_o = [\omega_F]_o \tag{2.7.5}$$

2.7.2 Decentralized Architecture

In the decentralized architecture, each AUV in the formation instantiates a local copy of the coordination variable $\xi_i = [r_{Fi}^T, v_{Fi}^T, q_{Fi}^T, \omega_{Fi}^T]$. The ξ_i represents the coordination variable instantiated in the i^{th} AUV corresponding to the coordination variable ξ defined in equation (2.7.1).

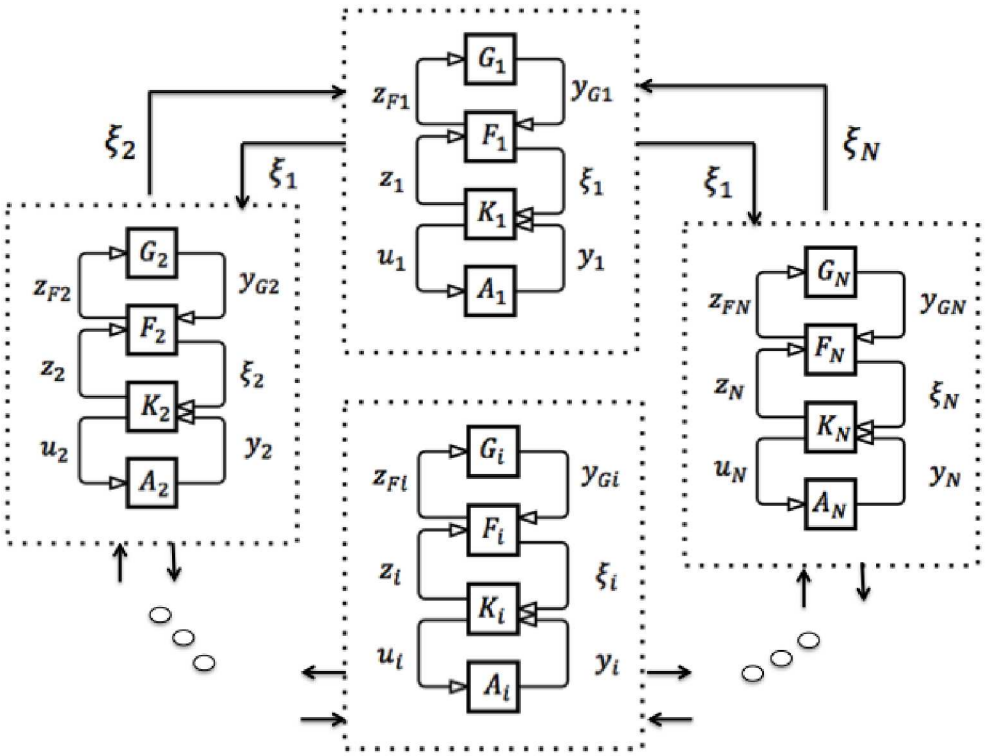


Figure 2.5: Decentralized architecture via the virtual structure approach [164].

A bidirectional ring topology is used to communicate the coordination variable

instantiations, to bring each local instantiation into consensus. The decentralized architecture via virtual structure approach is illustrated in Fig 2.5. In this figure, block G_i is a discrete event supervisor for the i^{th} AUV, block F_i is the formation control module, which produces and broadcasts coordination variable, system K_i is the local AUV controller for the i^{th} AUV, and A_i represents the i^{th} AUV. When the formation maneuver starts, each discrete event supervisor G_i outputs the current formation pattern to the formation control module F_i. Each formation control module implements a coordination variable instantiation ξ_i. Formation control module F_i then sends its coordination variable instantiation ξ_i to the local AUV controller K_i. Based on ξ_i the local controller K_i derives the desired states for the i^{th} AUV.

2.7.3 Formation Control Strategy

In the presented decentralized formation control via the virtual structure approach the major tasks need to be carried out is to control each virtual structure instantiation into consensus [164].

2.7.3.1 Formation Control Strategy for Each Virtual Structure

Let define ξ_i as the i^{th} coordination variable instantiation and ξ^d as the current k^{th} desired constant goal for the coordination vriable instantiations, i.e. the current formation pattern. The error state for the i^{th} coordination variable instantiation is defined as:

$$\widetilde{\xi}_i = \xi_i - \xi^d = [\widetilde{r}_{Fi}^T, \widetilde{v}_{Fi}^T, \widetilde{q}_{Fi}^T, \widetilde{\omega}_{Fi}^T]^T \tag{2.7.6}$$

where ξ^d represents a desired formation pattern to be achieved. There are two objectives for the coordination variable implemented in each AUV. The first objective is to reach its desired constant goal ξ^d defined by the formation pattern set and the second objective is to drive each instantiation to consensus, meaning that $\xi_1 = \xi_2 = ... = \xi_n$.

The goal seeking error between ξ_i and ξ^d is defined as:

$$E_G(t) = \sum_{i=1}^{n} \left\| \xi_i - \xi^d \right\|^2 \tag{2.7.7}$$

Also total consensus error between neighboring coordination variable instantiation is defined as:

$$E_A(t) = \sum_{i=1}^{n} \left\| \xi_i - \xi_{i+1} \right\|^2 \tag{2.7.8}$$

where $\xi_{n+1} = \xi_1$ and $\xi_0 = \xi_n$. Defining $E(t) = E_G(t) + E_A(t)$, then the control objective is to drive $E(t)$ to zero asymptotically.

The proposed control force f_{F_i} is given by [157]:

$$f_{F_i} = m_F[-K_G(r_{F_i} - r_F^d) - K_A(r_{F_i} - r_{F_{i+1}}) - K_A(r_{F_i} - r_{F_{i-1}}) - D_A(v_{F_i} - v_{F_{i+1}}) - D_A(v_{F_i} - v_{F_{i-1}})] \tag{2.7.9}$$

where K_G is a symmetrical positive-definite matrix and K_A and D_A are symmetrical positive-semi-definite matrices.

The proposed control torque τ_{Fi} is given as [157]:

$$\tau_{Fi} = k_G \widehat{q_F^{d*} q_{Fi}} - k_A \widehat{q_{F(i+1)}^{d*} q_{Fi}} - D_A(\omega_{Fi} - \omega_{F(i+1)}) - k_A \widehat{q_{F(i-1)}^{d*} q_{Fi}} - D_A(\omega_{Fi} - \omega_{F(i-1)}) \tag{2.7.10}$$

where $K_G > 0$ and $K_A \geq 0$ are scalars, D_A is symmetrical positive-semi-definite matrix and $\widehat{q}$ represents the vector part of the unit quaternion.

2.8 Propulsion System Modeling

Propellers are considered as the main source of force producing in AUVs, and their performance affects the AUVs mission fulfillment. The major problem in propellers is that the produced thrust is reduced by many factors such as, changes in the in-line

water velocity, cross flows, ventilation, in-and-out of water effects, wave-induced water velocities, interaction between the vessel hull and propellers and between propellers. This thrust losses affects the performance of the AUVs significantly. Therefore, applying a FDI system, which is capable of detecting and isolating the faults in the propeller is essential.

2.8.1 Propeller System

The applied propeller in this thesis contains a fixed pitch propeller, that is driven by an electric motor through a shaft and a gear box. The figure 2.6 illustrated the block

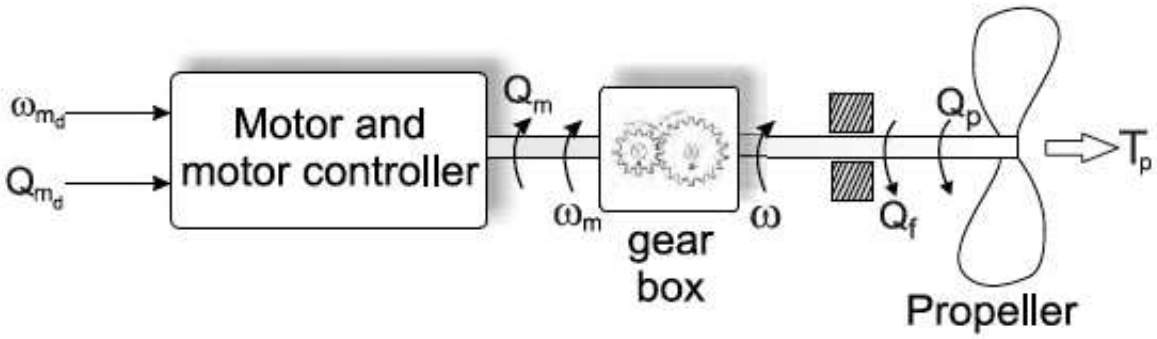


Figure 2.6: Block diagram of propeller system [165]

diagram of the propeller system. The parameters in the demonstrated block diagram are defined as follows:

- Q_m, is the motor torque applied to the shaft.
- ω_m, is the motor shaft angular speed.
- ω, is the propeller angular speed. The value of the ω is affected by the load due to the rotation of the blades in the water.
- $R_{gb} = \frac{\omega_m}{\omega}$, is the gear ratio.

- Q_p and Q_f are the propeller torque and the shaft friction torque respectively.
- T_p, is the thrust produced by the propeller.
- Q_{m_d} and ω_{m_d} are the desired motor torque and desired motor shaft speed respectively.

The motor controller is in charge of the shaft speed or the motor torque regulation.

2.8.2 Propeller Thrust and Torque Modeling

For a fixed pitch propeller, the shaft torque Q and force T (thrust) depend on the forward speed of the AUV, the advance speed u_a (ambient water speed), and the propeller rate n. In addition, other dynamic effects due to unsteady flows will influence the propeller thrust and torque. As in this thesis it is considered that the AUV is fully submerged the other dynamic effects are neglected. The thrust and torque model of the propeller is presented in detail in following section [171].

2.8.2.1 Quasi-Steady Thrust and Torque

In this work the thrust and torque models are approximated based on quasi-steady representation [171]. Quasi-steady modeling of thrust and torque are usually done in terms of lift and drag curves which are transformed to thrust and torque by using the angle of incidence. In this representation, the thrust and torque of the propeller are written as follows:

$$T_p = \rho D^4 K_T(J_0) n \left|n\right| \tag{2.8.1}$$

$$Q_p = \rho D^5 K_Q(J_0) n \left|n\right| \tag{2.8.2}$$

where ρ denotes to density of water, D is the propeller diameter, n is the propeller shaft speed, $J(0) = \frac{u_a}{nD}$ is the advance number and K_T and K_Q are the non-dimensional thrust and torque coefficients which can be calculated according to the following equations:

$$K_T(J_0) = \frac{T_p}{\rho D^4 n \left|n\right|} \tag{2.8.3}$$

$$K_Q(J_0) = \frac{Q_p}{\rho D^5 n \left|n\right|} \tag{2.8.4}$$

The numerical expressions for the aforementioned coefficients are obtained through open water tests.

During the normal operation of the propeller the AUV is moving, thus the water incident on the propeller have a velocity which is called ambient water velocity (u_a). This ambient water velocity is defined as follows:

$$u_a = (1 - w)u \tag{2.8.5}$$

where w is the wake fraction number that is typically chosen as 0.1 and 0.4 [1] and the u is the surge velocity of the AUV.

Therefore the quasi-steady model for estimating the thrust and torque produced by the propeller during operation is written as:

$$T_p = T_{n|n|} n \left|n\right| - T_{|n|u_a} \left|n\right| u_a \tag{2.8.6}$$

$$Q_p = Q_{n|n|} n \left|n\right| - Q_{|n|u_a} \left|n\right| u_a \tag{2.8.7}$$

where n is the propeller rotational speed, $T_{n|n|}$, $T_{|n|u_a}$, $Q_{n|n|}$ and $Q_{|n|u_a}$ are the propeller coefficients that can be calculated based on following equation:

$$Q_{n|n|} = \rho D^5 \beta_2, \quad T_{n|n|} = \rho D^4 \alpha_2, \quad Q_{|n|u_a} = \rho D^4 \beta_1, \quad T_{|n|u_a} = \rho D^3 \alpha_2 \tag{2.8.8}$$

where the α and β values are nondenominational constants. More details of this method can be found in [1].

2.8.3 Electric Motor Dynamics

Most thruster systems are driven by small DC motors designed for underwater operation condition. The dynamic of speed control DC motor can be written as [1]:

$$L_a \frac{di_a}{dt} = -R_a i_a - 2\pi K_M n + V_a \tag{2.8.9}$$

$$2\pi J_m \frac{dn}{dt} = K_M i_a - Q_p \tag{2.8.10}$$

$$Q_m = K_M i_a \tag{2.8.11}$$

where L_a is the armature inductance, R_a is the armature resistance, V_a is the armature voltage, K_M is the motor torque constant, Q_m is the motor torque, J_m is the moment of inertia of motor and thruster, n is the velocity of the motor in revolutions per second and the Q_p is the load from the propeller.

2.8.4 Propeller Shaft Dynamics

In the propeller system the motor connects to the propeller using a rigid shaft and a gear-box with the gear ratio of R_{gb}. The factor affect the shaft is the friction torque, $Q_f(w)$, assuming to depend on the shaft speed. The shaft dynamic can be presented as follows [165]:

$$J_m \dot{\omega} = R_{gb} Q_m - Q_p - Q_f(\omega) \tag{2.8.12}$$

where J_m is the total moment of inertia containing the shaft, the gear-box and the propeller. In general, the shaft moment of inertia should include the effect of the hydrodynamic added mass proportional to $\dot{\omega}$ which result in a time varying moment

of inertia. In the presented work, the added mass has been neglected by reason of the fact that its effect appeared to be not very significant and challenging to model.

According to the laboratory experiments, the friction torque has been modeled as [165]:

$$Q_f(\omega) = k_{f_1} arctan(\frac{\omega}{\epsilon}) + k_{f_2}\omega + k_{f_3} arctan(k_{f_4}\omega) \tag{2.8.13}$$

where k_{f_1}, k_{f_2}, k_{f_3} and ϵ are constants and positive that related to specification of thruster.

2.8.5 Propeller Simulation Model

According to the aforementioned sections the simulation model of the propeller is illustrated in Figure 2.7.

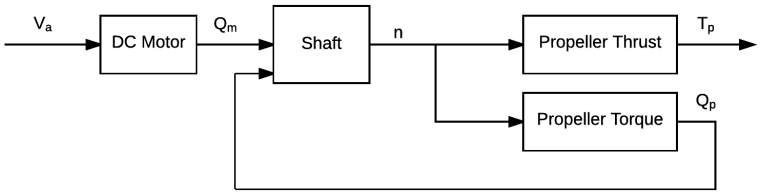


Figure 2.7: Simulation model of the propeller.

As it can be seen from the Figure 2.7 the thrust (T_p) and torque (Q_p) of the propeller depends on the shaft speed and can be calculated from equations (2.8.6) and (2.8.7) respectively. These are applied to the AUV dynamics as it is shown in following section.

2.9 The Developed Model

In this section the design of the closed-loop controller of the modeled AUVs are presented. The main elements of the closed-loop control system is presented in Figure

2.8.

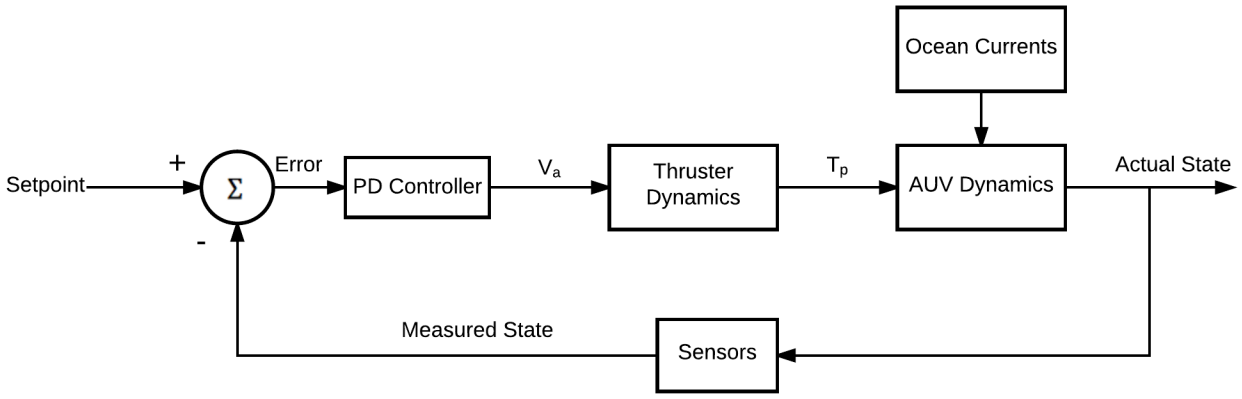


Figure 2.8: General closed-loop control system.

The above mentioned model is shown in Figure 2.9 with more details as follows:

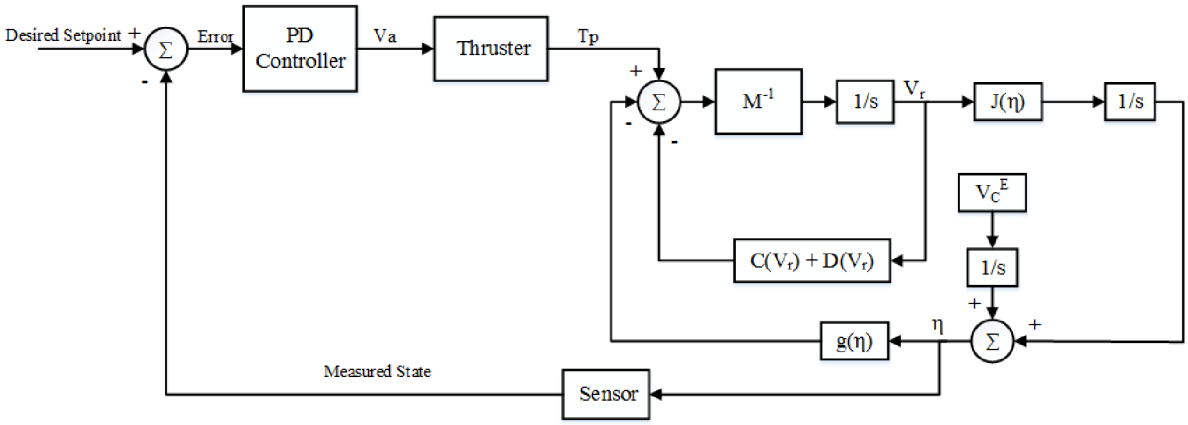


Figure 2.9: Block diagram of the developed model.

2.10 Possible Faults in AUVs

2.10.1 Sensor Failures

The underwater vehicles requires the information about their environment (e.g. existence of the obstacles), and also their position and velocity measurements, therefore

they are equipped with sensors. Since a single sensor is not capable of providing all of these information and measurements the sensor fusion technique is utilized. Kalman filtering is one of the most common approaches in sensor fusion that provides the required variables for the controller. The sensors in an AUV are mentioned as follows [101]:

- IMU (Inertial Measurement Unit): This sensor is in charge of providing information about the vehicle's linear acceleration and angular velocity through a combination of accelerometers, magnetometers and gyroscopes.
- Depth Sensor: It determines the depth of AUV by measuring the water pressure.
- Altitude and frontal sonars: They are applied to indicate the existence of obstacles and the distance from the sea bottom.
- Ground Speed Sonar: It measures the linear velocity of the vehicle with reference to the ground.
- Current meter: It provides the relative measurements between velocity vehicle and water.
- GNSS (Global Navigation Satellite System): It is utilized to reset the drift error of the IMU and localize exactly the vehicle; it works only at the surface.
- Compass: It gives the vehicle yaw.
- Baseline Acoustic: It gives exact localization of the vehicle in a specific range of underwater environment by assistance of one or more transmitters.
- Vision system: It can be utilized for tracking the structures like pipelines.

The types of sensors that are utilized in an AUV depends on the application of it. In each of the aforementioned sensors the faults can be occurred. The output zeroing or external disturbances can be considered as a failure in sensors of the AUV.

2.10.2 Actuator Failures

Thrusters are the main actuators in most underwater vehicles. Thus, thrusters are considered as one of the common and important of fault sources in AUVs. In a 6 degree of freedom (DOF) AUV, at least 6 thrusters are applied to generate 3 linearly independent translations and 3 linearly independent rotations. Generally, more than 6 thrusters are used in a 6 DOF AUV to provide a redundancy that makes the AUVs capable of fulfilling the mission in the presence of thruster failures. The actuators of the underwater vehicles are expressed in below [102]:

- Azimuth thrusters: Thruster units that can be rotated an angle α about the z-axis during the mission and produce two force components (F_x, F_y) in the horizontal plane.

- Fixed direction (non-rotable) thrusters: In contrast to azimuth thrusters, where an angle α can vary with time, fixed direction thrusters are featured with a fixed angle $\alpha = \alpha_0$. In other words, the orientation of these types of thrusters is fixed ahead and cannot be changed throughout the mission.

- Control surfaces: Control surfaces can be placed at different locations in AUVs to provide lift and drag forces. For instance fins are mounted for diving, rolling and pitching, rudders are installed for steering, etc.

The common faults in thruster of an AUV are listed as following:

- Thruster blocking: It happens when a solid body is between the propeller blades

or for the rope entanglements [103]. This fault can be detected by observing the current required by the thruster.

- Flooded thruster: A thruster flooded with water has been monitored during a Romeo's mission [104]. This failure leads to an electrical dispersion causing an increasing blade rotation velocity; therefore the force of the thruster become higher than the desired one.

- Fin stuck or lost: This failure can result in loss of steering ability explaining in [105] by means of simple numerical simulations. In addition, it can lead to other issues namely intermittent functioning or a non-null offset [103].

- Rotor failure: A possible result of different failures of the thrusters is the zeroing of the blade rotation that causes the thruster not to work. This failure has been observed in several mission experiments with ODIN [106 - 109], RAUVER [110] and Roby 2 [111] and Romeo [104].

- Hardware-software failure: Crashes in the hardware or software are considered as a fault in AUVs. In order to deal with this issues the redundancy techniques are applied [103, 112].

In this thesis three fault scenarios with various severities are considered namely as, thruster blocking, flooded thruster and loss of effectiveness in rotor. It should be noticed that the low severity faults in the thruster of the AUV does not make the topology of the formation unstable but it degrade the efficiency of the formation in fulfilling the mission, while the high severity faults can cause the insatiability in the formation. Therefore, existence of the reliable and autonomous fault detection, isolation and identification is essential.

2.10.3 Other Failures

Besides the faults that can occur in sensors and thrusters, the failures in other subsystems such as power systems, communication modules and payloads may result in termination of the mission [103].

2.11 Conclusion

In this chapter, the model of the dynamic neural network that is used in this thesis have been explained fully. The extended dynamic back-propagation algorithm is presented. The evolutionary algorithms and its principles have been provided. Different techniques regarding to evolving ANN with EA have and the evolutionary artificial neural networks been explained. GA and its operators have been completely described. The dynamics of the AUV and the architecture of the formation control of multiple AUVs have been provided. Finally, the mathematical model of the propulsion system of the AUV and the possible faults in AUVs have been explained.

Chapter 3

Agent-Level and Formation-Level Fault Detection Strategies

Health monitoring in formation of multiple AUVs has an important and critical role in their missions. The occurrence of a fault in thrusters of the AUVs in a formation can degrade the efficiency and reliability of the formation. Therefore, existence of the autonomous and reliable fault detection, isolation and identification system is necessary. It is obvious that formation of small AUVs can perform the same duties of a single large AUV when the coordination of those small AUVs fulfills the mission's requirements. Hence, detection of faults in thrusters of the AUV which could result in loss of coordination is highly desirable. In systems with high complexity like AUVs, existence of an intelligent and autonomous fault diagnosis system with high degree of accuracy and precision is essential. The mathematical models of the system are essential to achieve the desirable level of precision and accuracy in fault diagnosis scheme. Developing the accurate model for complex nonlinear systems like AUVs and its components can be challenging. Considering aforementioned limitations in this chapter we propose fault detection schemes for the thruster of the AUV based on neural networks. In this method, the dynamic neural network is utilized as a

nonlinear observer for fault diagnosis where genetic algorithm (GA) is employed to train the dynamic neural network parameters. In this work an agent-level fault detection (ALFD) and formation-level fault detection (FLFD) approaches for formation of multiple AUVs are developed. In our ALFD scheme, the absolute measurements of AUV are used for fault detection purpose however in our proposed FLFD scheme, both relative and absolute measurements are considered as diagnostic signals. In both ALFD and FLFD schemes, dynamic neural networks are employed to detect faults in the AUV.

3.1 Topology of the Formation

Different network topologies can be roughly categorized into groups such as, ring, line, fully connected, trees and bus. In this work the formation of AUVs includes four AUVs having bidirectional ring topology which are controlled by consensus based virtual structure controller that is fully explained in Chapter 2. The proposed method in this thesis can be investigated for other topologies as a future work. In this work, it is assumed that the four AUVs formation evolves as a rigid body and the formation shape is preserved, and in the healthy situation each AUV maintains a fixed relative orientation within the formation throughout the maneuvers. In this architecture each AUV receives information from its two adjacent neighbors as it is depicted in Figure 3.1.

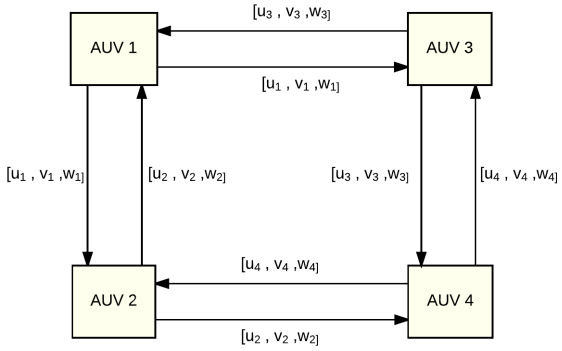


Figure 3.1: Formation of 4 AUVs.

3.2 Fault Detection Methodology

3.2.1 Proposed Agent-Level Fault Detection Scheme

In our proposed agent-level fault detection scheme, dynamic neural network (DNN) is used for AUV fault detection. The structure of the proposed agent-level fault detection scheme is depicted in Figure 3.2. In agent-level fault detection scheme when

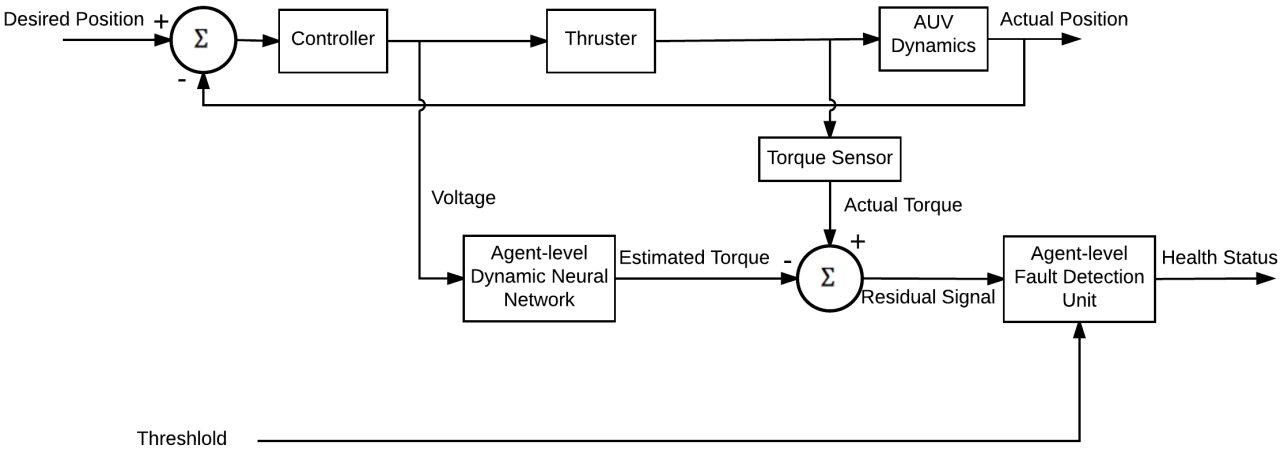


Figure 3.2: Structure of the agent-level fault detection scheme.

a fault injected in thruster of the AUV , the residual from estimated output (Dynamic

neural network) and actual torque (Thruster) is compared with the threshold in the agent-level fault detection unit to indicate the health status of the monitored system. As it can seen from Figure 3.2 in order to measure the actual torque of the thruster the torque sensor is used. Some practical torque sensors named as Vibrac series and ATI Mini-45 are presented in [181, 182].

It should be noticed that the agent-level dynamic neural network in Figure 3.2 is trained and validated with proper data. The training procedure of the dynamic neural network for ALFD scheme is fully explained in section 3.3.1.

3.2.2 Proposed Formation-Level Fault Detection Scheme

The formation-level fault detection is unique to multi platform missions. Each AUV in the formation are considered as different components of the formation system. Hence, at this level, fault detection is the binary decision determining whether or not any fault exists in the formation components. At formation-level, fault detection is based on both relative and absolute information. In this approach, the AUV_i has two adjacent neighbors namely, AUV_{i-1} and AUV_{i+1}. The AUV_i is called a neighbor of AUV_{i-1} , if AUV_i receives information from AUV_{i-1} and vice versa. In other words, the proposed consensus-based virtual structure formation has bi-directional ring topology structure.

In order to detect the fault in a formation, in the first step the faulty AUV must be recognized. In our proposed formation-level fault detection algorithm, two dynamic neural networks in each AUV are employed to detect the fault in formation. It should be considered before using the DNNs in the fault detection scheme, each of them is trained with one relative and one absolute information. The training and testing procedure of the dynamic neural networks for FLFD scheme is fully explained in section 3.3.1.

Figure 3.3 demonstrates these two DNNs in the fault detection unit in AUV_i. In this method when a fault occurred in one of the AUVs, for instance AUV_i, both residuals corresponding to each of the DNNs in AUV_i passed the threshold, therefore in this case the AUV_i considered as a faulty AUV in a formation. Meanwhile, when the fault occurred in AUV_i, in AUV_{i-1} and AUV_{i+1} just one of the DNN which is related to AUV_i shows the occurrence of the fault and in other DNN the residual did not pass the threshold. Hence in our proposed algorithm, when the residuals of both DNNs in fault detection unit of the AUV passed the threshold, that AUV considered as a faulty AUV.

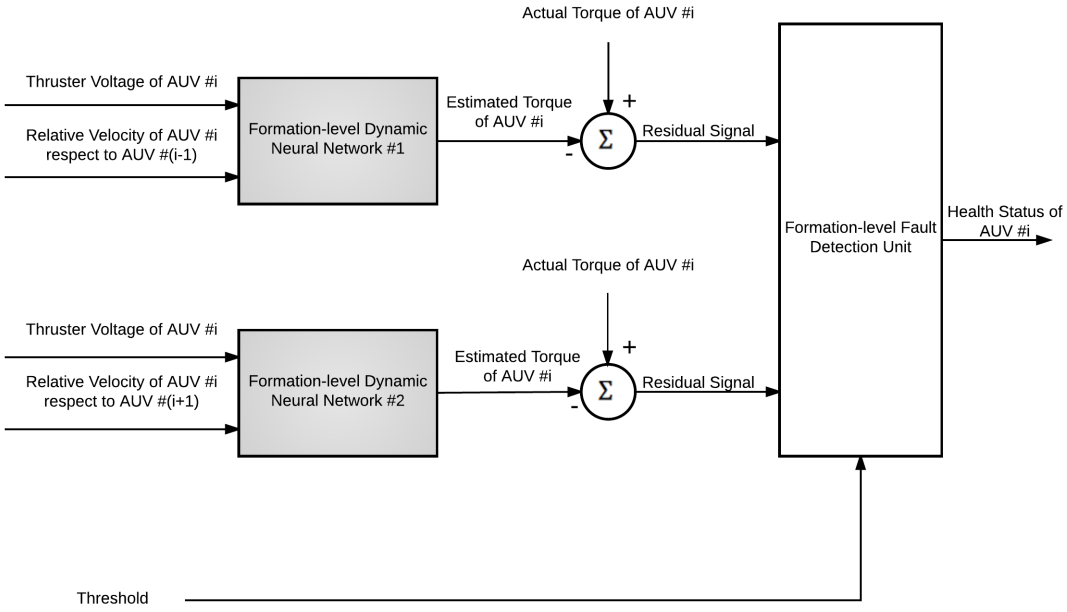


Figure 3.3: Fault detection unit in AUV_i

3.3 Training DNN with Genetic Algorithm

The DNNs in both agent-level and formation-level fault detection schemes are trained with genetic algorithm. The GA is applied to adjust all the parameters of the DNN including connection weights, feedback and feedforward filter parameters and the

activation function slopes. The steps of the proposed algorithm are explained in details as follows.

- First step (Initial Population)

  The initial population consists of N_{pop} chromosomes that are selected at random. According to the proposed initialization method no replication is permitted in the population. As a result each time that a new chromosome is generated it is compared with the ones produced before, if it has a replication it is neglected and a new one is generated if not it becomes a member of the first generation. Since the encoding method that is used in this thesis is based on the real-valued encoding, therefore the genes of the chromosomes are real numbers in value that are generated randomly. Each of the chromosomes includes N_{var} variables which represent all the connection weights, feedback filter parameters, feedforward filter parameters and slope parameters. If we assume that the DNN has the total of N connection weights, M feedback filter parameters, L feedforward filter parameters and P slope parameters, then the l^{th} chromosome of the population can be written as:

$$Chromosome_l = [w_{1l}, \cdots, w_{Nl}, a_{1l}, \cdots, a_{Ml}, b_{1l}, \cdots, b_{Ll}, g_{1l}, \cdots, g_{Pl}] \quad (3.3.1)$$

  where the w is the connection weight, a is the feedback filter parameter, b denotes the feedforward filter parameter and the g is the slope parameter.

  Considering that the initial population has N_{pop} chromosomes, the matrix of the initial population is calculated as follows:

$$Initial\ Population\ Matrix = rand(N_{pop}, N_{var}) \quad (3.3.2)$$

- Second step (Fitness Function)

  The fitness function is responsible for evaluating the optimality of the chromosomes. In this work root mean squared error (RMSE) is used as fitness function in our genetic algorithm which is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2} \tag{3.3.3}$$

  The proposed fitness function is the error between the desired output and the current output. In equation (3.3.3), $\widehat{y_i}$ and y_i denote the estimated output and desired output respectively and the n is the number of input data.

- Third step (Termination Criterion)

  In the proposed method the GA evolves from one generation to another till the termination criterion is achieved. The termination criterion we have chosen for the algorithm is related with root mean squared error (RMSE). If the RMSE is less than the desired value the algorithm stops and returns the optimal parameters of the DNN.

- Forth step (Selection)

  In this stage, the algorithm decides which chromosomes in the initial population are appropriate to survive and probably reproduce offspring in the next generation. The proposed selection method in this work is inspired by natural selection where fitness values associated with the chromosomes are ranked from highest values to lowest values. Then the chromosomes with the most fitness survive and the least fit chromosomes are eliminated and replace by the new generated offsprings. The natural selection procedure occur at each iteration of the algorithm in order to allow the population of chromosomes to develop from one generation to another for finding the most fit chromosomes as determined

by the fitness function. In this thesis in each iteration 50% of the chromosomes with highest rank are survived to the next generation and the remaining are eliminated.

- Fifth step (Crossover)

  In this step, firstly two chromosomes from the population are chosen to produce the offsprings. It is assumed that the i^{th} and j^{th} chromosomes are selected. Secondly, a DNN parameter in both chromosomes is selected to be the crossover point. The crossover point selection is written as following:

$$\alpha = roundup\left\{random * N_{var}\right\} \tag{3.3.4}$$

The α in equation (3.3.4) denotes the randomly selected parameter of the DNN in both chromosomes. The randomly selected parameter can be a connection weight, feedforward filter parameter, feedback filter parameter or slope parameter. As an example it is assumed that a connection weight is chosen. Therefore, chromosomes that are chosen for generating new offsprings can be written as:

$$\begin{aligned} Chromosome_i &= [w_{1i}, \cdots, w_{\alpha i}, \cdots, w_{Ni}, a_{1i}, \cdots, a_{Mi}, b_{1i}, \cdots, b_{Li}, g_{1i}, \cdots, g_{Pi}] \\ Chromosome_j &= [w_{1j}, \cdots, w_{\alpha j}, \cdots, w_{Nj}, a_{1j}, \cdots, a_{Mj}, b_{1j}, \cdots, b_{Lj}, g_{1j}, \cdots, g_{Pj}] \end{aligned} \tag{3.3.5}$$

In the next step, the selected parameters in each chromosomes (i.e $w_{\alpha i}$ and $w_{\alpha j}$) are replaced with the new values based on the following equation:

$$\begin{aligned} w_{newi} &= w_{\alpha i} - \beta[w_{\alpha i} - w_{\alpha j}] \\ w_{newj} &= w_{\alpha j} - \beta[w_{\alpha i} - w_{\alpha j}] \end{aligned} \tag{3.3.6}$$

where β is also a random value between 0 and 1. The final step is to produce the offsprings for the population by replacing the new connection weights in the

chromosomes and swapping the connection weights of the two selected chromosomes at the crossover point. The new connection weight values are named as $w_{\alpha 1}$ and $w_{\alpha 2}$ and replaced with old values. The generated offsprings are expressed as follows:

$$\begin{aligned} offspring_1 &= [w_{1i}, \cdots, w_{\alpha 1}, \cdots w_{Nj}, a_{1j}, \cdots, a_{Mj}, b_{1j}, \cdots, b_{Lj}, g_{1j}, \cdots, g_{Pj}] \\ offspring_2 &= [w_{1j}, w_{2j}, \cdots, w_{\alpha 2}, \cdots w_{Ni}, a_{1i}, \cdots, a_{Mi}, b_{1i}, \cdots, b_{Li}, g_{1i}, \cdots, g_{Pi}] \end{aligned} \tag{3.3.7}$$

In the aforementioned crossover technique, if the first variable of the chromosomes is selected, then only the variables to the right of the selected variable are swapped and if the last variable of the chromosomes is selected, then only the variables to the left of the selected variable are swapped.

- Mutation

  The mutation is considered as one of the GA operators that prevent the GA from converging too quickly into one region of the cost surface. As a result, the GA can be trapped in the area of the local minima instead of the global minima. To avoid this problem of the fast convergence, the mutation is applied in some of the variables of the chromosomes to force the algorithm to search other areas of the cost surface. In the GA, in the first step the mutation rate is chosen. In the next stage random numbers are chosen to select the rows and columns of the variables to be mutated. Then, a mutated variable is replaced by a new random variable. In order to find the total number of mutations, the mutation rate should multiply by the whole number of variables that can be mutated in the population.

The flow chart of our proposed method is illustrated in Figure 3.4 and the proposed genetic algorithm parameters that are applied in this thesis are mentioned in Table

3.6.

3.3.1 Training Phase Results in ALFD Scheme

In this thesis, the DNN in agent-level fault detection scheme is trained with different data sets which are obtained from healthy operating condition of the AUV from the simulation model. The training is done for different data sets, where each data set includes 4000 samples. The voltage of the thruster is considered as an input the torque is considered as the output. In order to provide a model that is similar to a practical AUV, all data for training are considered under presence of the measurement noise. These data are normalized in the range of [0,1] before the training process begins. The identification models corresponding to ALFD training phase is depicted in Figure 3.5.

As it is shown in Figure 3.5 the DNN in ALFD scheme requires one input to construct an acceptable identification model and produce the output.

As it is mentioned, in the first step the structure of the DNN has been decided. Structure of the network includes, the number of layers, the number of neurons in each layer, the order of the IIR filters in each neuron. In this work, it has been observed that utilizing two hidden layers for the DNN in ALFD scheme, increases the performance of the network in comparison with using one hidden layer. The training procedure is started with a small network structure, then the number of neurons and hidden layers are increased till the optimum structure of the DNN is achieved. The DNN parameters including weights, activation function slopes and filters feedforward parameter matrix, are initialized with small random values and the IIR filter's denominator coefficients are initialized to zeros to assure stable learning. The order of IIR filters in ALFD scheme are set to 2 since the higher orders do not enhance the performance of the network and increases computational complexity.

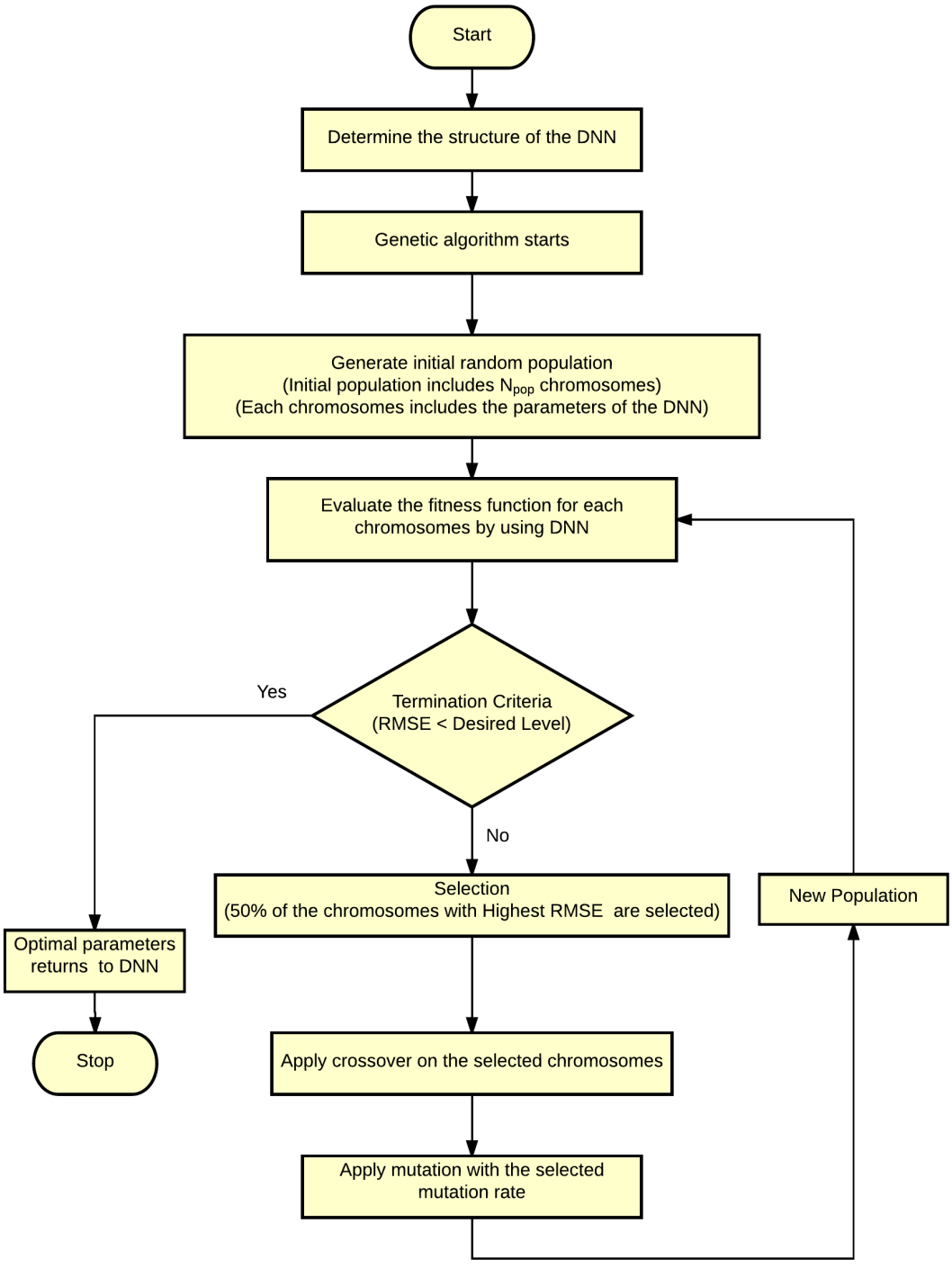


Figure 3.4: Flow chart of the proposed method.

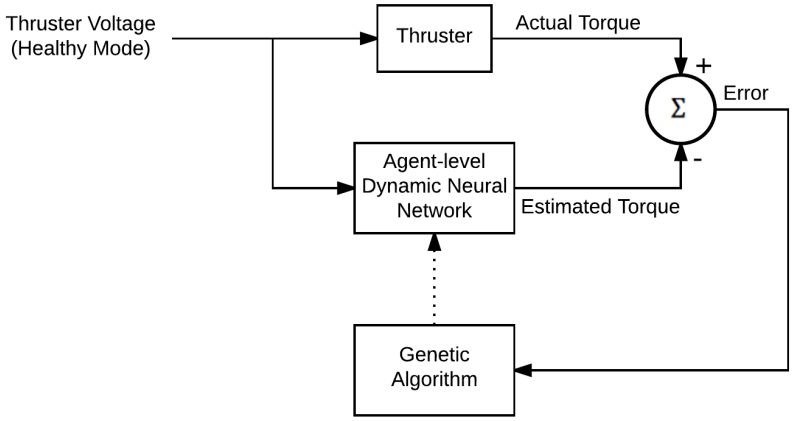


Figure 3.5: Identification model in training phase of ALFD scheme.

The optimum structure of the network and the specification of the proposed DNN for agent-level fault detection (ALFD) is shown in Table 3.1.

Table 3.1: Dynamic neural network specifications in ALFD scheme.

Structure of the Network	1-6-1-1
IIR Filter Order	2^{nd} Order
$F(.)$ Hidden Layers	Hyperbolic Tangent Sigmoid
$F(.)$ Output Layer	Linear

In the aforementioned table the four successive numbers in the structure of the network are the number of neurons in the input layer, first hidden layer, second hidden layer and output layer respectively.

As it is expressed in training phase the variables of each chromosomes in the population includes, the connection weights, feedback filter parameters, feedforward parameters and slope parameters of the DNN. The DNN in our ALFD scheme has 61 parameters that must be optimized with GA which are indicated in Table 3.2.

In addition, the genetic algorithm parameters which is applied in agent-level fault detection scheme are expressed in 3.3.

The performance of the dynamic neural network in one of the AUVs during the training process in ALFD scheme is shown in Figure 3.6. The average value of the

Table 3.2: Parameters of DNN in ALFD scheme that are optimized with GA.

Parameters	Number of Parameters
Weights of the hidden layer 1	$W_1(N(1), K) = W_1(6, 1) = 6$
Weights of the hidden layer 2	$W_2(N(2), N(1)) = W_2(1, 6) = 6$
Weights of the output layer	$W_3(N(3), N(2)) = W_3(1, 1) = 1$
Filter parameters A_1	$A_1(N(1), D) = A_1(6, 2) = 12$
Filter parameters A_2	$A_2(N(2), D) = A_2(1, 2) = 2$
Filter parameters A_3	$A_3(N(3), D) = A_3(1, 2) = 2$
Filter parameters B_1	$B_1(N(1), D + 1) = B_1(6, 3) = 18$
Filter parameters B_2	$B_2(N(2), D + 1) = B_2(1, 3) = 3$
Filter parameters B_3	$B_3(N(3), D + 1) = B_3(1, 3) = 3$
Filter parameters g_1	$g_1(N(1), 1) = g_1(6, 1) = 6$
Filter parameters g_2	$g_2(N(2), 1) = g_2(1, 1) = 1$
Filter parameters g_3	$g_3(N(3), 1) = g_3(1, 1) = 1$

K = Number of inputs
D = Order of the IIR filter
N(1) = Number of neurons in the hidden layer 1
N(2) = Number of neurons in the hidden layer 2
N(3) = Number of neurons in output layer

Table 3.3: Genetic algorithm parameters.

Population size	20
Termination criterion	$RMSE < 0.04$
Mutation rate	0.25
Selection type	Natural Selection

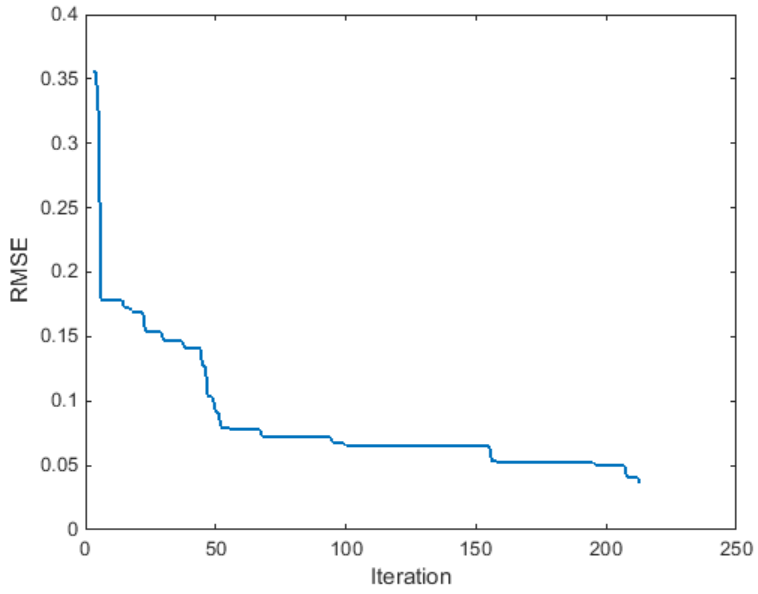


Figure 3.6: The performance of the dynamic neural network in training phase for ALFD scheme.

root mean squared error and its standard deviation in the training phase are 0.0415 and 0.00601 respectively, which is quite well.

3.3.2 Training Phase Results in FLFD Scheme

In this thesis, the two dynamic neural networks in each fault detection unit of the AUVs are trained with various data sets which includes thruster voltage and relative velocity as their input and the torque as the output. These data sets are obtained from healthy operating condition of the AUV from the simulation model. Each of the data set includes 4000 samples. In order to provide a model that is similar to a practical AUV, all data for training are considered under presence of the measurement noise. These data are normalized in the range of [0,1] before the training process begins. The identification models corresponding to FLFD training phase is depicted in Figure 3.7.

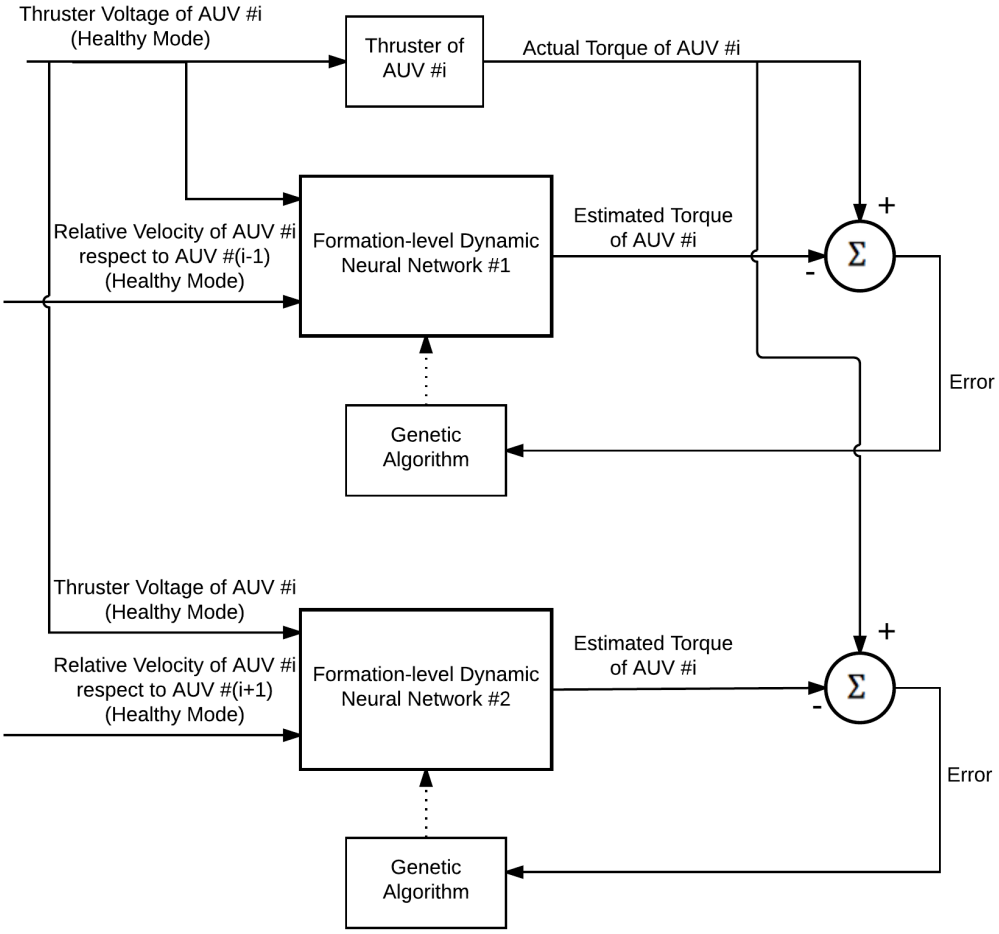


Figure 3.7: Identification model in training phase of FLFD scheme.

According to Figure 3.7 each of the DNNs need two inputs to create the identification model.

As it is mentioned, in the first step the structure of the DNN has been decided. Structure of the network includes, the number of layers, the number of neurons in each layer, the order of the IIR filters in each neuron. In this work, it has been observed that utilizing two hidden layers for the DNNs in FLFD scheme, increases the performance of the network in comparison with using one hidden layer. The training procedure is started with a small network structure, then the number of neurons and hidden layers are increased till the optimum structure of the DNNs is achieved. The DNNs parameters including weights, activation function slopes and filters feedforward parameter matrix, are initialized with small random values and the IIR filter's denominator coefficients are initialized to zeros to assure stable learning. The order of IIR filters in ALFD scheme are set to 2 since the higher orders do not enhance the performance of the network and increases computational complexity.

The optimum structure of the network and the specification of the proposed DNN for formation-level fault detection (FLFD) is shown in Table 3.4.

Table 3.4: Dynamic neural network specifications in FLFD scheme.

Structure of the Network	2-4-1-1
IIR Filter Order	2^{nd} Order
$F(.)$ Hidden Layers	Hyperbolic Tangent Sigmoid
$F(.)$ Output Layer	Linear

In the aforementioned table the four successive numbers in the structure of the network are the number of neurons in the input layer, first hidden layer, second hidden layer and output layer respectively.

As it is expressed in training phase the variables of each chromosomes in the population includes, the connection weights, feedback filter parameters, feedforward parameters and slope parameters of the DNN. The DNN in our FLFD scheme has 49

parameters that must be optimized with GA which are indicated in Table 3.5.

Table 3.5: Parameters of DNNs in FLFD scheme that are optimized with GA.

Parameters	Number of Parameters
Weights of the hidden layer 1	$W_1(N(1), K) = W_1(4, 2) = 8$
Weights of the hidden layer 2	$W_2(N(2), N(1)) = W_2(1, 4) = 4$
Weights of the output layer	$W_3(N(3), N(2)) = W_3(1, 1) = 1$
Filter parameters A_1	$A_1(N(1), D) = A_1(4, 2) = 8$
Filter parameters A_2	$A_2(N(2), D) = A_2(1, 2) = 2$
Filter parameters A_3	$A_3(N(3), D) = A_3(1, 2) = 2$
Filter parameters B_1	$B_1(N(1), D + 1) = B_1(4, 3) = 12$
Filter parameters B_2	$B_2(N(2), D + 1) = B_2(1, 3) = 3$
Filter parameters B_3	$B_3(N(3), D + 1) = B_3(1, 3) = 3$
Filter parameters g_1	$g_1(N(1), 1) = g_1(4, 1) = 4$
Filter parameters g_2	$g_2(N(2), 1) = g_2(1, 1) = 1$
Filter parameters g_3	$g_3(N(3), 1) = g_3(1, 1) = 1$
K = Number of inputs	
D = Order of the IIR filter	
N(1) = Number of neurons in the hidden layer 1	
N(2) = Number of neurons in the hidden layer 2	
N(3) = Number of neurons in output layer	

In addition, the genetic algorithm parameters which id applied in formation-level fault detection scheme are expressed in 3.6

Table 3.6: Genetic algorithm parameters.

Population size	20
Termination criterion	$RMSE < 0.04$
Mutation rate	0.2
Selection type	Natural Selection

The performance of the dynamic neural network in one of the AUVs during the training process in FLFD scheme is shown in Figure 3.8.

The average value of the root mean squared error and its standard deviation in the training phase are 0.0392 and 0.0037 respectively, which is quite acceptable.

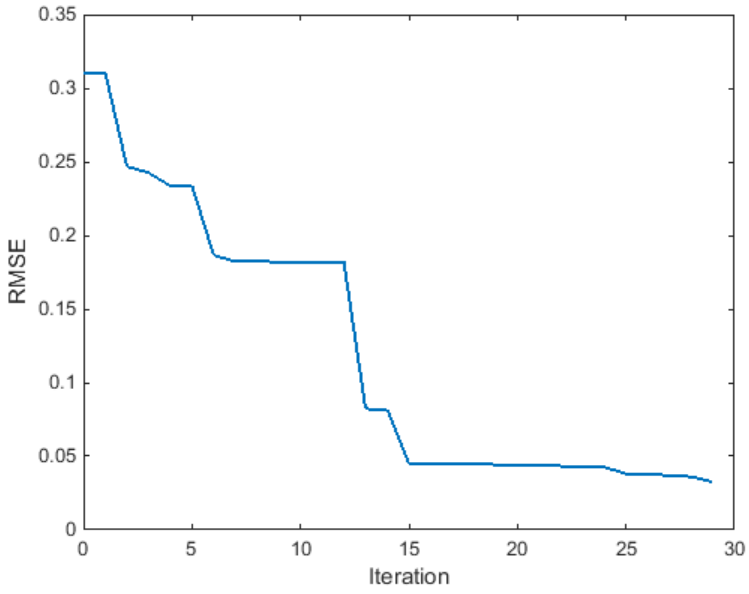


Figure 3.8: The performance of the two dynamic neural networks in training phase for FLFD scheme.

3.3.3 Comparison of EDBP and GA

In this section the performance of the extended dynamic back-propagation and genetic algorithm for training the dynamic neural network is compared. For both algorithm, the network architecture is, $N_{1-6-1-1}$, where the four successive numbers represent number of neurons in input layer, first hidden layer, second hidden layer and output layer respectively. The network parameters including weights, activation function slopes and filters feedforward parameter matrix, are initialized with small random values and the IIR filter's denominator coefficients are initialized to zeros to assure stable learning. The order of IIR filters are set to 2 since the higher orders do not enhance the performance of the network and increases computational complexity. The dynamic neural network in each of the methods is trained with 4000 samples. In order to have the identical training for comparing the GA and EDBP, the training data is normalized from 0 to 1.

In order to compare the performance of the dynamic neural network during training phase of the EDBP and GA, the RMSE of these two methods with respect to the 500, 1000, 2000 and 4000 number of iterations are depicted in Figures 3.9 to 3.12.

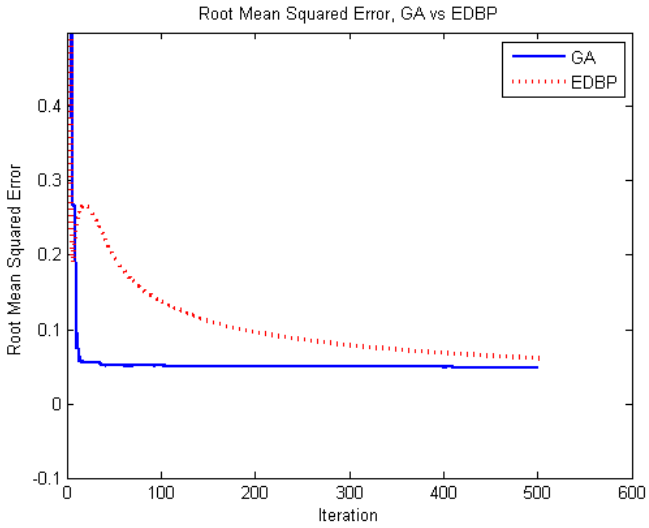


Figure 3.9: RMSE in training phase of EDBP and GA for 500 iterations.

The depicted curves of RMSE with respect to different number of iterations in Figures 3.9 to 3.12 indicate that the GA converge faster in comparison with the EDBP, in other words it provides better results in less number of iterations.

3.4 Threshold Determination

Comparing the actual and estimated outputs (i.e. system and DNN's output, respectively), the health status of the system can be evaluated. The threshold determines if the data set presented to the network corresponds to a healthy or faulty scenario. The threshold value is calculated by using collected healthy data. Due to apparent model uncertainties residuals always deviate from zero, dependent on the input signals. Hence, using an adaptive threshold instead of a fixed one, can improve the performance of a fault detection scheme significantly with respect to false alarm rate

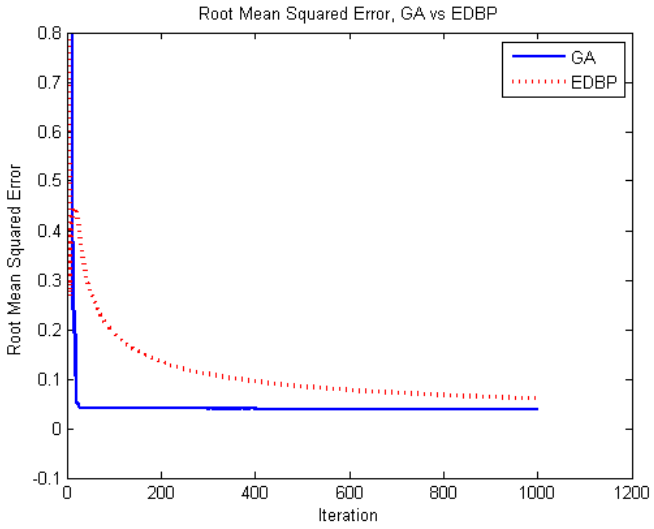


Figure 3.10: RMSE in training phase of EDBP and GA for 1000 iterations.

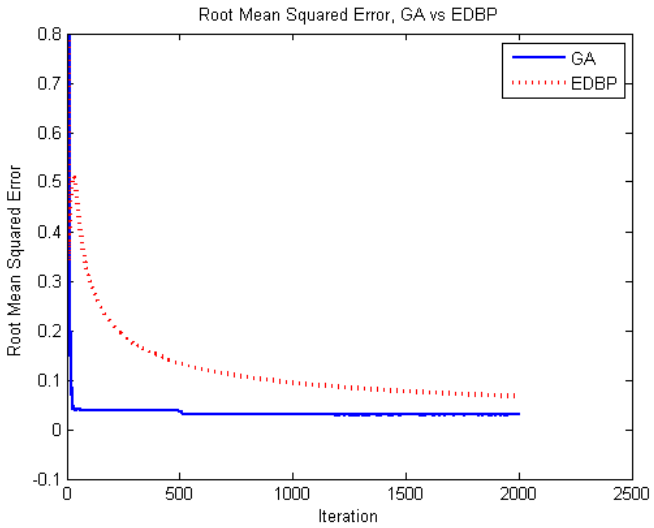


Figure 3.11: RMSE in training phase of EDBP and GA for 2000 iterations.

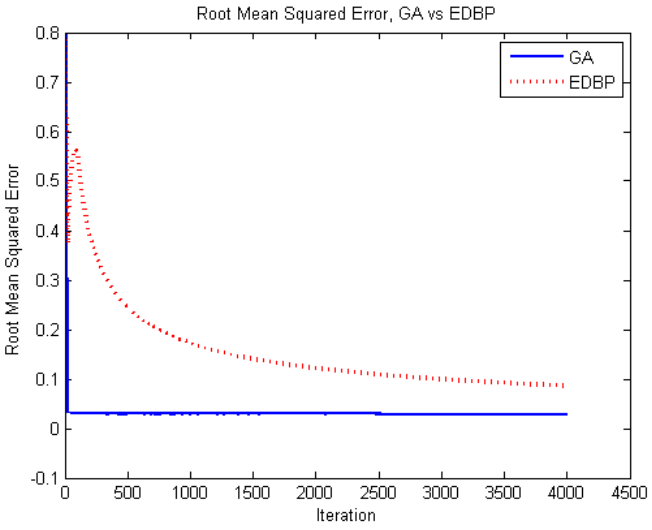


Figure 3.12: RMSE in training phase of EDBP and GA for 4000 iterations.

and the delay in detection [163]. The adaptive threshold for agent-level fault detection scheme and formation-level fault detection scheme are demonstrated in Figures 3.13 and 3.14.

As it is depicted in Figures 3.13 and 3.14, the adaptive threshold consist of a filter with lead-lag behavior which is driven by the input signals. This causes the filter output to be zero for steady-state inputs; otherwise it is a measure for the dynamic input excitation. In order to rectify this signal the absolute value is computed. A constant value is added due to the effects of measurement noise on the residual. A first-order low-pass filter can be applied in order to smooth the adaptive threshold.

In these figures K_1 and K_2 denote the sensitivity parameters that are used to adjust the thresholds, T_1, T_2, T_3 and T_4 are constants which are determined in the healthy operation condition of the AUV. The values of these parameters for ALFD and FLFD are expressed in Table 3.7.

It is important to note that due to noise in the AUV signal measurements, if smaller values for K_1 and K_2 is chosen, the false alarms increase in the monitored

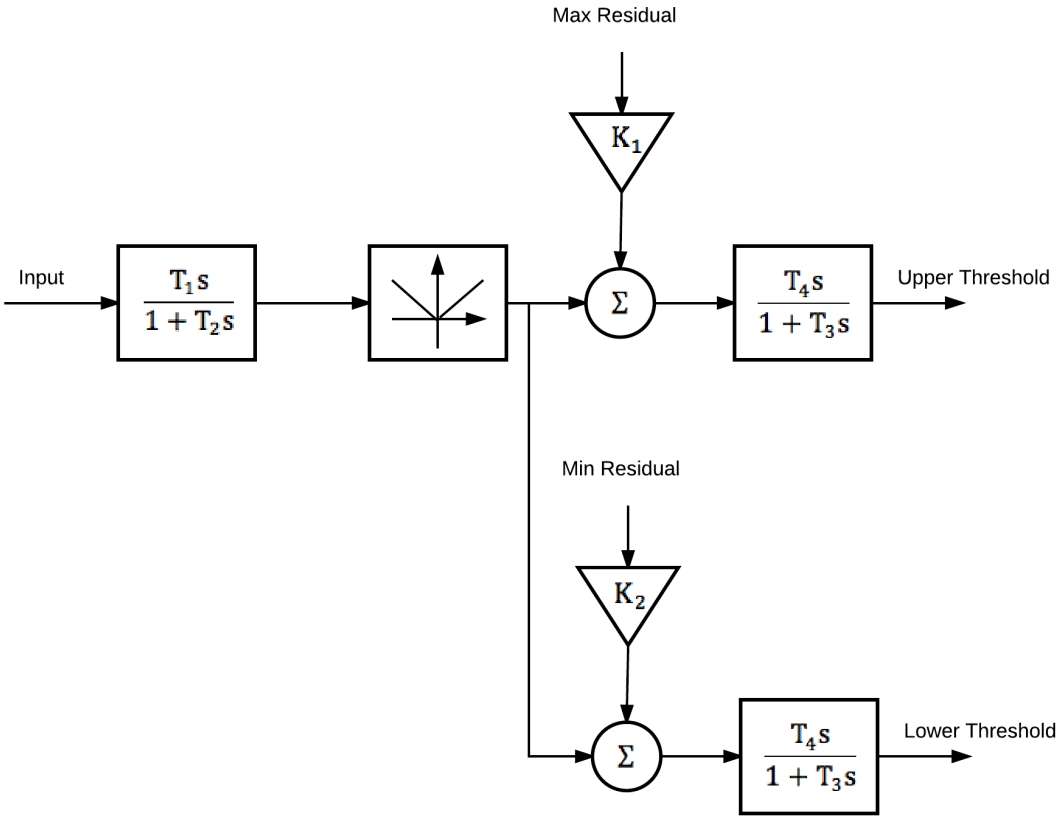
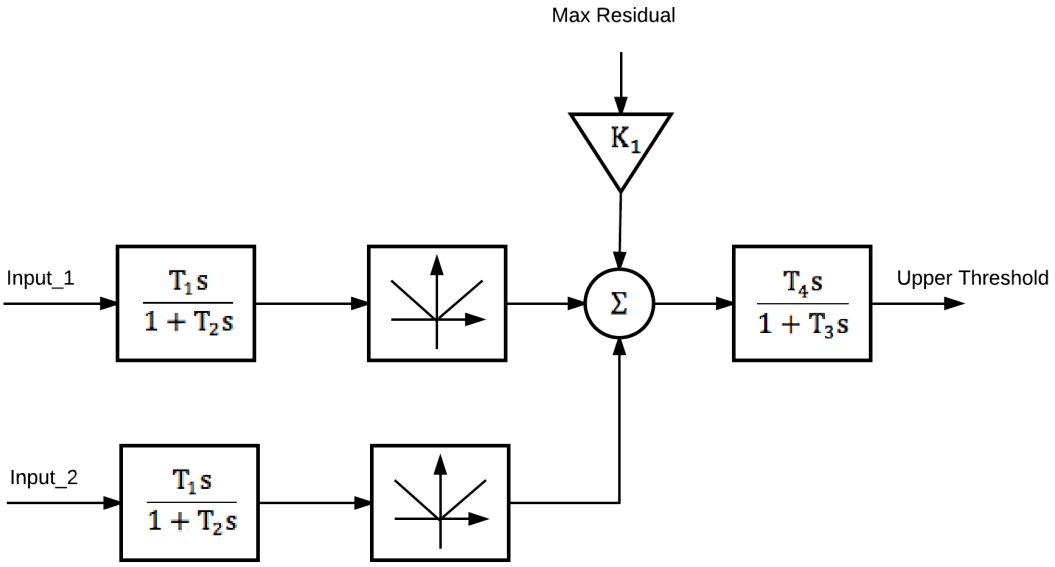


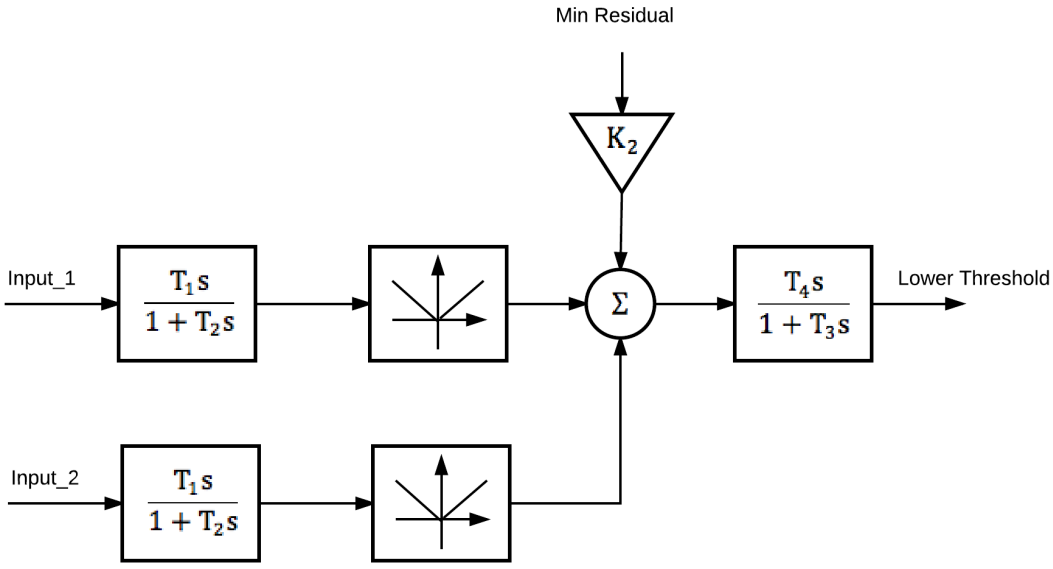
Figure 3.13: Adaptive threshold for agent-level fault detection scheme

Table 3.7: Threshold parameters for ALFD and FLFD.

ALFD parameters	FLFD parameters
$K_1 = 9.88$	$K_1 = 10$
$K_2 = 9.83$	$K_2 = 9.94$
$T_1 = 0.22$	$T_1 = 0.08$
$T_2 = 70$	$T_2 = 40$
$T_3 = 1.7$	$T_3 = 1.1$
$T_4 = 0.1$	$T_4 = 0.1$



(a) Upper threshold



(b) Lower threshold

Figure 3.14: Adaptive threshold for formation-level fault detection scheme

system, however larger values for these parameters can decrease the ability of detecting low severity faults in the thruster. In order to determine the threshold, the aforementioned parameters (K_1, K_2, T_1, T_2, T_3, T_4) are determined through comparing the residuals under various healthy scenarios. In the next step, the threshold are defined in real time. According to the values of the K_1 and K_2 that are calculated under the healthy scenario, two threshold named as upper threshold and lower threshold are produced to cover the residual signals and to detect the faults in the AUV. When the fault is injected in the thruster, if the residual signal passes the upper or lower threshold, it can be concluded that the fault is detected.

3.5 Characterization of Possible Fault Scenarios in Thruster of AUV

In order to be able to develop the fault detection algorithm and to inject faults in the AUV, the potential sources of faults in the thruster must be identified. Experimental experience with thruster in different AUVs missions has revealed that the potential failures may occur in thruster namely as thruster blocking, flooded thruster and rotor failure [179]. In this thesis the loss of effectiveness in rotor which is decreasing in blade rotation, is considered as one of the fault scenarios in the thruster of the AUV. The aforementioned faults are the most common faults in AUV's thruster [179]. For each fault scenario the proposed fault detection algorithm is performed to verify the capability of our proposed fault detection algorithm in the AUVs.

3.6 Fault Detection Logic

3.6.1 Agent-level Fault Detection Logic

The trained dynamic neural network is used to detect the thruster faults in the AUV. The DNN is applied to generate the residual signals. The difference between the actual torque of the thruster that is measured by torque sensors in AUV and the estimated torque that is gained from DNN is called residual signal. In the proposed fault detection scheme, the first moment that the residual exceeds the threshold lines after the fault injection, is considered as the fault detection time. After injecting the fault, the samples that exceed the threshold lines are considered as true faulty detection and the samples that stay inside the threshold lines are considered as the false healthy detection.

3.6.2 Formation-level Fault Detection Logic

The trained dynamic neural networks are used to detect the thruster faults in the AUV. Two DNNs in fault detection unit of the AUV are applied to generate the residual signals. The difference between the actual torque of the thruster that is measured by torque sensors in AUV and the estimated torques that are gained from DNNs is called residual signals. In the proposed formation-level fault detection scheme, the first moment that the residual signals exceed the threshold lines after the fault injection, is considered as the fault detection time. After injecting the fault, the samples that exceed the threshold lines are considered as true faulty detection and the samples that stay inside of the threshold lines are considered as the false healthy detection.

3.7 Simulation Results

The simulations performed in this thesis have been implemented in Matlab and Simulink. Using the mathematical model of the thruster that is presented in Chapter 2, we have simulated the thruster behavior for the generated torque in each AUVs. In practical the torque sensor can be used in the AUV to measure the torque. Although it increases system complexity and cost, but it provides most accurate and stable results [173]. It is important to note that in our simulations all fault validation results are obtained with a Gaussian random noise for all measurements. In addition, the ocean currents are considered as the environmental disturbances which the effect of it is inserted in the dynamic of the AUV which is fully explained in Chapter 2. The formation consists of four AUVs that have the bi-directional ring topology are controlled by using consensus-based virtual structure controller that is explained in Chapter 2. In this work, the data exchange and communication links between the AUVs are considered as bidirectional, which means that the DNNs in fault detection unit of each AUV use the relative velocity measurements of the AUV with respect to its two adjacent neighbors. In our simulations, we consider these four AUVs are located on a rectangular in order to inspect the underwater pipelines. Figure 3.1 demonstrates the formation of 4 identical AUVs. In our simulation, two different missions are considered for the AUVs. In the first mission (heading mission) it is assumed that the four AUVs have the $[\phi, \theta, \psi] = [0, 0, 0]$ as their initial attitude and their desired attitude is [0, 0, 10]. The initial position, $[x, y, z]$ of the 4 AUVs are, [0, 0, 30], [10, 0, 30], [10, 10, 30] and [0, 10, 30] respectively. Their desired position are [17, 100, 30], [27, 100, 30], [27, 110, 30] and [17, 110, 30] respectively. Their initial and desired angular velocity, $[p, q, r]$, of 4 AUVs are [0 ,0 ,0]. The initial velocity of them is 0.3 m/s and the desired velocity is 2.3 m/s.

In the second mission (depth mission) it is assumed that the four AUVs have the $[\phi, \theta, \psi]$= [0, 0, 0] as their initial attitude and their desired attitude is [0, 15, 0]. The initial position, $[x, y, z]$, of the 4 AUVs are, [0, 0, 30], [10, 0, 30], [10, 10, 30] and [0, 10, 30] respectively. Their desired position are [100, 0, 56], [110, 0, 56], [110, 10, 56] and [100, 10, 56] respectively. Their initial and desired angular velocity, $[p, q, r]$, of 4 AUVs are [0 ,0 ,0]. The initial velocity of them is 0.2 m/s and the desired velocity is 2.2 m/s. The process that AUV from its initial velocity reached its desired velocity is called surge mission. From the control point of view, the goal of formation control is that the the position, rotation, linear and angular velocities of each AUV track a set of desired state of the formation. Table 3.8 shows the expected settling time and tracking errors that are used to evaluate the performance of the formation control of AUVs. The simulations are carried out for 3000 sec (60000 Samples) of the thruster operations in AUVs. Furthermore, the gains and parameters of the controller that are used in formation controller are indicated in Table 3.9. Furthermore, the gains

Table 3.8: Expected settling time and tracking error in each AUV.

Variable	Settling time (sec)	Tracking error
V_x	100	1e-2 m/s
V_y	100	1e-2 m/s
V_z	100	1e-2 m/s
r_x	100	1e-3 m
r_y	100	1e-3 m
r_z	100	1e-3 m
q_1	100	1e-3
q_2	100	1e-3
q_3	100	1e-3
ω_x	100	1e-2 rad/s
ω_y	100	1e-2 rad/s
ω_z	100	1e-2 rad/s

and parameters of the controllers that are used in formation controller are indicated in Table 3.9.

Moreover, the permanent fault are injected in to the system after the formation

Table 3.9: Controller gains for each AUV and virtual structure.

Parameter	Value
K_G	$20I_{3\times3}$
D_S	$20I_{3\times3}$
K_S	$20I_{3\times3}$
K_F	$10I_{3\times3}$
K_r	$15I_{3\times3}$
k_v	$15I_{3\times3}$
k_q	$10I_{3\times3}$
K_ω	$10I_{3\times3}$
k_G	$10I_{3\times3}$
k_S	$10I_{3\times3}$

is held and they occur in the steady state condition of the AUV.

3.7.1 Agent-Level Fault Detection Analysis

3.7.1.1 Fault Detection Analysis under Thruster Blocking Fault Scenarios

In this fault scenario, low torque condition is considered as a fault in a thruster. In order to simulate this fault scenario , thruster torque is dropped by 1% to 12% from its nominal value. This permanent fault is injected to the thruster in the steady state condition at t = 2300 sec (sample # 46000). The residual signals corresponding to this fault scenario are illustrated in Figures 3.15 to 3.26.

According to the residual signals in Figures 3.15 to 3.26, it can be seen that:

- From 1% to 4% drop in thruster torque the residual does not pass the threshold, which means the ALFD scheme does not detect the fault.
- From 5% to 11% drop in thruster torque the residual passes the threshold which means according to proposed agent-level fault detection logic the fault is detected. It has been seen that after detection of the fault, in some intervals the samples do not exceed the threshold, these samples are considered as the false healthy detection.

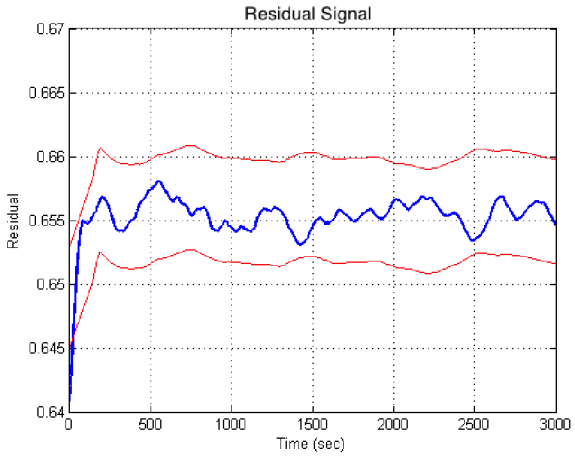


Figure 3.15: Residual signal for 1% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

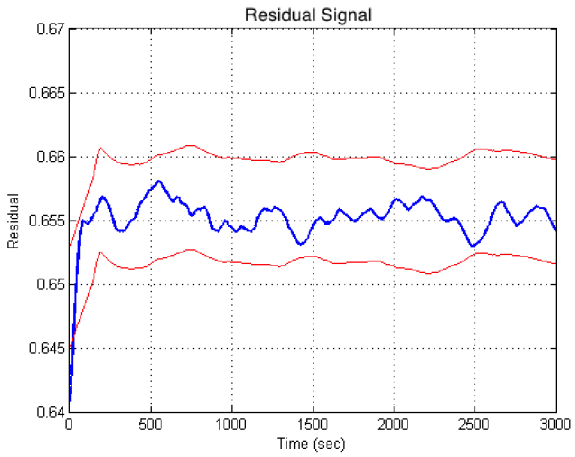


Figure 3.16: Residual signal for 2% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

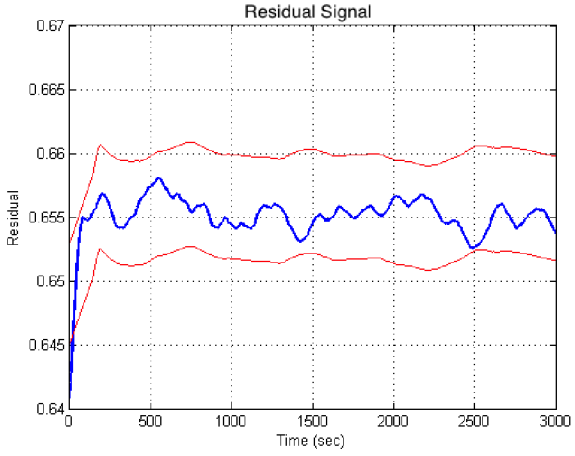


Figure 3.17: Residual signal for 3% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

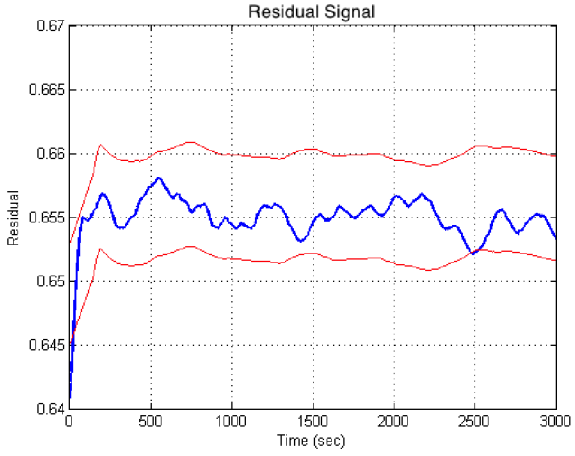


Figure 3.18: Residual signal for 4% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

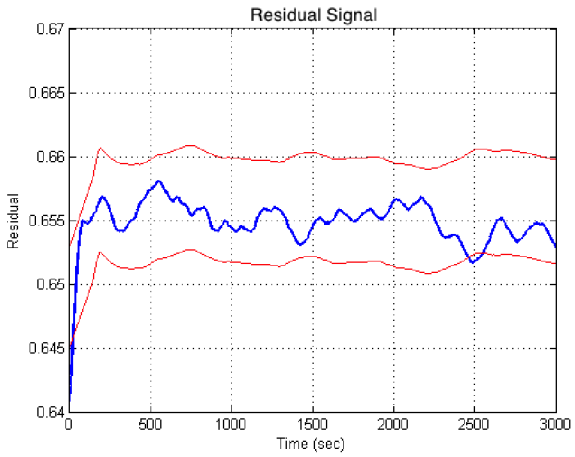


Figure 3.19: Residual signal for 5% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

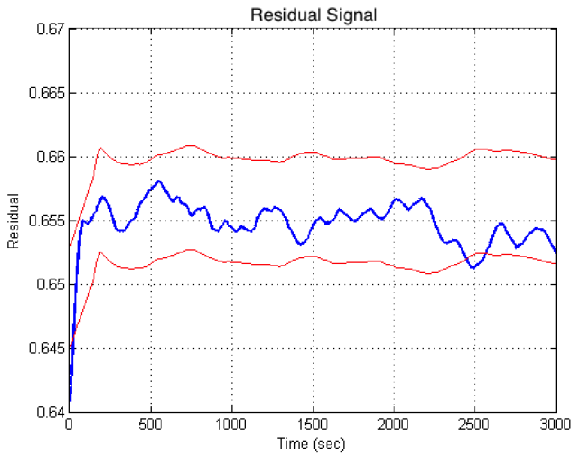


Figure 3.20: Residual signal for 6% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

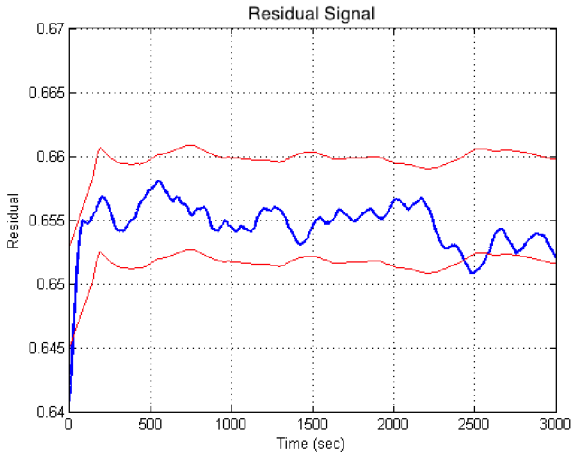


Figure 3.21: Residual signal for 7% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

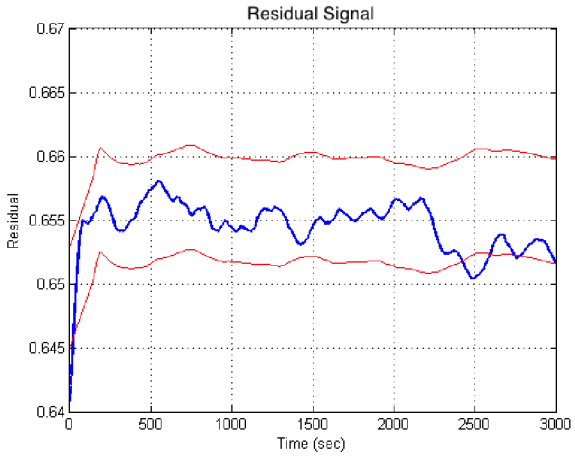


Figure 3.22: Residual signal for 8% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

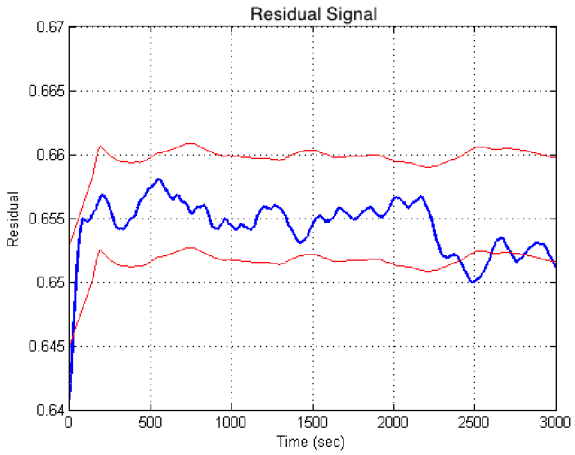


Figure 3.23: Residual signal for 9% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

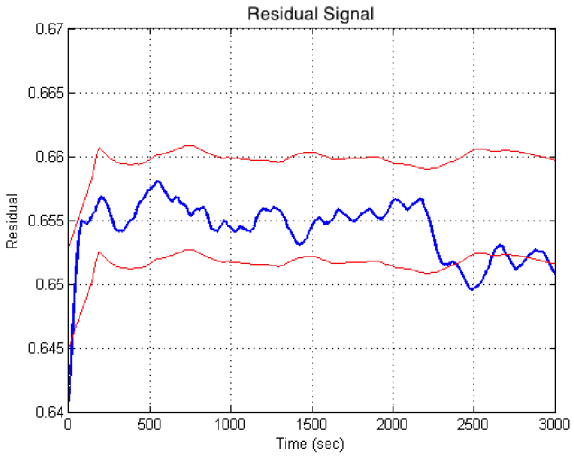


Figure 3.24: Residual signal for 10% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

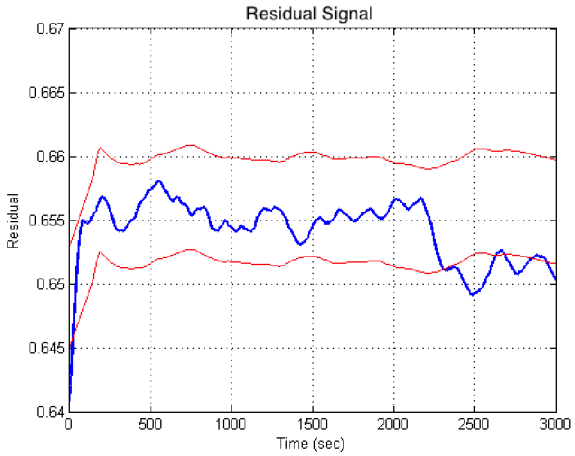


Figure 3.25: Residual signal for 11% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

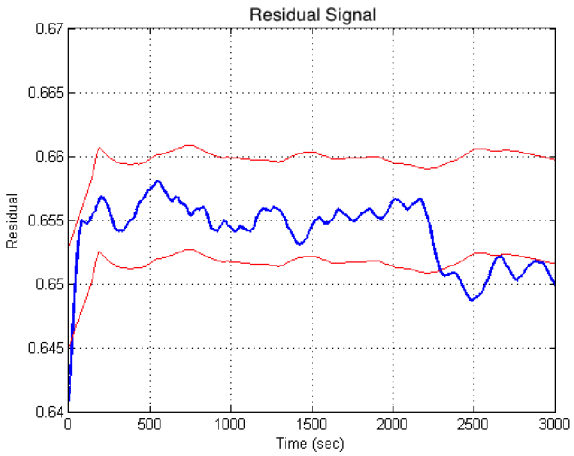


Figure 3.26: Residual signal for 12% drop in thruster torque under thruster blocking fault scenario (Fault injection at t = 2300 sec).

- From 12% to larger percentages drop in thruster torque, the residual passes the threshold and remains outside of the threshold. In these fault scenarios the occurred fault is successfully detected.

In Table 3.10, the fault injection time and fault detection time corresponding to ALFD scheme in thruster blocking fault scenario are expressed.

Table 3.10: Fault injection time and detection time in thruster blocking fault scenario for ALFD scheme.

Percentage drop in thruster torque	Fault injection time(sec)	Fault detection time(sec)
1%	2300	Not detected
2%	2300	Not detected
3%	2300	Not detected
4%	2300	Not detected
5%	2300	2480
6%	2300	2464
7%	2300	2440
8%	2300	2428
9%	2300	2406
10%	2300	2390
11%	2300	2317
12%	2300	2303

According to this table, it can be concluded that the proposed method is capable of detecting at least 5% drop in thruster torque in a thruster blocking fault scenario in a short and proper time period. In addition, it has been observed that the more sever thruster blocking faults leads to the larger residuals and smaller fault detection delays.

3.7.1.2 Fault Detection Analysis under Flooded Thruster Fault Scenarios

The Flooded Thruster fault causes an increasing in the blade rotation velocity; therefore in this fault scenario, high rotation velocity condition is considered as a fault in a

thruster. In order to simulate this fault scenario, the rotation velocity is increased by 1% to 16% from its nominal value. These permanent faults are injected to a thruster in the steady state condition at t = 2575 sec (sample # 51500). The residual signals corresponding to this fault scenario are illustrated in Figures 3.27 to 3.42.

According to the residual signals in Figures 3.27 to 3.42, it can be seen that:

- From 1% to 8% increase in rotation velocity the residual does not pass the threshold, which means the ALFD scheme does not detect the fault.
- From 9% to 12% increase in rotation velocity the residual passes the threshold which means according to proposed agent-level fault detection logic the fault is detected. It has been seen that after detection of the fault, in some intervals the samples do not exceed the threshold, these samples are considered as the false healthy detection.
- From 13% to larger percentages increase in rotation velocity the residual passes the threshold an remains outside of the threshold. In these fault scenarios the occurred fault is successfully detected.

In Table 3.11, the fault injection time and fault detection time corresponding to ALFD scheme in flooded thruster fault scenario are expressed. According to this table, it can be concluded that the proposed method is capable of detecting at least 9% increase in rotation velocity in a flooded thruster fault scenario in a short and proper time period. In addition, it has been observed that the more sever flooded thruster faults leads to the larger residuals and smaller fault detection delays.

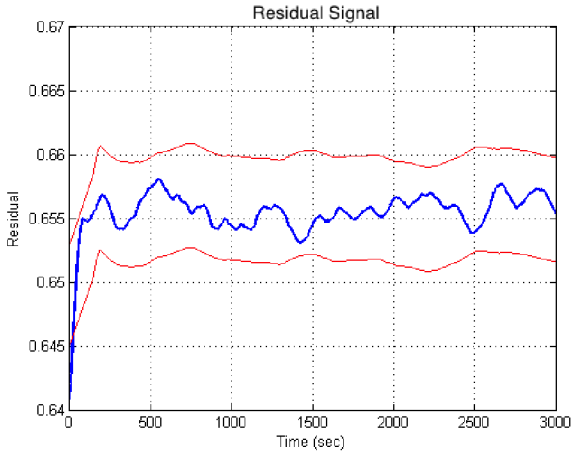


Figure 3.27: Residual signal for 1% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

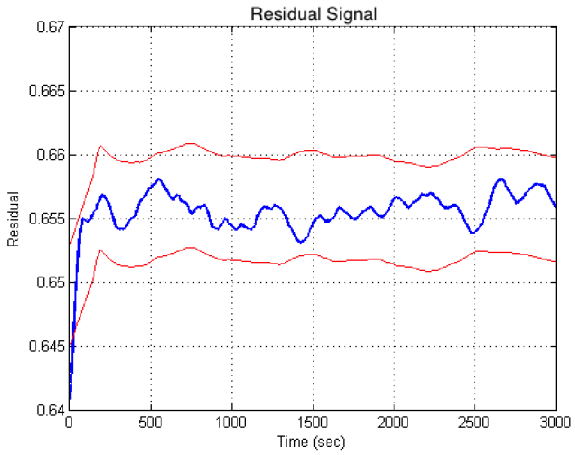


Figure 3.28: Residual signal for 2% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

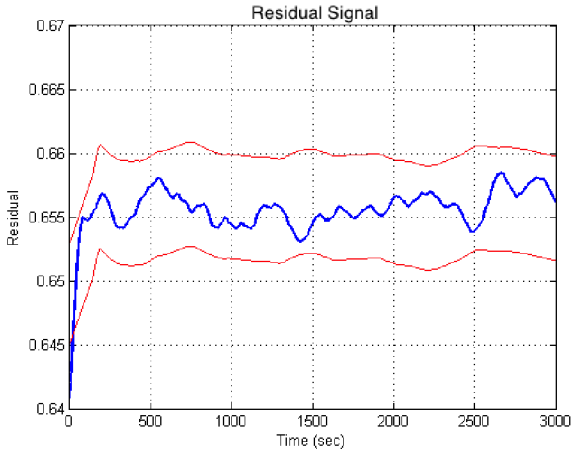


Figure 3.29: Residual signal for 3% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

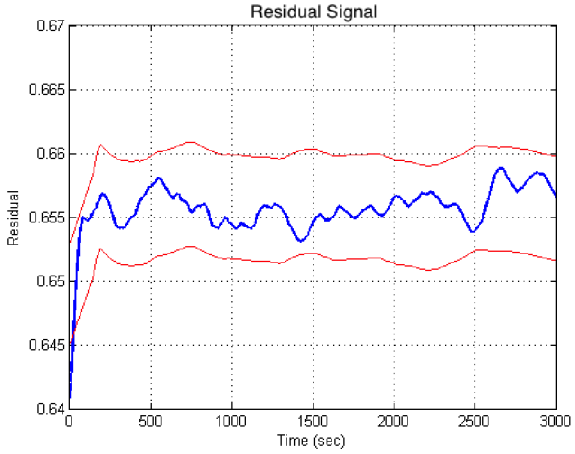


Figure 3.30: Residual signal for 4% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

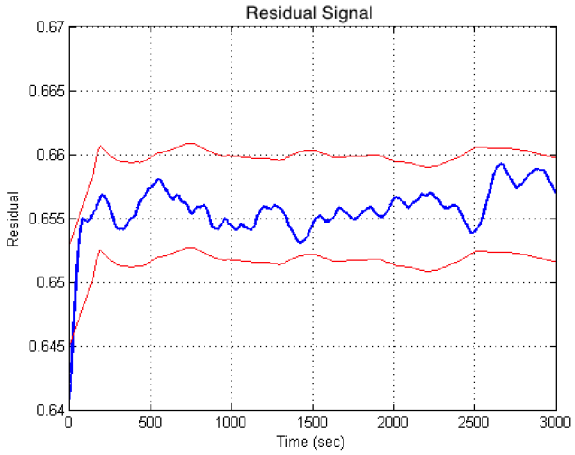


Figure 3.31: Residual signal for 5% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

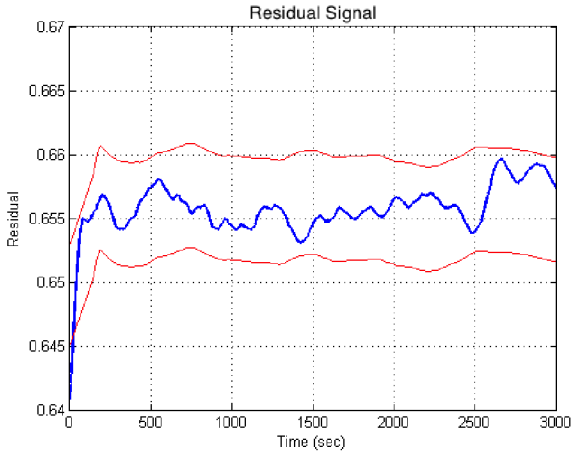


Figure 3.32: Residual signal for 6% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

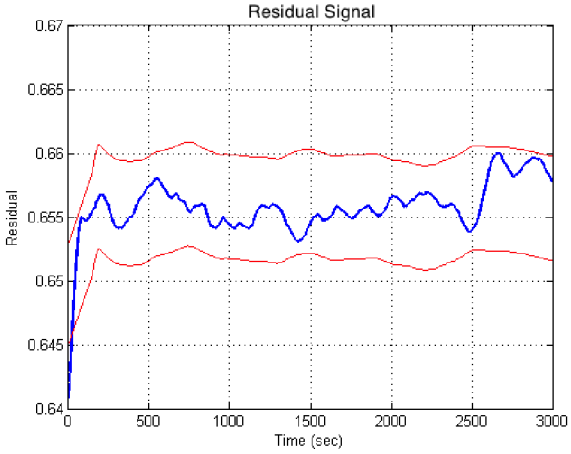


Figure 3.33: Residual signal for 7% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

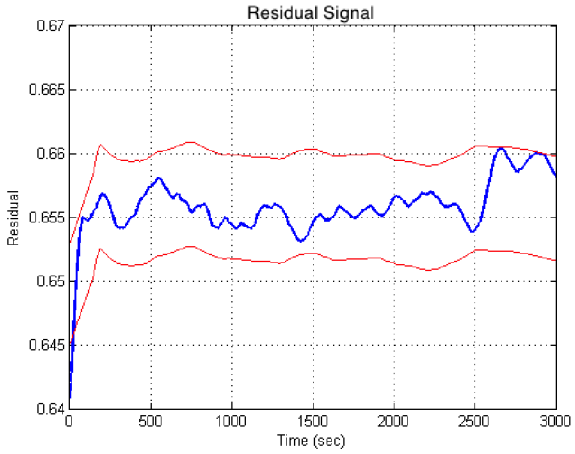


Figure 3.34: Residual signal for 8% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

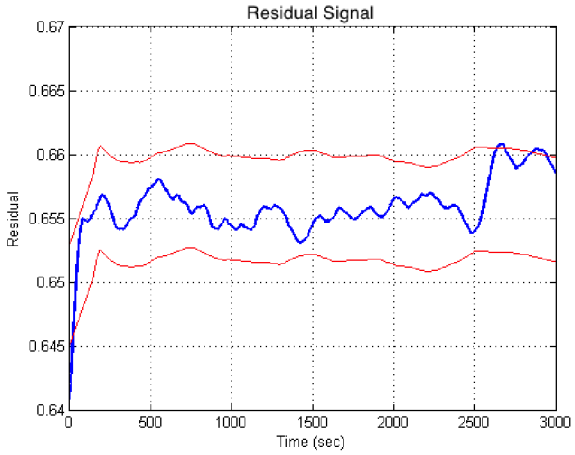


Figure 3.35: Residual signal for 9% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

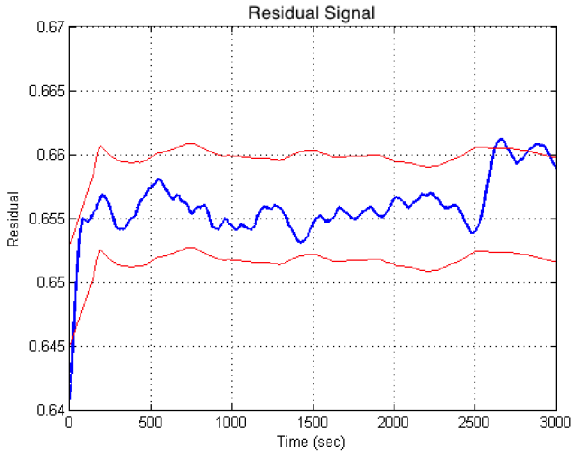


Figure 3.36: Residual signal for 10% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

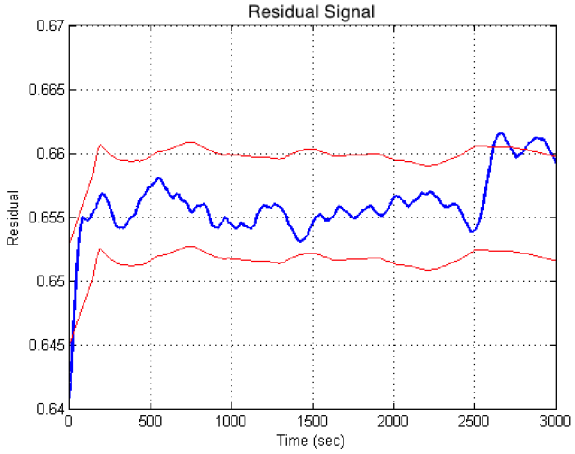


Figure 3.37: Residual signal for 11% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

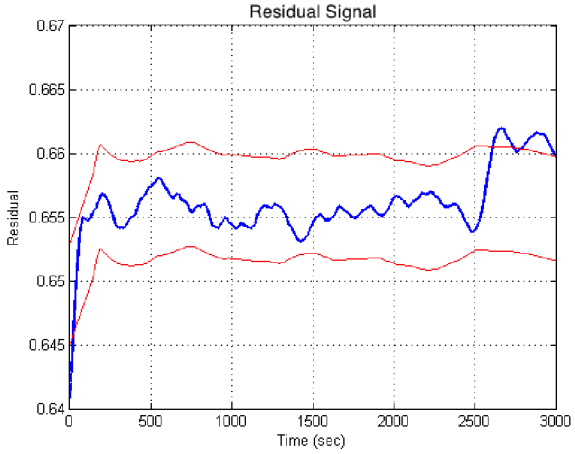


Figure 3.38: Residual signal for 12% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

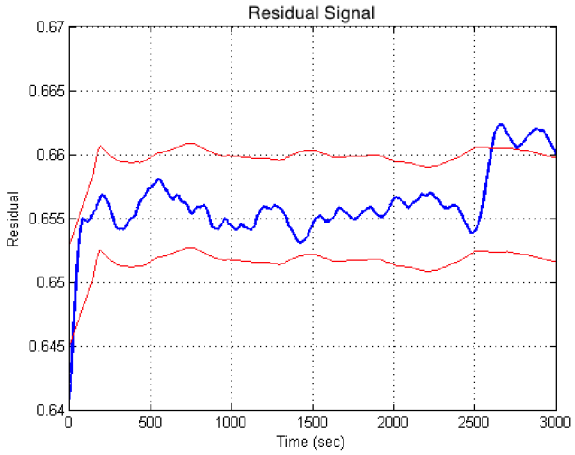


Figure 3.39: Residual signal for 13% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

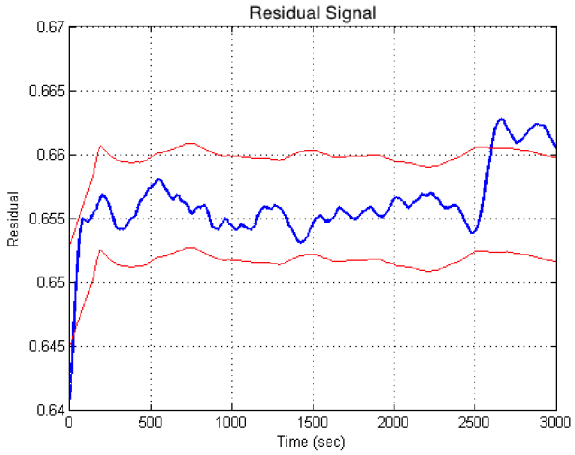


Figure 3.40: Residual signal for 14% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

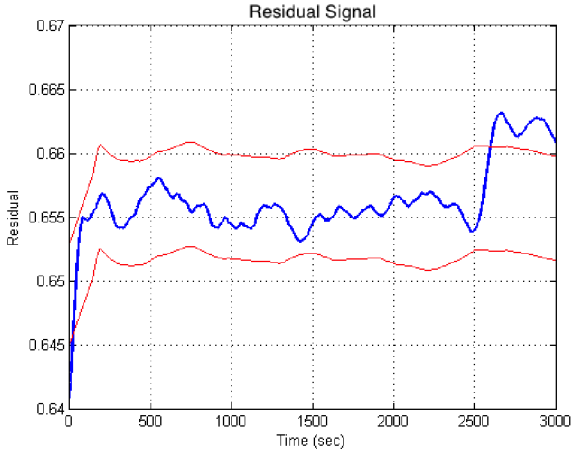


Figure 3.41: Residual signal for 15% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

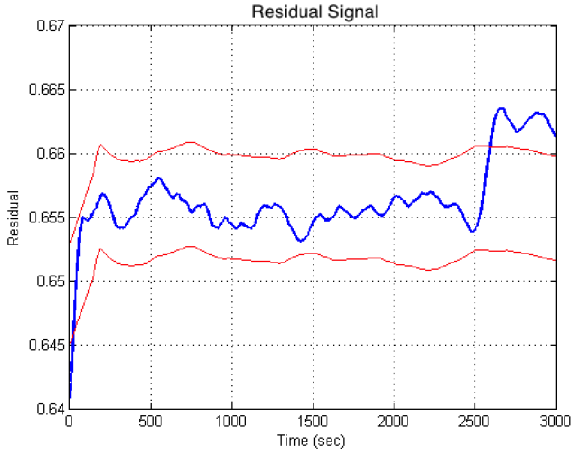


Figure 3.42: Residual signal for 16% increase in rotation velocity under flooded thruster fault scenario (Fault injection at t = 2575 sec).

Table 3.11: Fault injection time and detection time in flooded thruster fault scenario for ALFD scheme.

Percentage increase in rotation velocity	Fault injection time(sec)	Fault detection time(sec)
1%	2575	Not detected
2%	2575	Not detected
3%	2575	Not detected
4%	2575	Not detected
5%	2575	Not detected
6%	2575	Not detected
7%	2575	Not detected
8%	2575	Not detected
9%	2575	2642
10%	2575	2631
11%	2575	2624
12%	2575	2612
13%	2575	2606
14%	2575	2602
15%	2575	2596
16%	2575	2588

3.7.1.3 Fault Detection Analysis under Loss of Effectiveness in Rotor Fault Scenarios

As it is mentioned previously, in this thesis the loss of effectiveness in rotor which is decreasing in blade rotation, is considered as one of the fault scenarios in the thruster of the AUV. Hence, in this fault scenario, low rotation velocity condition is considered as a fault in a thruster. In order to simulate this fault scenario, the rotation velocity is dropped by 1% to 16% from its nominal value. These permanent faults are injected to a thruster in the steady state condition at t = 2390 sec (sample # 47800). The residual signals corresponding to this fault scenario are illustrated in Figure 3.43 to Figure 3.58.

According to the residual signals in Figures 3.43 to 3.58, it can be seen that:

- From 1% to 4% drop in rotation velocity the residual signals do not pass the

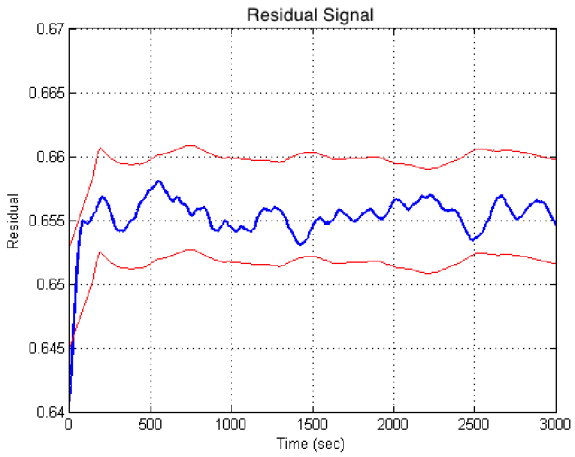


Figure 3.43: Residual signal for 1% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

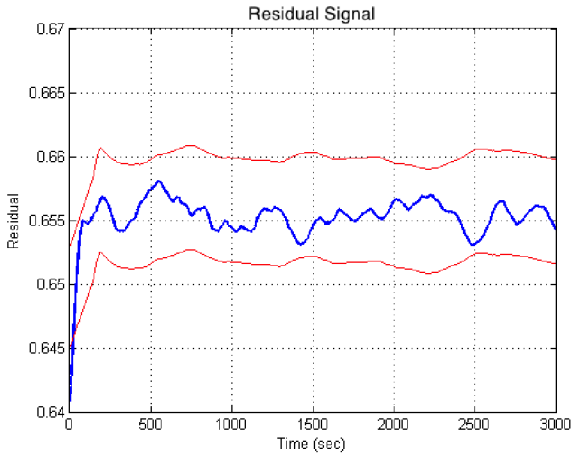


Figure 3.44: Residual signal for 2% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

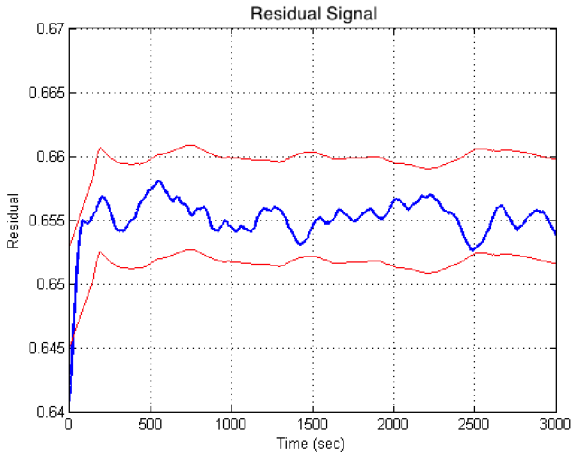


Figure 3.45: Residual signal for 3% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

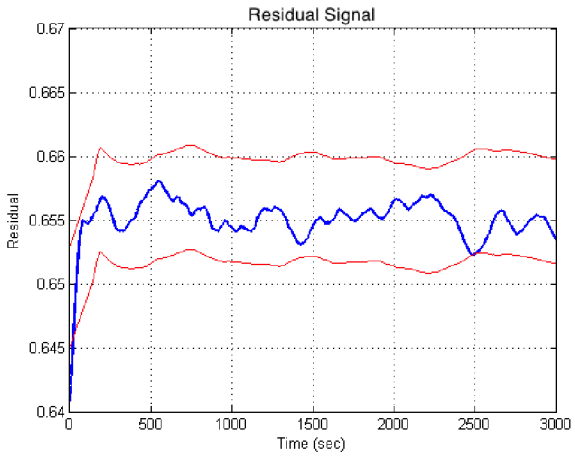


Figure 3.46: Residual signal for 4% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

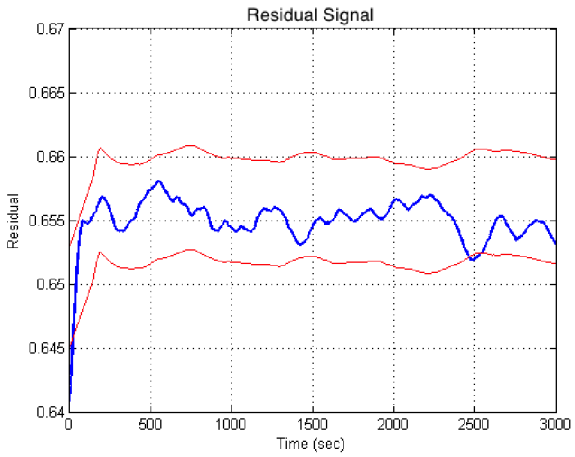


Figure 3.47: Residual signal for 5% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

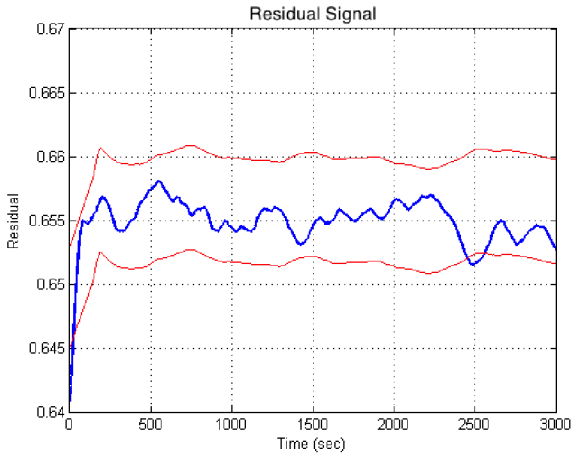


Figure 3.48: Residual signal for 6% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

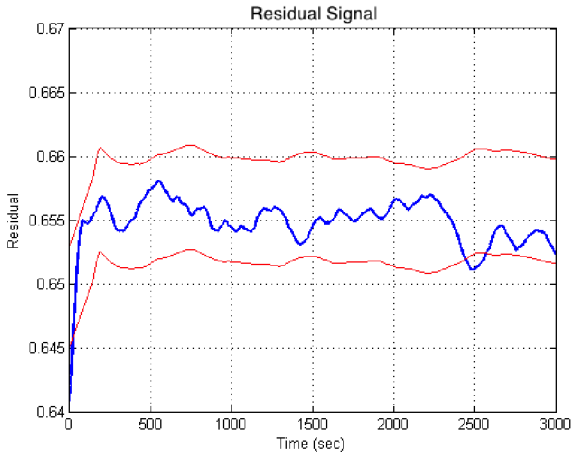


Figure 3.49: Residual signal for 7% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

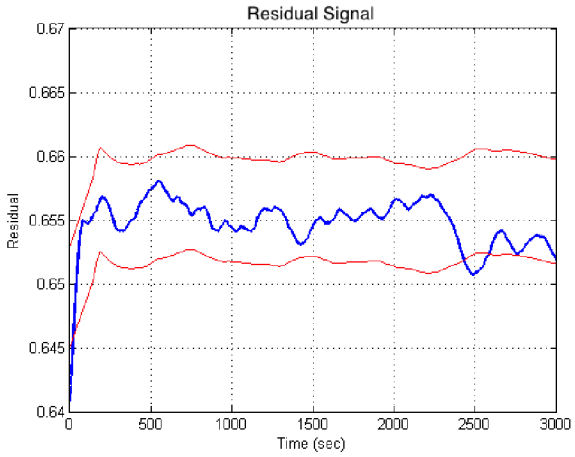


Figure 3.50: Residual signal for 8% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

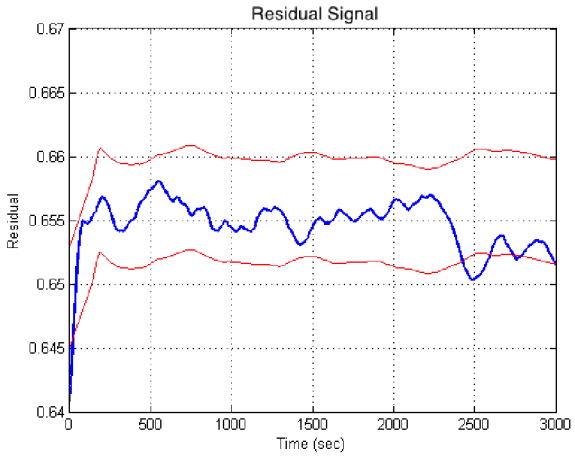


Figure 3.51: Residual signal for 9% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

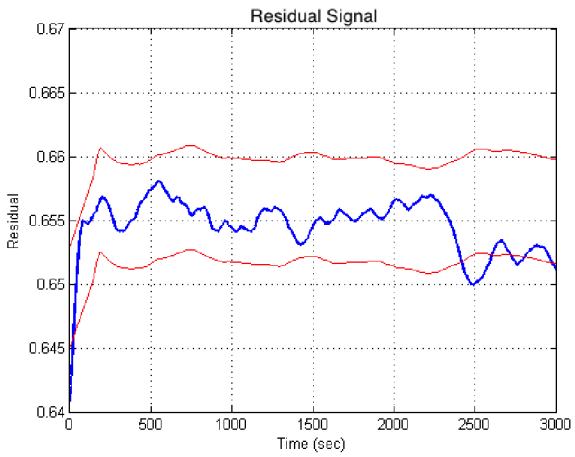


Figure 3.52: Residual signal for 10% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

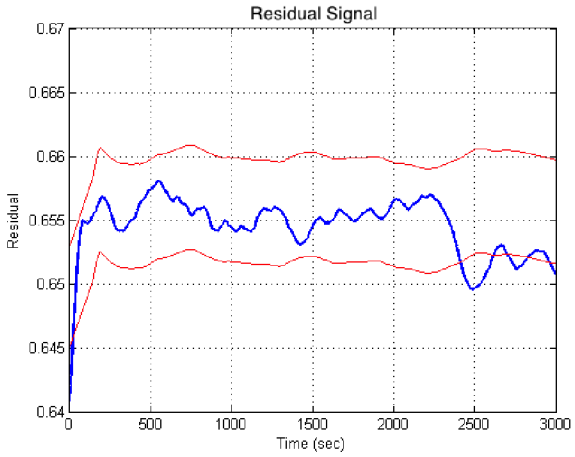


Figure 3.53: Residual signal for 11% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

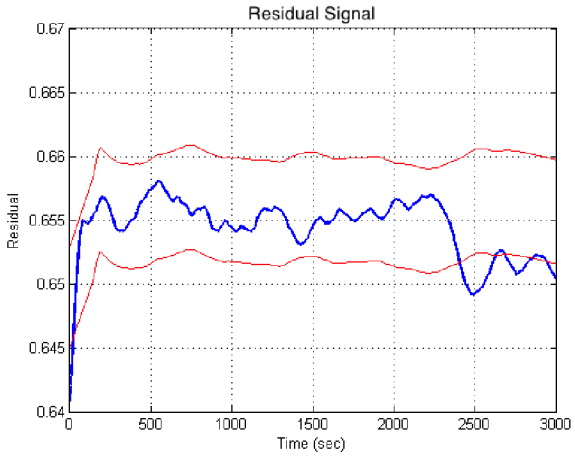


Figure 3.54: Residual signal for 12% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

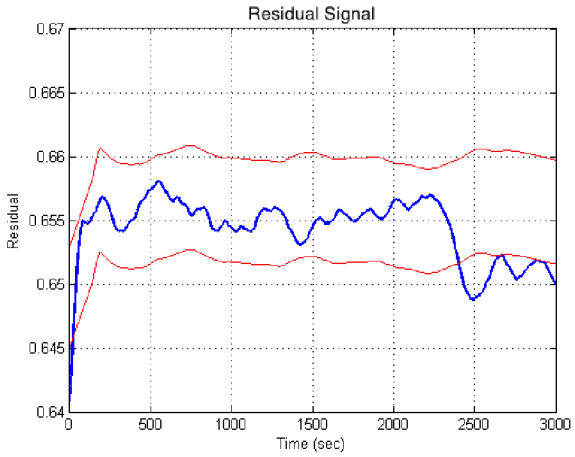


Figure 3.55: Residual signal for 13% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

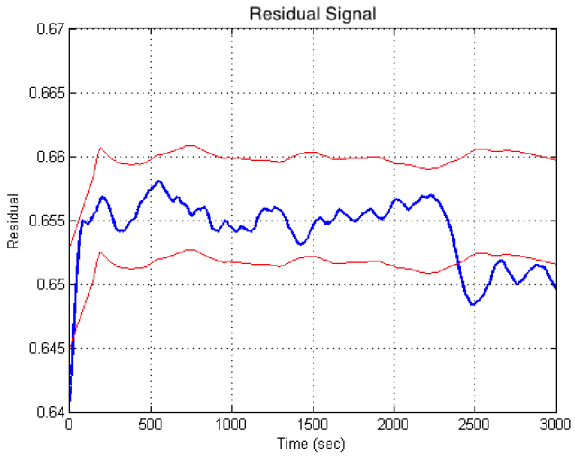


Figure 3.56: Residual signal for 14% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

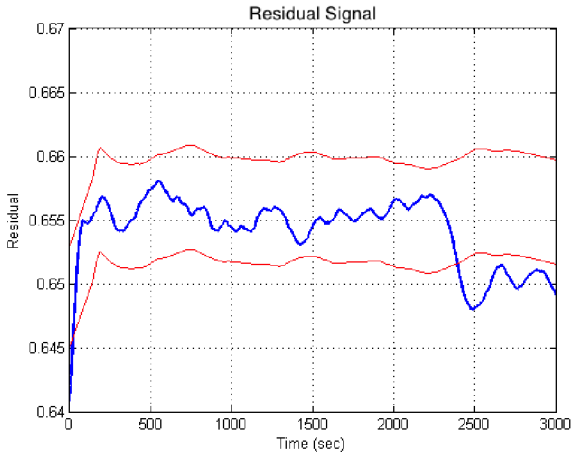


Figure 3.57: Residual signal for 15% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

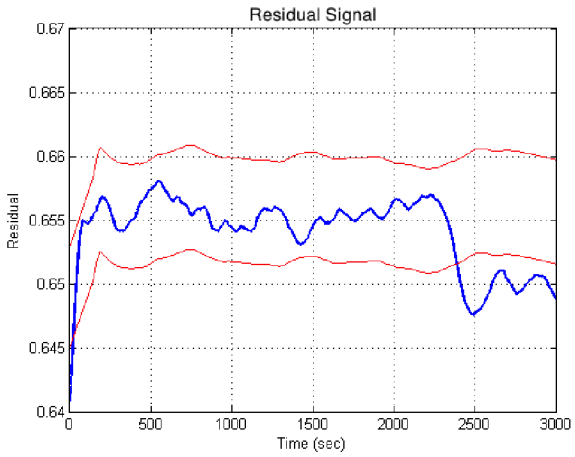


Figure 3.58: Residual signal for 16% drop in rotation velocity under loss of effectiveness in rotor fault scenario (Fault injection at t = 2390 sec).

threshold, which mean the ALFD scheme does not detect the fault.

- From 5% to 12% drop in rotation velocity the residual signals pass the threshold which means according to proposed agent-level fault detection logic the fault is detected. It has been seen that after detection of the fault, in some intervals the samples do not exceed the threshold, these samples are considered as the false healthy detection.

- From 13% to larger percentages drop in rotation velocity, the residual passes the threshold and remains outside of the threshold. In these fault scenarios the fault is successfully detected.

In Table 3.12, the fault injection time and fault detection time corresponding to ALFD scheme in loss of effectiveness in rotor fault scenario are expressed.

Table 3.12: Fault injection and detection time in loss of effectiveness in rotor fault scenario for ALFD scheme.

Percentage drop in rotation velocity	Fault injection time(sec)	Fault detection time(sec)
1%	2390	Not detected
2%	2390	Not detected
3%	2390	Not detected
4%	2390	Not detected
5%	2390	2478
6%	2390	2468
7%	2390	2454
8%	2390	2440
9%	2390	2432
10%	2390	2429
11%	2390	2422
12%	2390	2416
13%	2390	2411
14%	2390	2404
15%	2390	2400
16%	2390	2394

According to this table, it can be concluded that the proposed method is capable of detecting at least 5% drop in rotation velocity in a loss of effectiveness in rotor fault scenario in a short and proper time period. In addition, it has been observed that the more sever loss of effectiveness in rotor faults leads to the larger residuals and smaller fault detection delays.

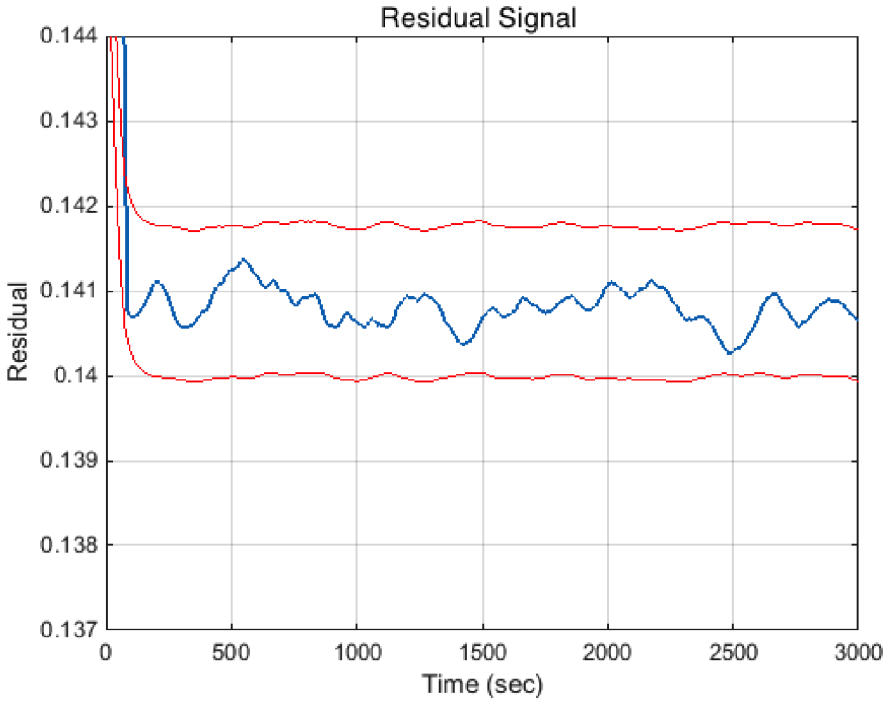
3.7.2 Formation-Level Fault Detection Analysis

3.7.2.1 Fault Detection Analysis under Thruster Blocking Fault Scenarios

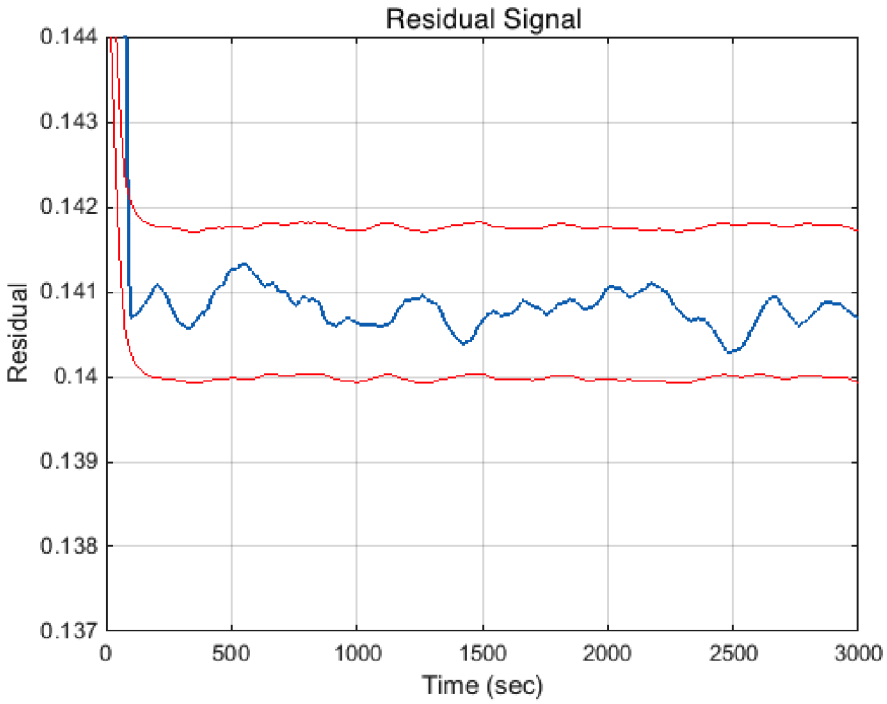
In this fault scenario, similar to agent-level fault detection scheme, low torque condition is considered as a fault in a thruster. In order to simulate this fault scenario , thruster torque is dropped by 1% to 8% . This permanent fault is injected to the thruster in the steady state condition at t = 1735 sec (sample # 34700). The residuals corresponding to this fault scenario are illustrated in Figures 3.59 to 3.65 when the fault occurred in AUV_1.

According to the residual signals in Figures 3.59 to 3.65, it can be seen that:

- From 1% to 2% drop in thruster torque both residual signals do not pass the threshold, which means the FLFD scheme does not detect the fault.
- From 3% to 4% drop in thruster torque the residual signals pass the threshold which means according to proposed formation-level fault detection logic the fault is detected. It has been seen that after detection of the fault, in some intervals the samples do not exceed the threshold, these samples are considered as the false healthy detection.
- From 5% to larger percentages drop in thruster torque, the residuals pass the threshold and the occurred fault can be successfully detected.

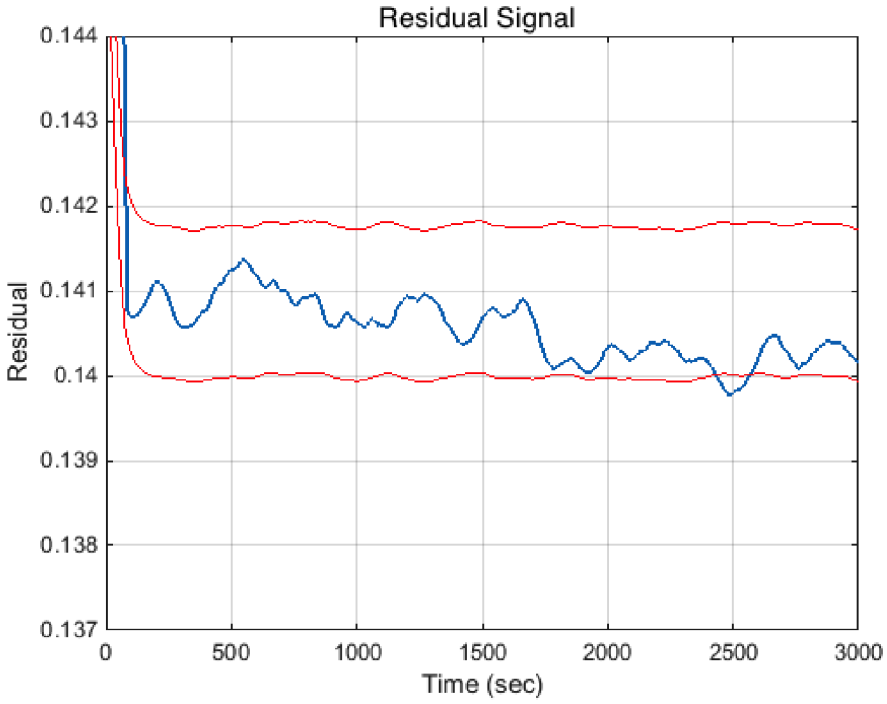


(a) Residual in AUV_1 with respect to AUV_2

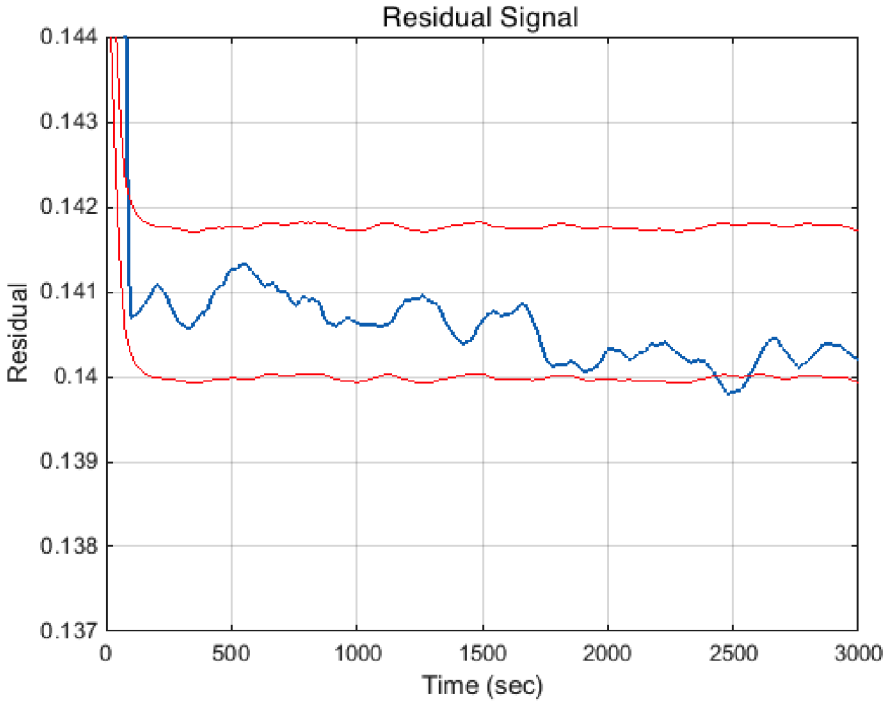


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.59: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 2% drop in thruster torque in thruster blocking fault scenario in FLFD scheme (Fault injection at t = 1735 sec).

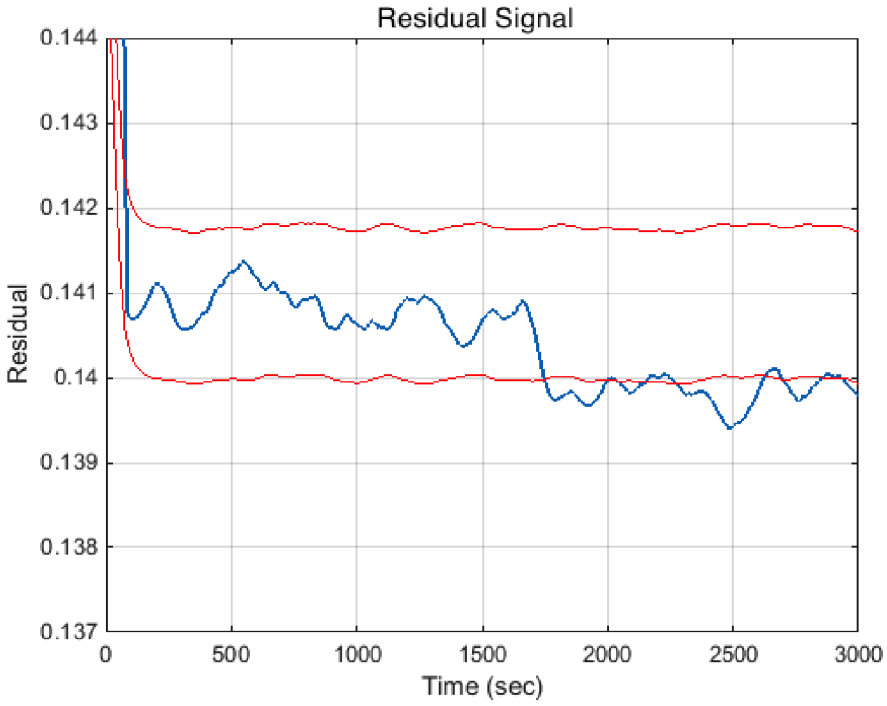


(a) Residual in AUV_1 with respect to AUV_2

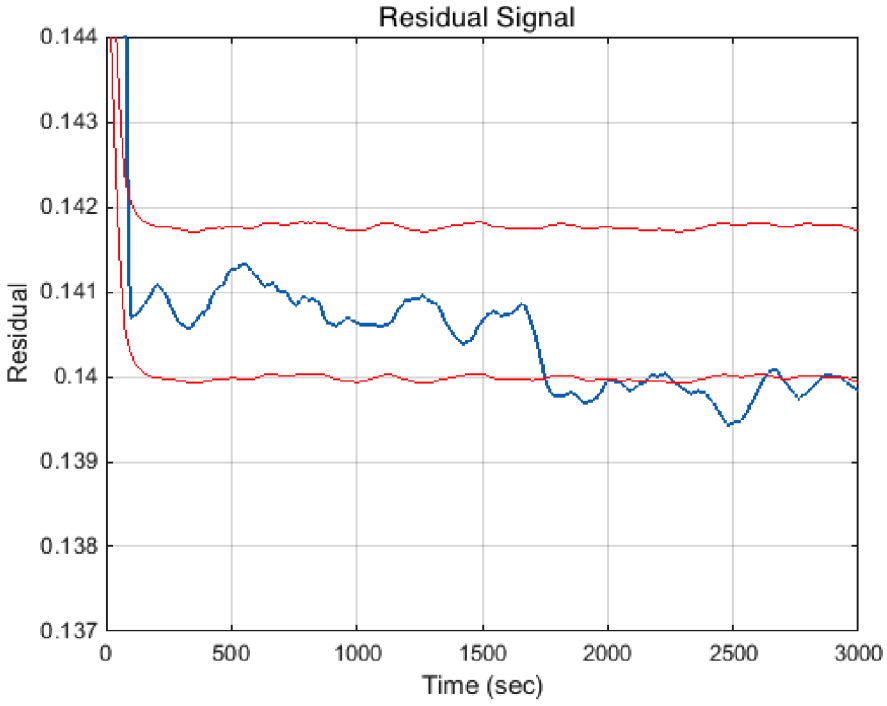


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.60: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 3% drop in thruster torque in thruster blocking fault scenario in FLFD scheme (Fault injection at t = 1735 sec).

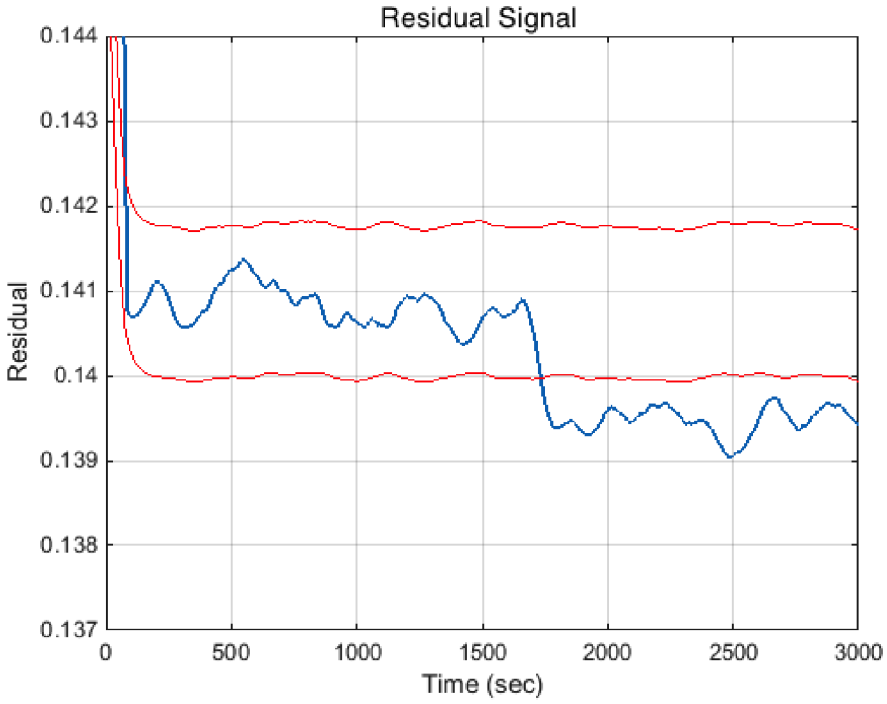


(a) Residual in AUV_1 with respect to AUV_2

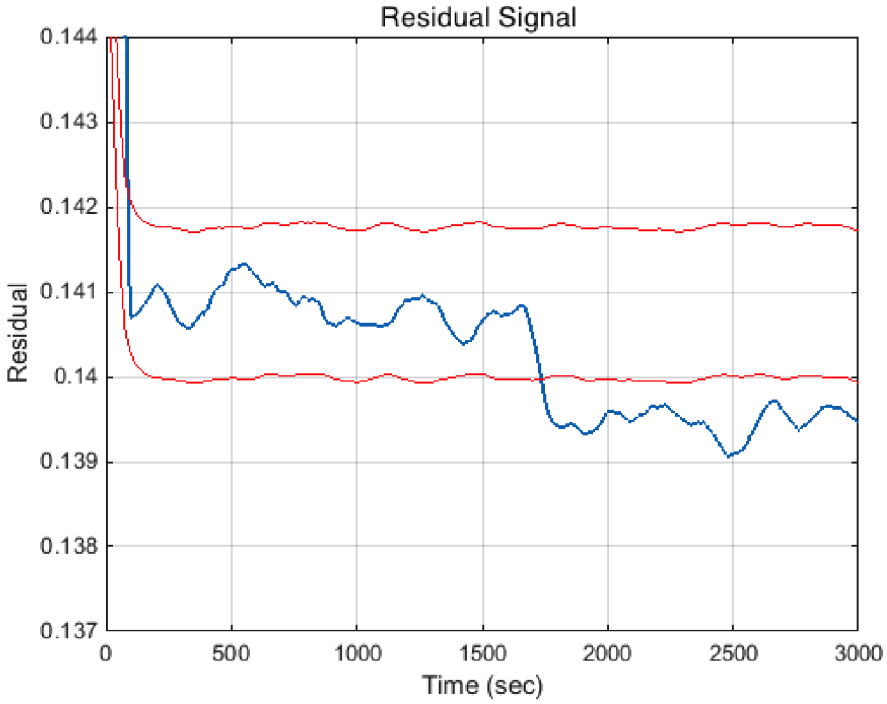


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.61: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 4% drop in thruster torque in thruster blocking fault scenario in FLFD scheme (Fault injection at t = 1735 sec).

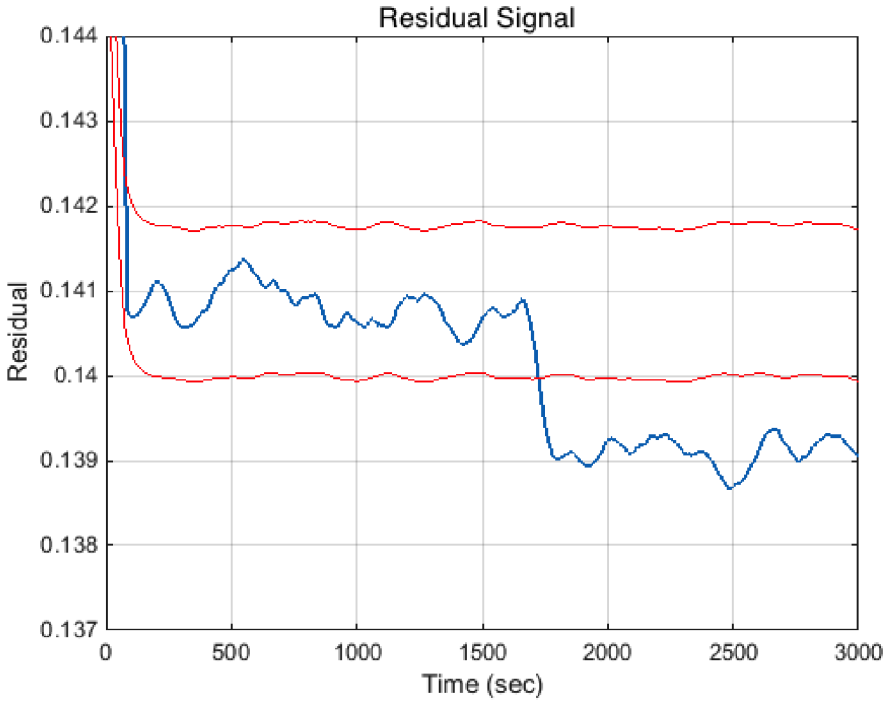


(a) Residual in AUV_1 with respect to AUV_2

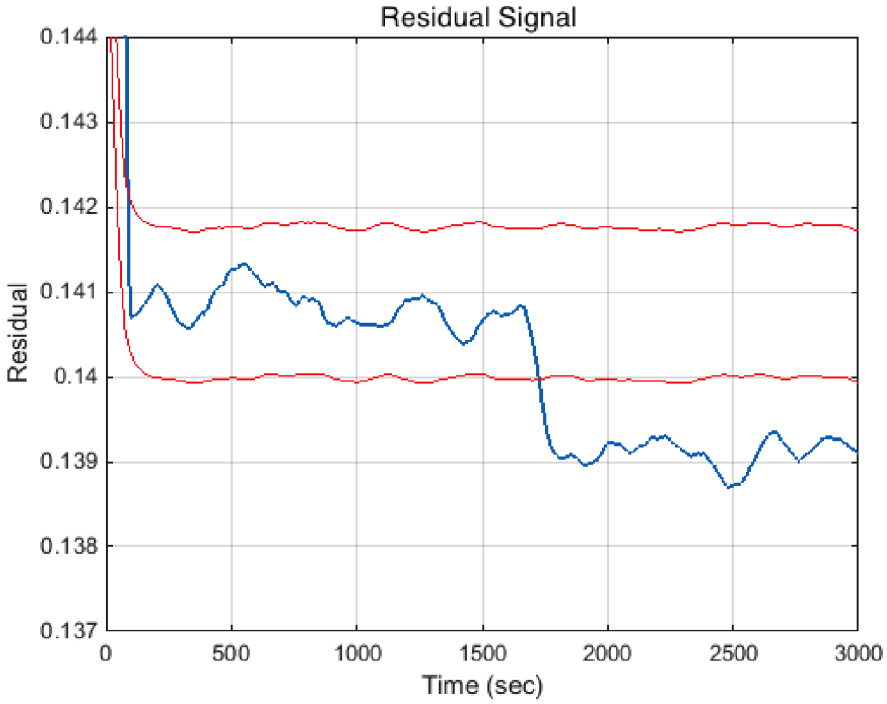


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.62: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 5% drop in thruster torque in thruster blocking fault scenario in FLFD scheme (Fault injection at t = 1735 sec).

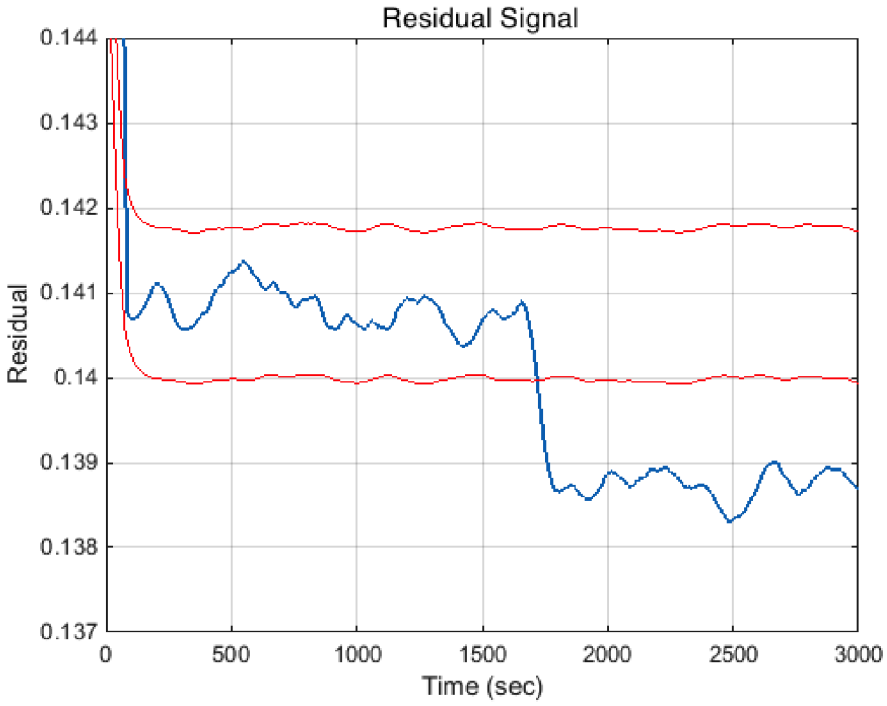


(a) Residual in AUV_1 with respect to AUV_2

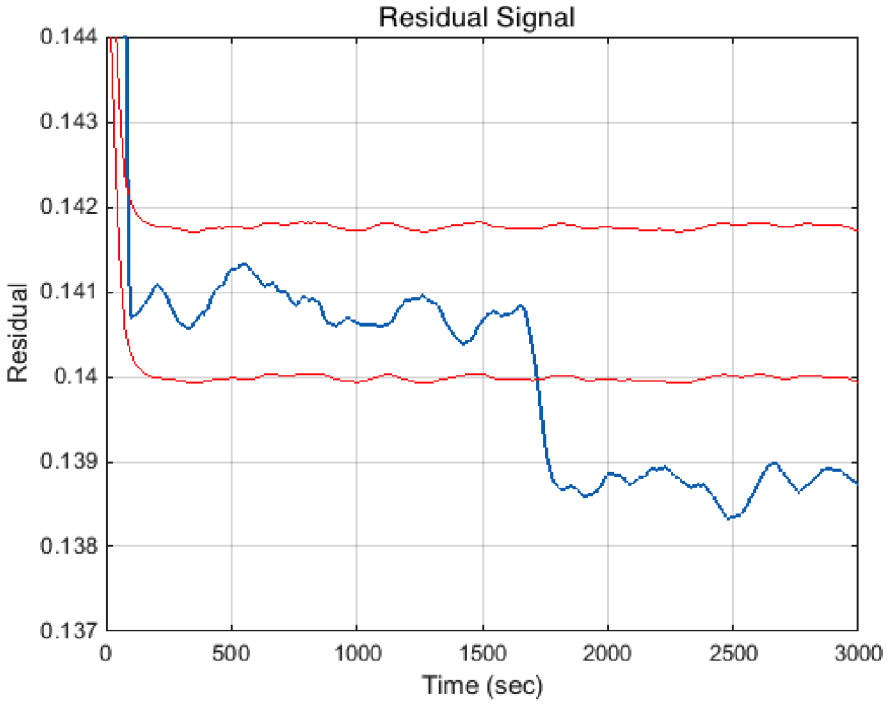


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.63: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 6% drop in thruster torque in thruster blocking fault scenario in FLFD scheme (Fault injection at t = 1735 sec).

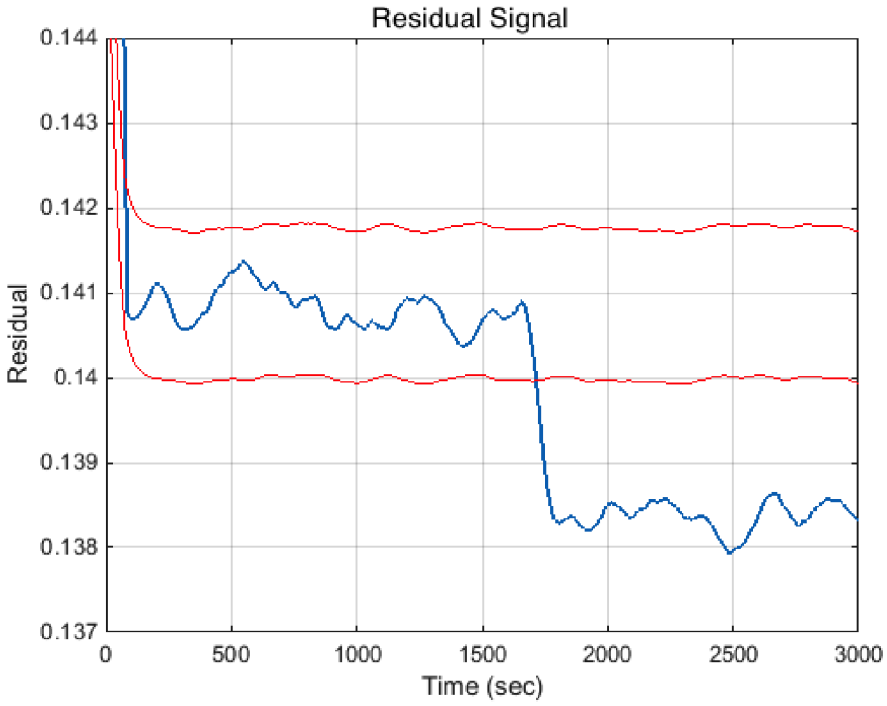


(a) Residual in AUV_1 with respect to AUV_2

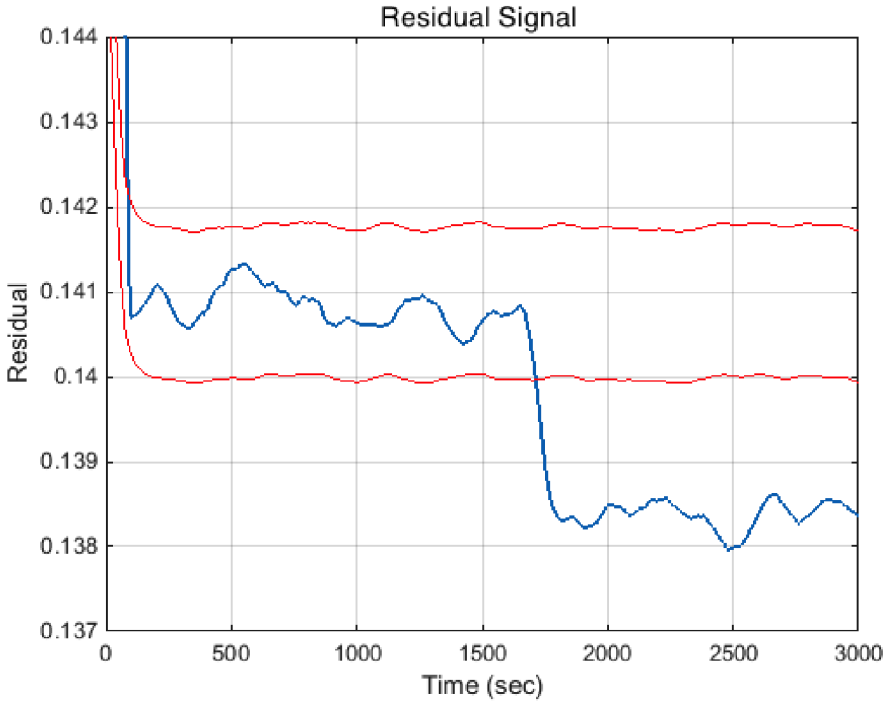


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.64: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 7% drop in thruster torque in thruster blocking fault scenario in FLFD scheme (Fault injection at t = 1735 sec).



(a) Residual in AUV_1 with respect to AUV_2



(b) Residual in AUV_1 with respect to AUV_3

Figure 3.65: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 8% drop in thruster torque in thruster blocking fault scenario in FLFD scheme (Fault injection at t = 1735 sec).

In Table 3.13, the fault injection time and fault detection time corresponding to FLFD scheme in thruster blocking fault scenario are expressed.

Table 3.13: Fault injection time and detection time in thruster blocking fault scenario for FLFD scheme.

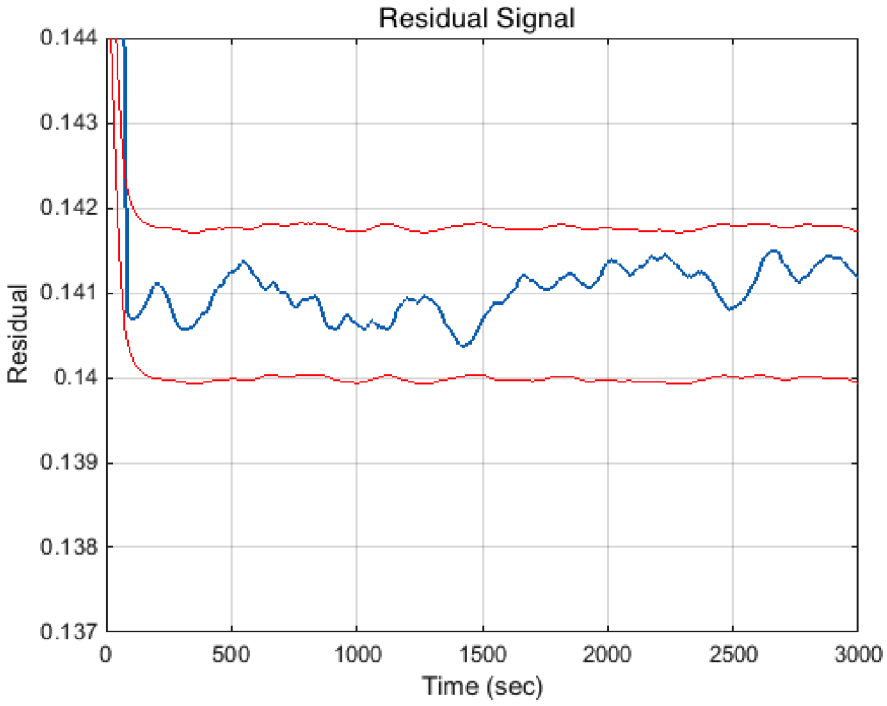
Percentage drop in thruster torque	Fault injection time(sec)	Fault detection time(sec)
1%	1735	Not detected
2%	1735	Not detected
3%	1735	1938
4%	1735	1744
5%	1735	1740
6%	1735	1740
7%	1735	1740
8%	1735	1740

According to this table, it can be concluded that the proposed method is capable of detecting at least 3% drop in thruster torque in a thruster blocking fault scenario in a short and proper time period. In addition, it has been observed that the more sever thruster blocking faults leads to the larger residuals and smaller fault detection delays.

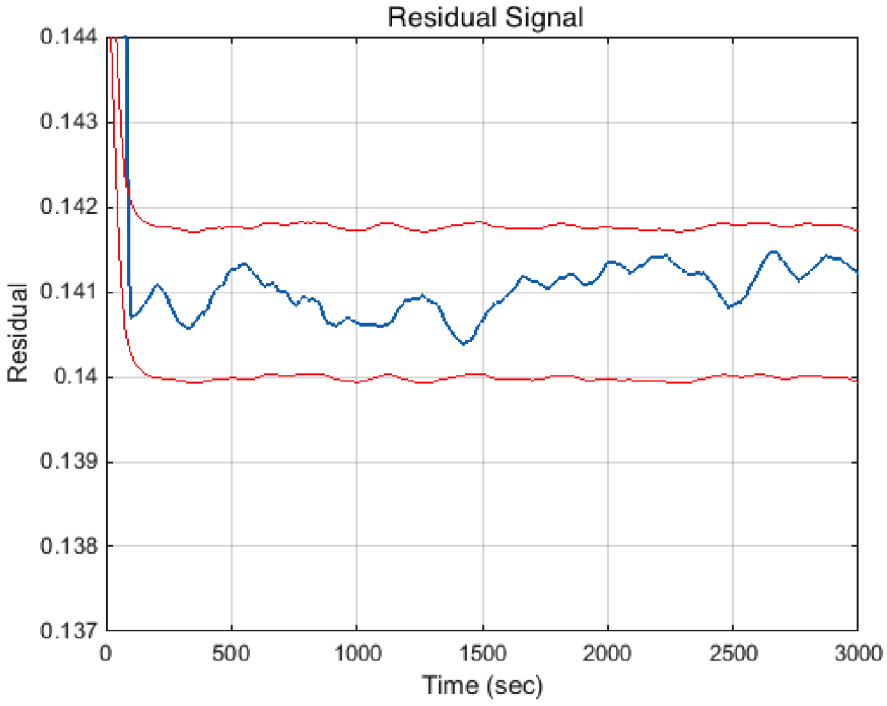
3.7.2.2 Fault Detection Analysis under Flooded Thruster Fault Scenarios

The Flooded Thruster fault causes an increasing in the blade rotation velocity; therefore in this fault scenario, high rotation velocity condition is considered as a fault in a thruster. In order to simulate this fault scenario, the rotation velocity is increased by 1% to 8% from its nominal value. These faults are injected to a thruster in the steady state condition at t = 1600 sec (sample # 32000). The residuals corresponding to this fault scenario are illustrated in Figures 3.66 to 3.72 when the fault occurred in AUV_1.

According to the residual signals in Figures 3.66 to 3.72, it can be seen that:

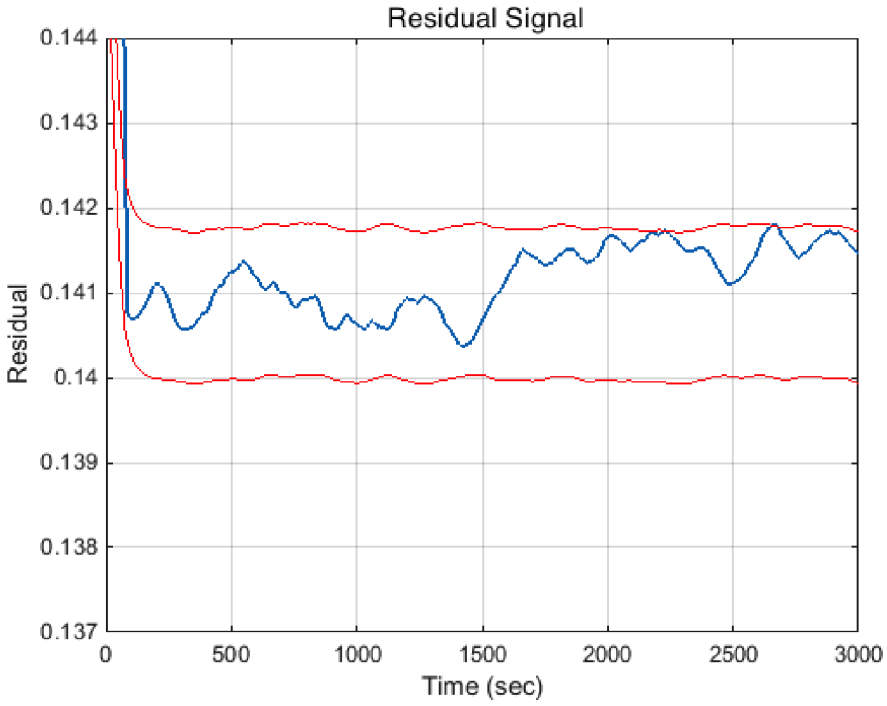


(a) Residual in AUV_1 with respect to AUV_2

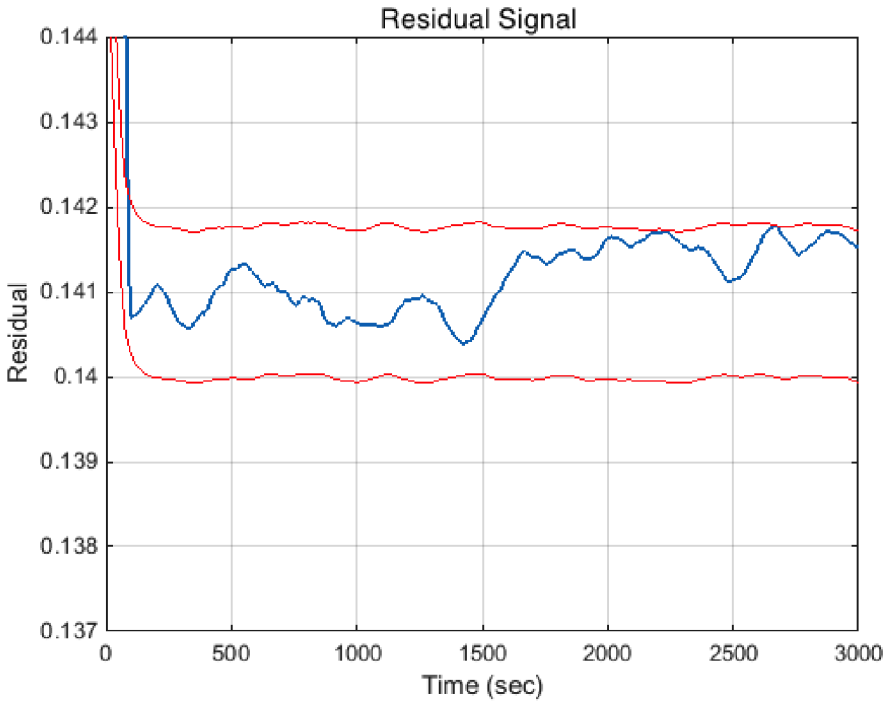


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.66: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 2% increase in rotation velocity in flooded thruster fault scenario in FLFD scheme (Fault injection at t = 1600 sec).

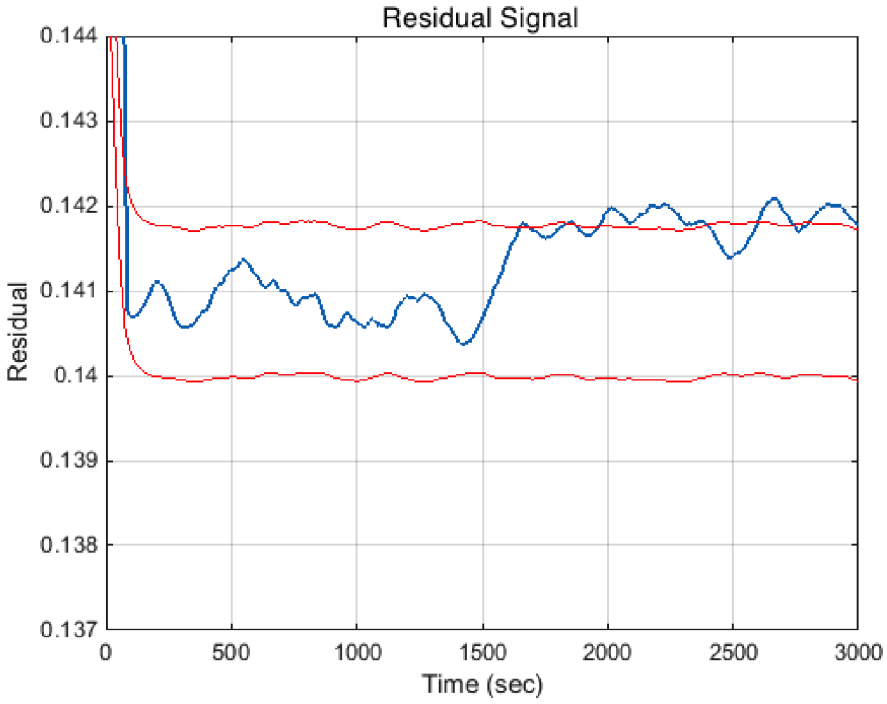


(a) Residual in AUV_1 with respect to AUV_2

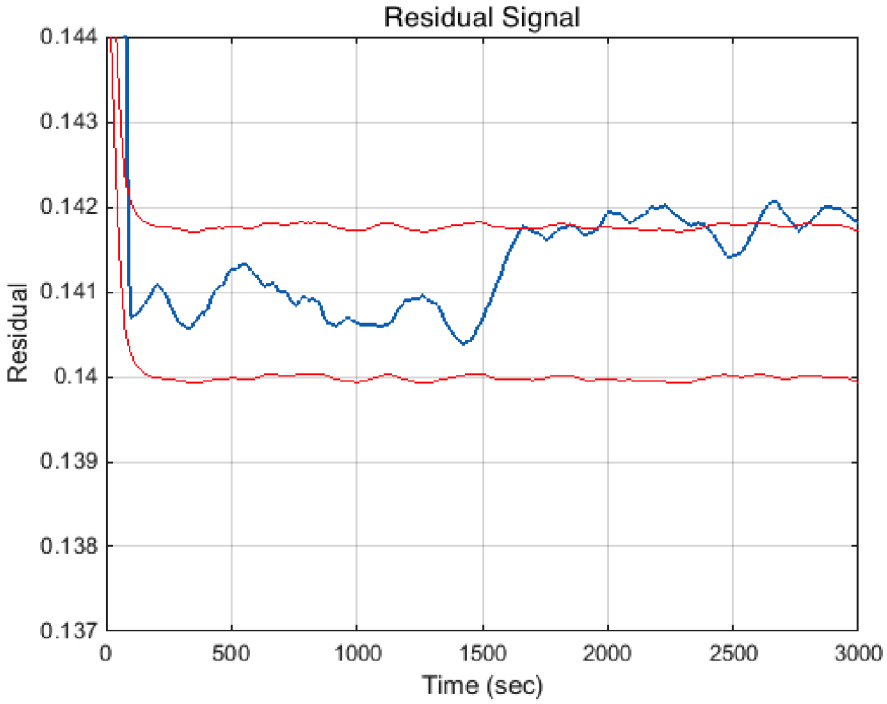


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.67: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 3% increase in rotation velocity in flooded thruster fault scenario in FLFD scheme (Fault injection at t = 1600 sec).

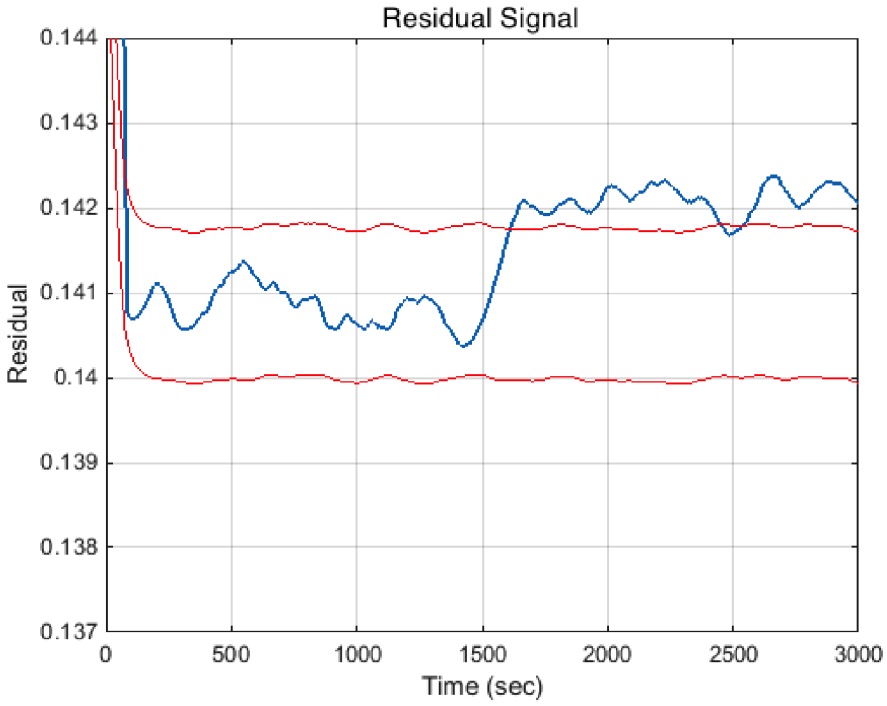


(a) Residual in AUV_1 with respect to AUV_2

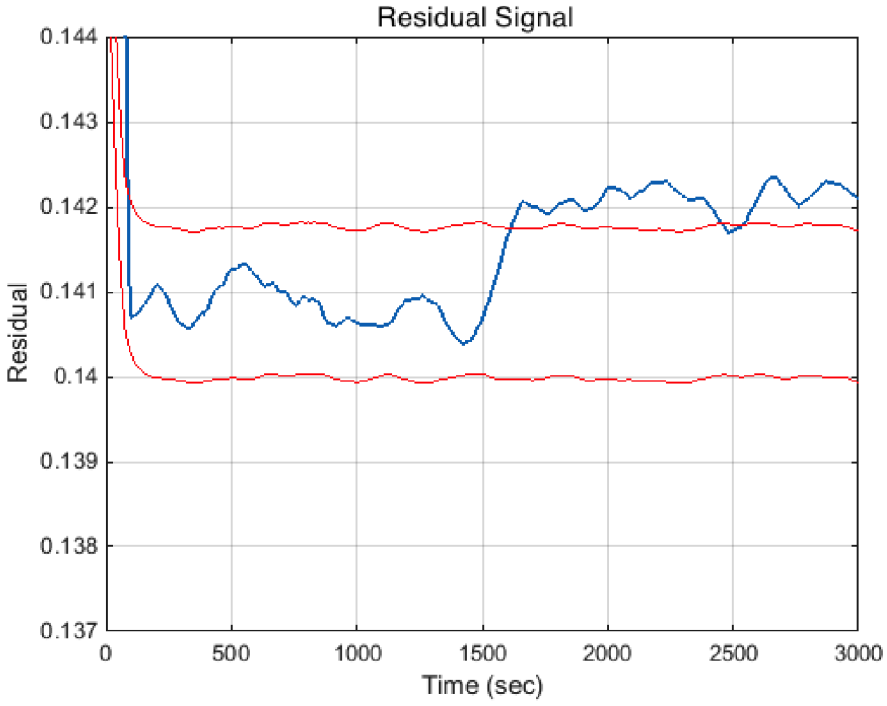


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.68: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 4% increase in rotation velocity in flooded thruster fault scenario in FLFD scheme (Fault injection at t = 1600 sec).

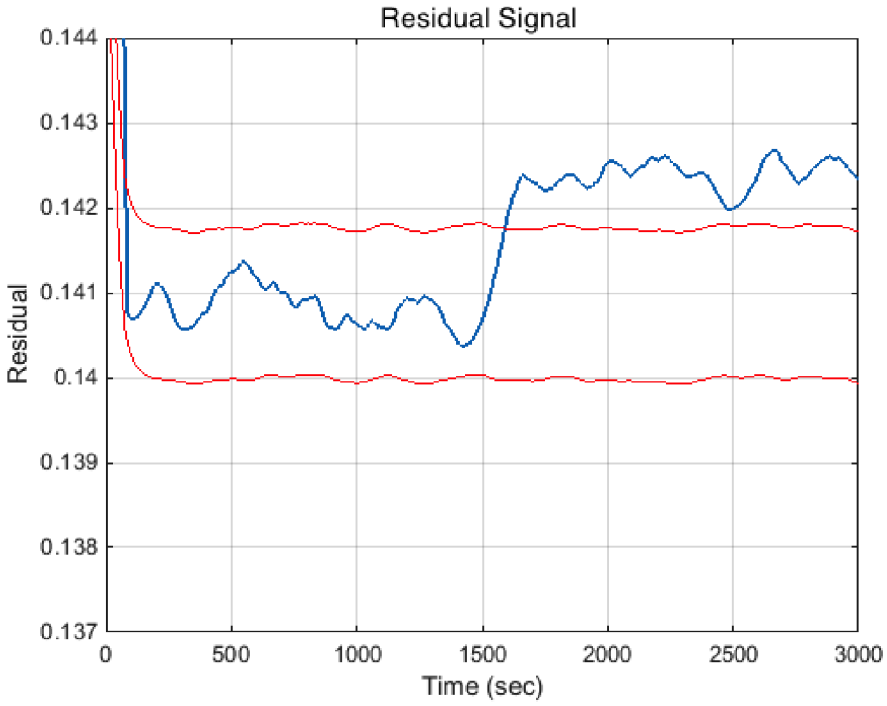


(a) Residual in AUV_1 with respect to AUV_2

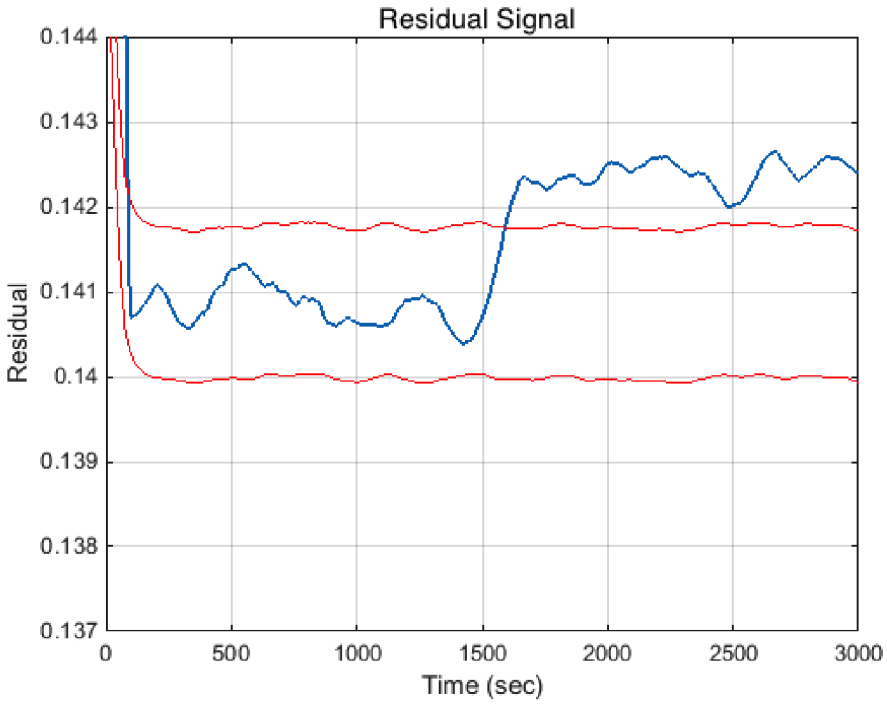


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.69: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 5% increase in rotation velocity in flooded thruster fault scenario in FLFD scheme (Fault injection at t = 1600 sec).

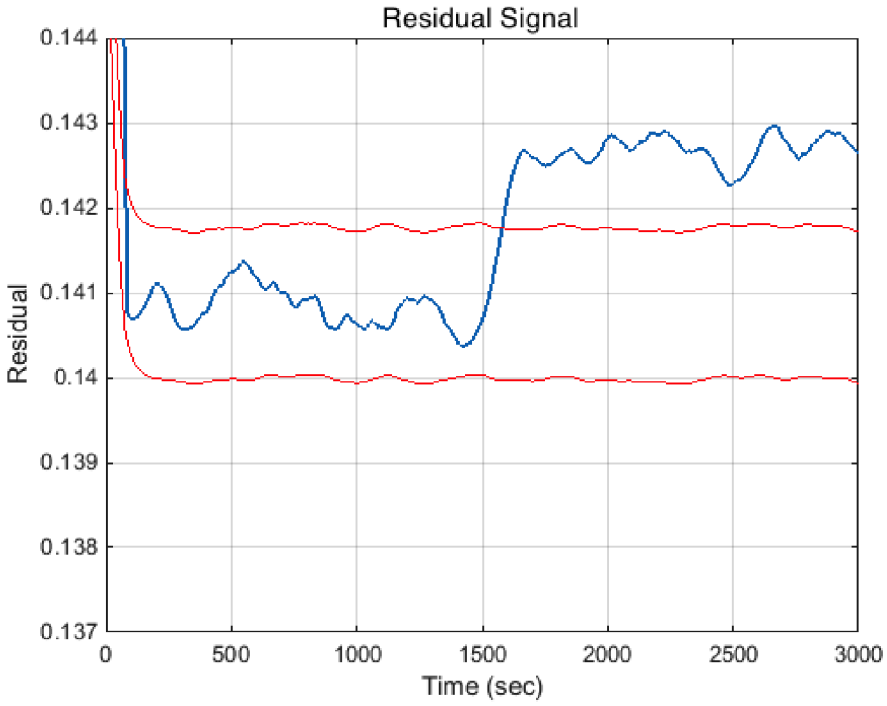


(a) Residual in AUV_1 with respect to AUV_2

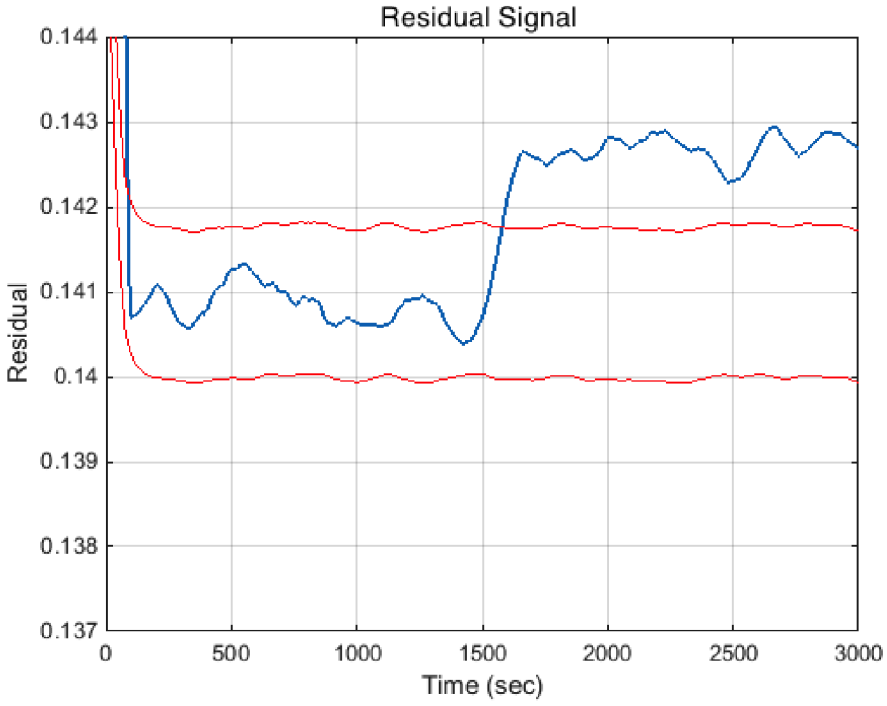


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.70: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 6% increase in rotation velocity in flooded thruster fault scenario in FLFD scheme (Fault injection at t = 1600 sec).

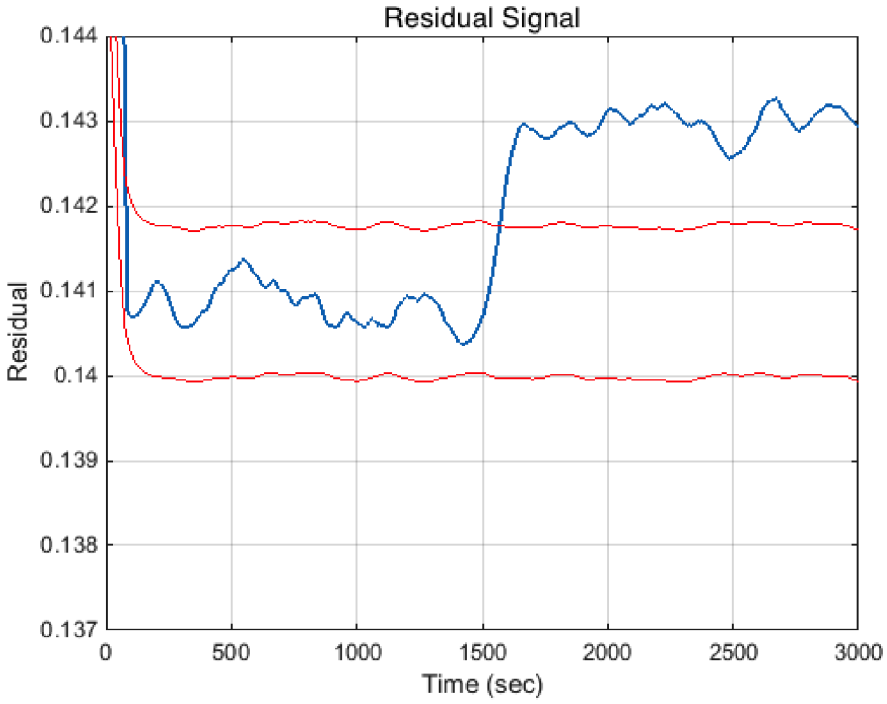


(a) Residual in AUV_1 with respect to AUV_2

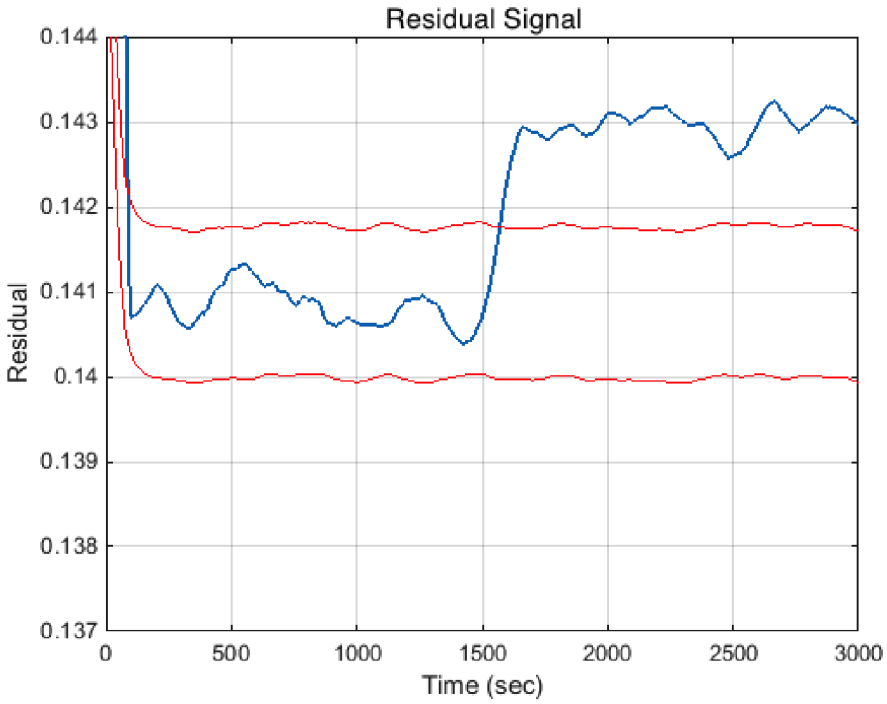


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.71: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 7% increase in rotation velocity in flooded thruster fault scenario in FLFD scheme (Fault injection at t = 1600 sec).



(a) Residual in AUV_1 with respect to AUV_2



(b) Residual in AUV_1 with respect to AUV_3

Figure 3.72: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 8% increase in rotation velocity in flooded thruster fault scenario in FLFD scheme (Fault injection at t = 1600 sec).

- From 1% to 3% increase in rotation velocity, both residual signals do not pass the threshold, which means the FLFD scheme does not detect the fault.
- From 4% to 5% increase in rotation velocity the residual passes the threshold which means according to proposed formation-level fault detection logic the fault is detected. It has been seen that after detection of the fault, in some intervals the samples do not exceed the threshold, these samples are considered as the false healthy detection.
- From 6% to larger percentages increase in rotation velocity, the residuals pass the threshold and the occurred fault can be successfully detected.

In Table 3.14, the fault injection time and fault detection time corresponding to FLFD scheme in flooded thruster fault scenario are expressed.

Table 3.14: Fault injection and detection time in flooded thruster fault scenario for FLFD scheme.

Percentage increase in rotation velocity	Fault injection time(sec)	Fault detection time(sec)
1%	1600	Not detected
2%	1600	Not detected
3%	1600	Not detected
4%	1600	1667
5%	1600	1620
6%	1600	1604
7%	1600	1604
8%	1600	1604

According to this table, it can be concluded that the proposed method is capable of detecting at least 4% increase in rotation velocity in a flooded thruster fault scenario in a short and proper time period. In addition, it has been observed that the more sever flooded thruster faults leads to the larger residuals and smaller fault detection delays.

3.7.2.3 Fault Detection Analysis under Loss of Effectiveness in Rotor Fault Scenarios

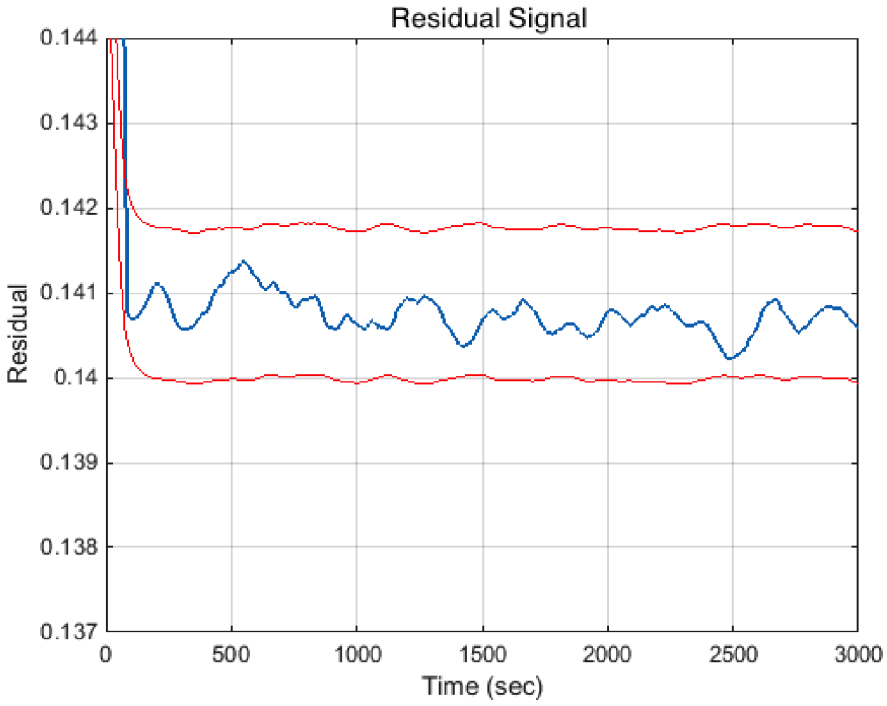
The Rotor Failure fault is considered as zeroing the blade rotation, that cause the thruster does not work. In this work, the loss of effectiveness in rotor is considered as the fault. Hence, in this fault scenario, low rotation velocity condition is considered as a fault in a thruster. In order to simulate this fault scenario, the rotation velocity is dropped by 1% to 8% from its nominal value. These permanent faults are injected to a thruster in the steady state condition at t = 1760 sec (sample # 35200). The residuals corresponding to this fault scenario are illustrated in Figures 3.73 to 3.79 when the fault occurred in AUV_1.

According to the residual signals in Figures 3.73 to 3.79, it can be seen that:

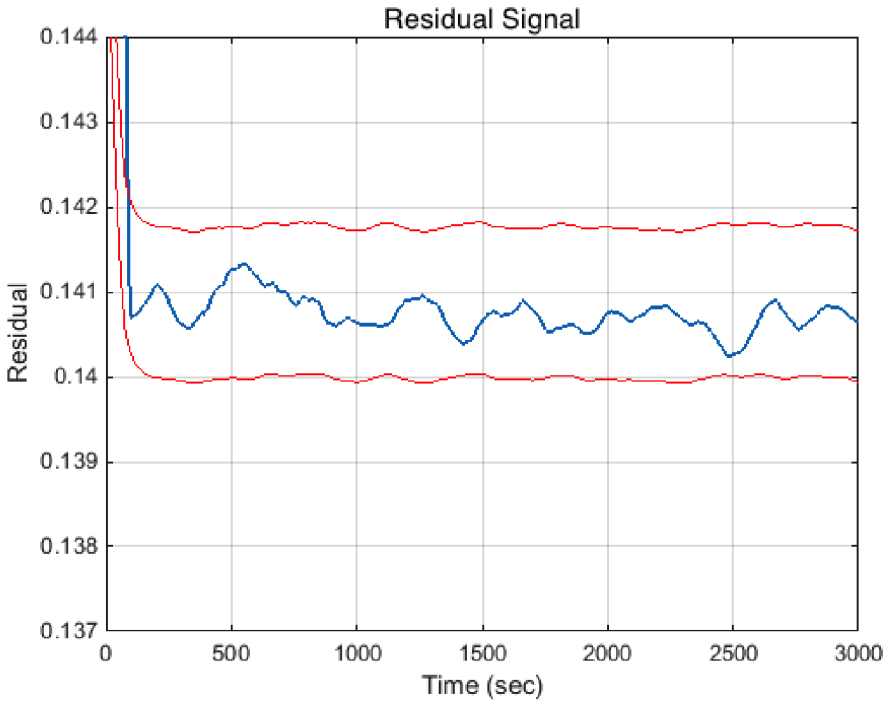
- From 1% to 2% drop in rotation velocity, both residual signals do not pass the threshold, which means the FLFD scheme does not detect the fault.
- Form 3% to 5% drop in rotation velocity the residual signals pass the threshold which means according to proposed formation-level fault detection logic the fault is detected. It has been seen that after detection of the fault, in some intervals the samples do not exceed the threshold, these samples are considered as the false healthy detection.
- From 6% to larger percentages drop in rotation velocity, the residuals pass the threshold and the occurred fault can be successfully detected.

In Table 3.15, the fault injection time and fault detection time corresponding to FLFD scheme in loss of effectiveness in rotor fault scenario are expressed.

According to this table, it can be concluded that the proposed method is capable of detecting at least 3% drop in rotation velocity fault scenario in a short and proper

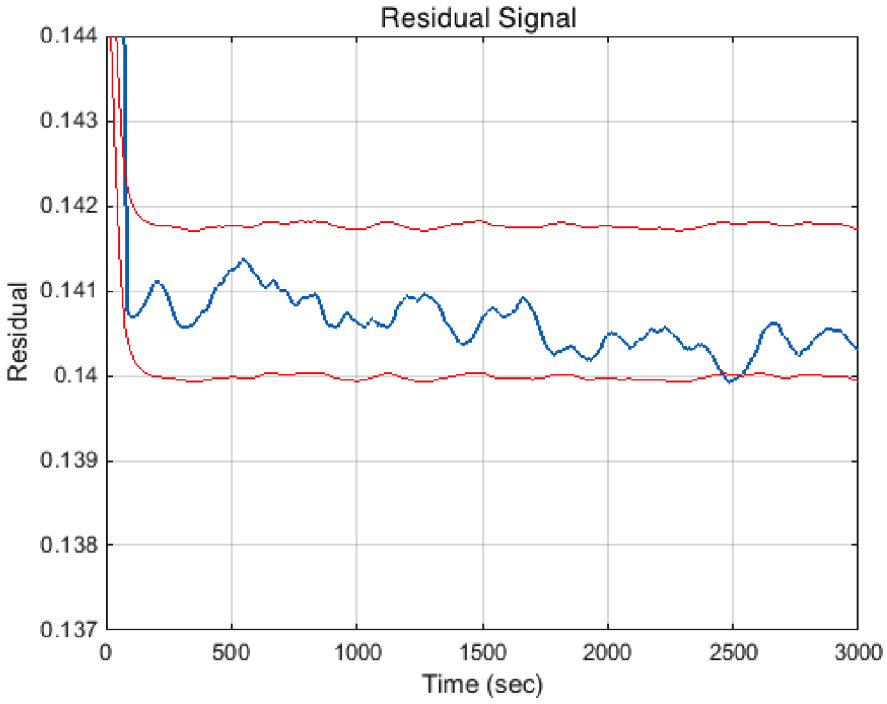


(a) Residual in AUV_1 with respect to AUV_2

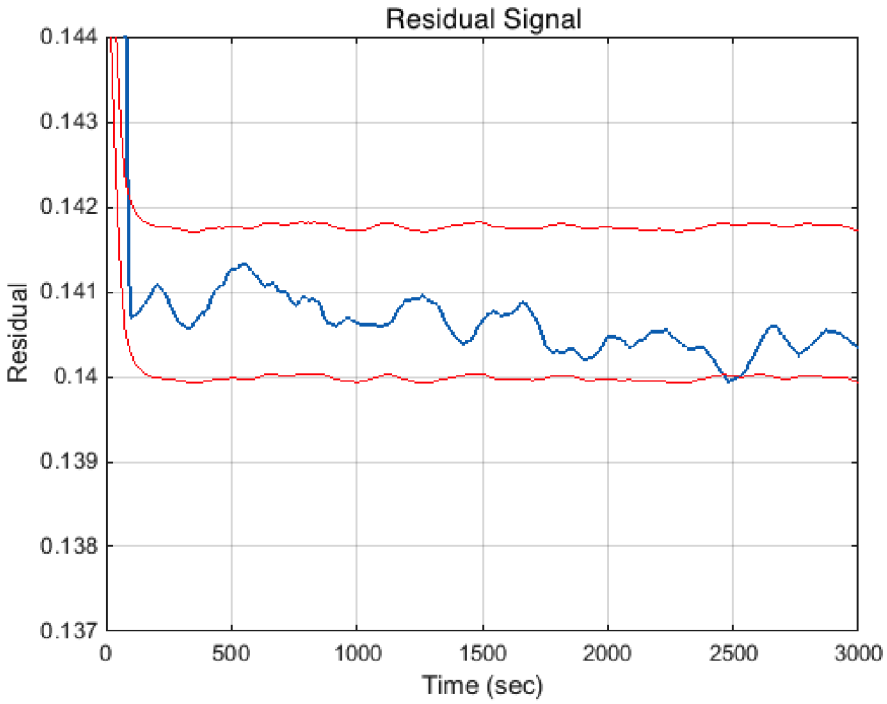


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.73: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 2% drop in rotation velocity in loss of effectiveness in rotor fault scenario in FLFD scheme (Fault injection at t = 1760 sec).

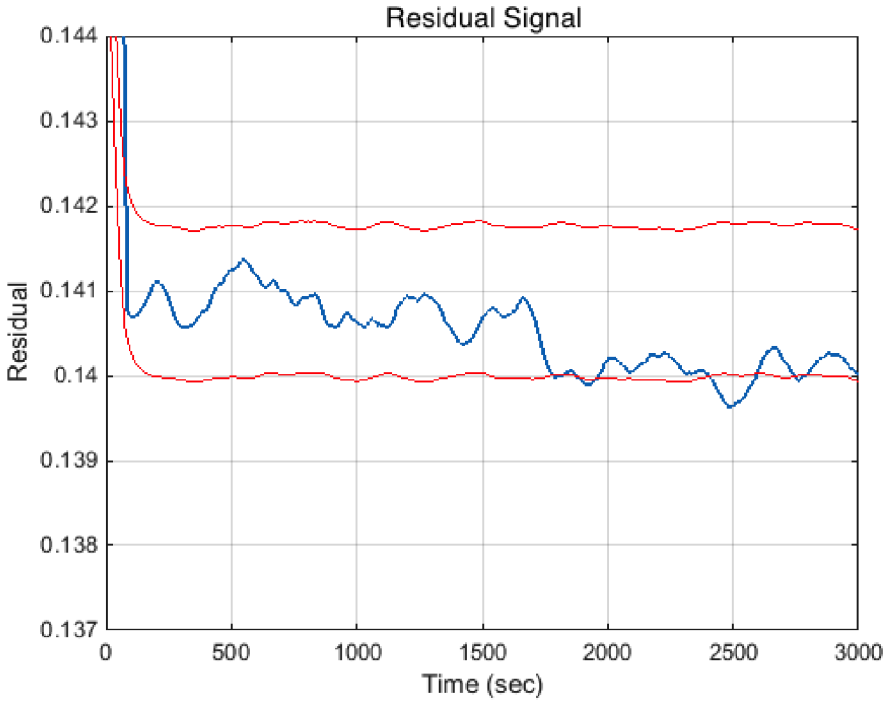


(a) Residual in AUV_1 with respect to AUV_2

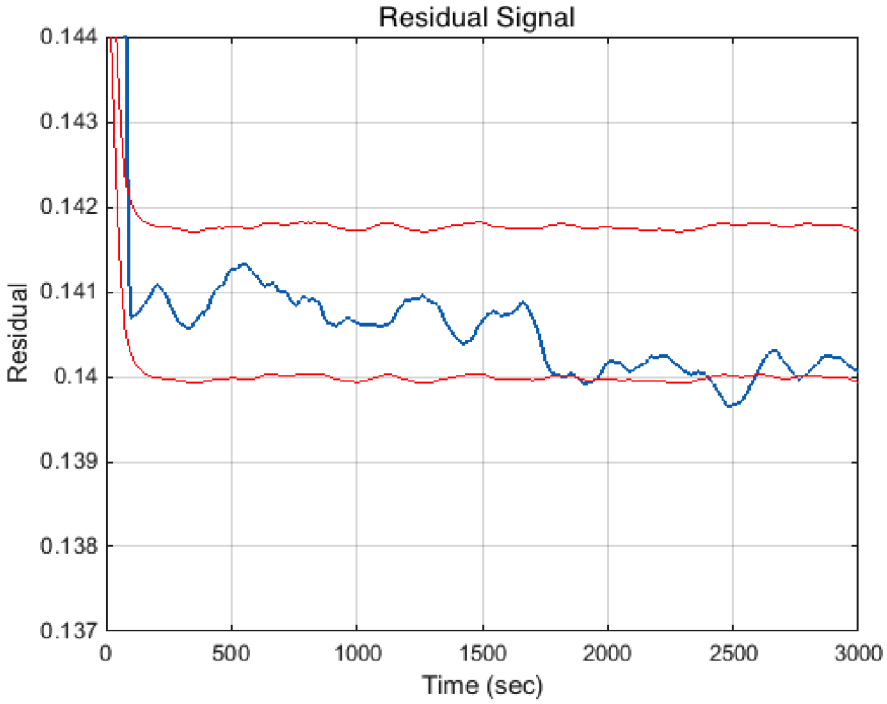


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.74: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 3% drop in rotation velocity in loss of effectiveness in rotor fault scenario in FLFD scheme (Fault injection at t = 1760 sec).

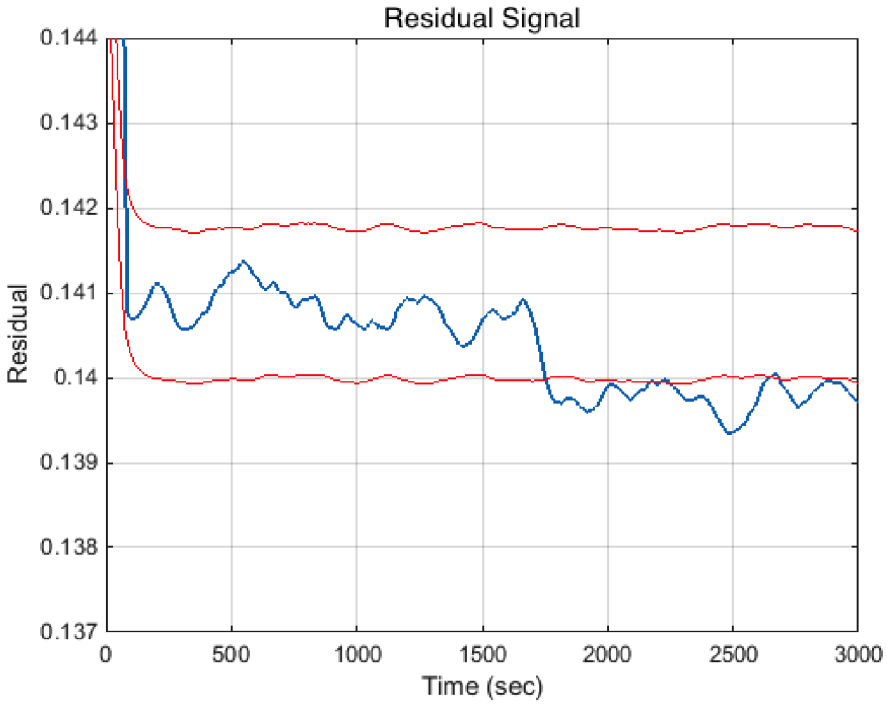


(a) Residual in AUV_1 with respect to AUV_2

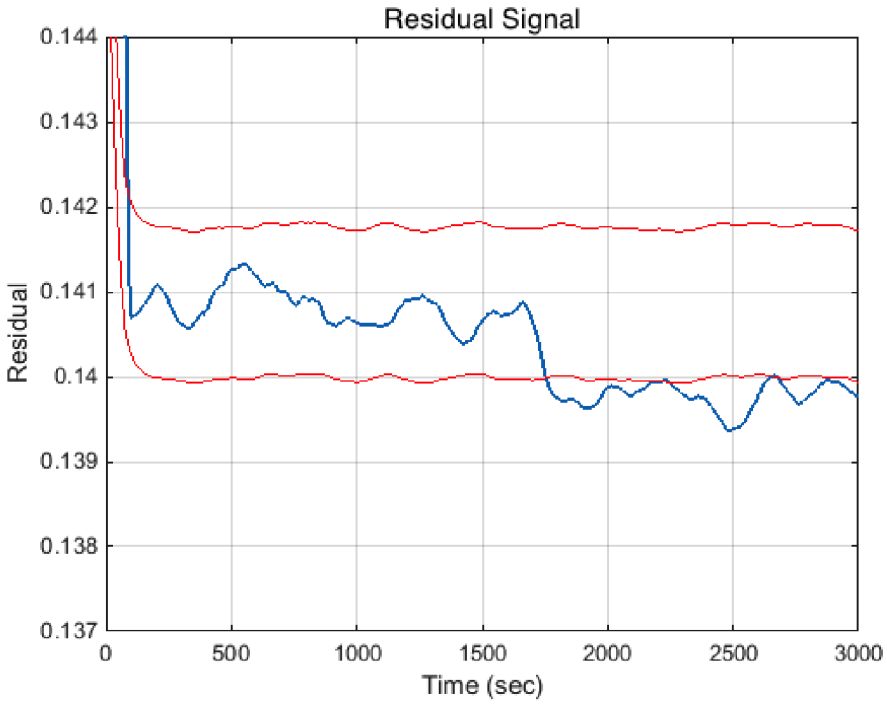


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.75: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 4% drop in rotation velocity in loss of effectiveness in rotor fault scenario in FLFD scheme (Fault injection at t = 1760 sec).

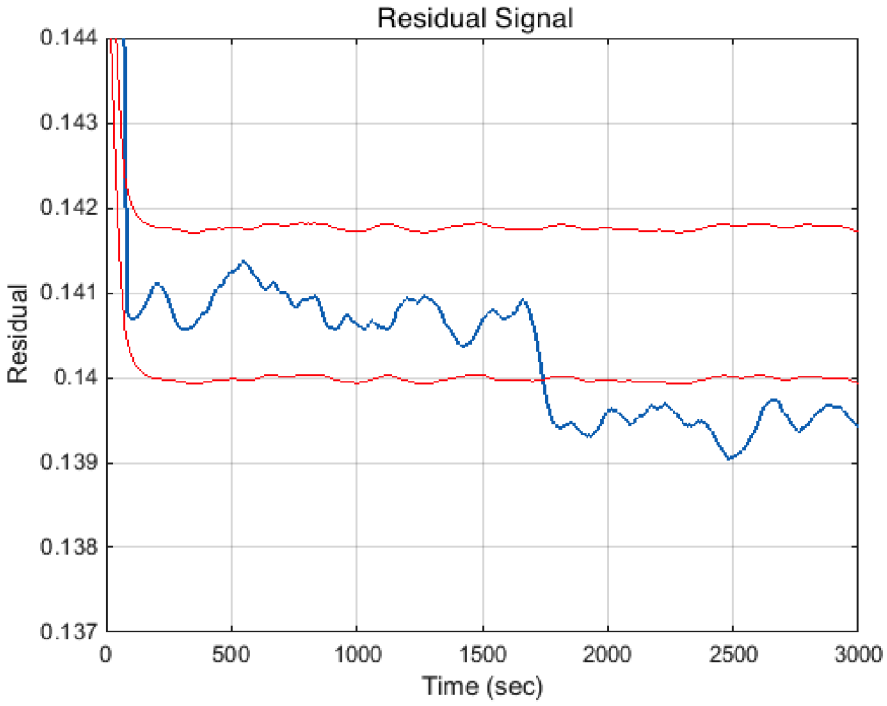


(a) Residual in AUV_1 with respect to AUV_2

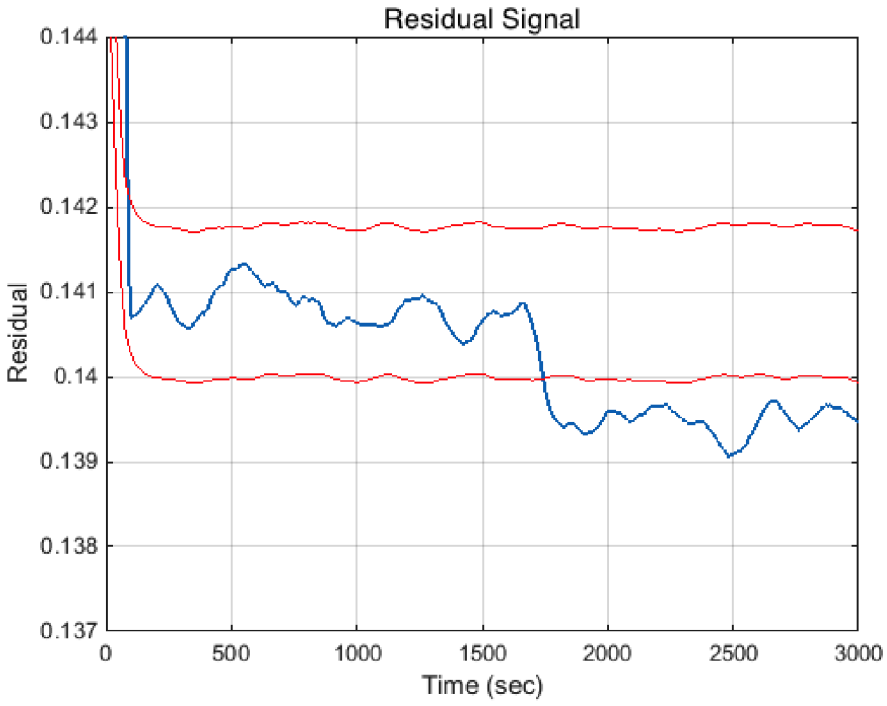


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.76: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 5% drop in rotation velocity in loss of effectiveness in rotor fault scenario in FLFD scheme (Fault injection at t = 1760 sec).

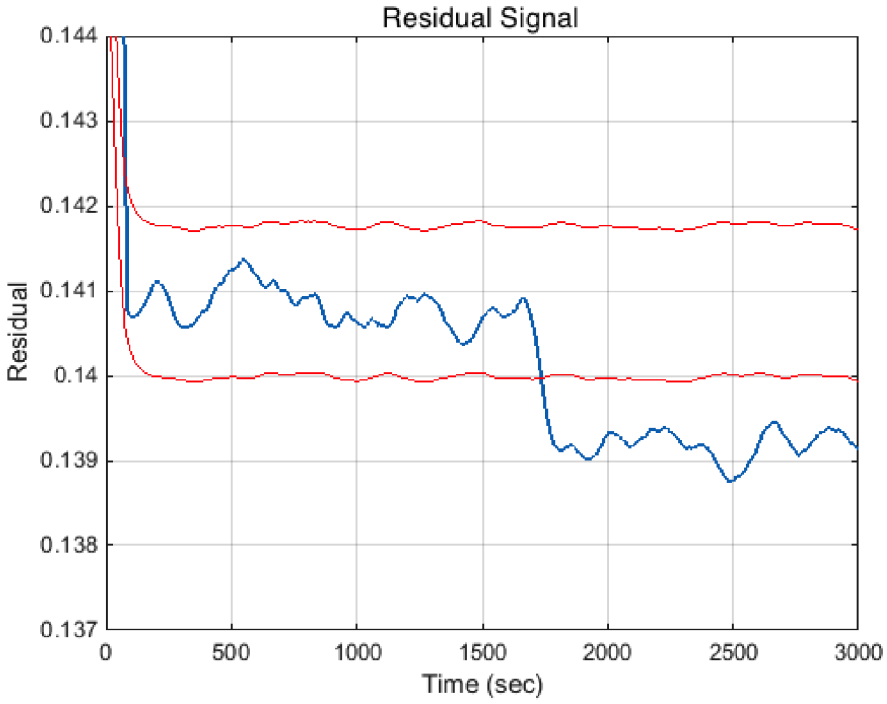


(a) Residual in AUV_1 with respect to AUV_2

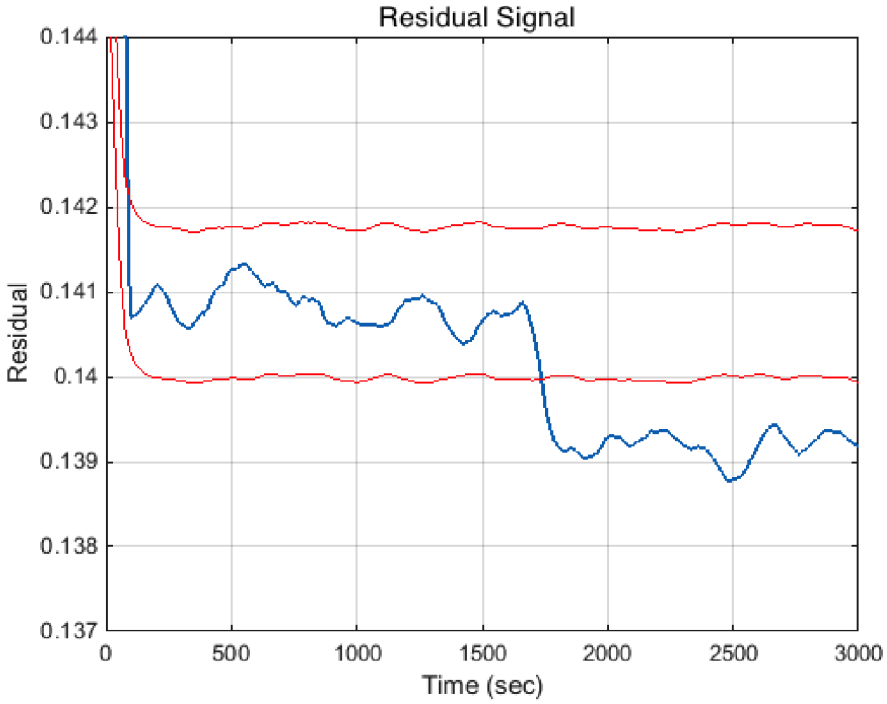


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.77: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 6% drop in rotation velocity in loss of effectiveness in rotor fault scenario in FLFD scheme (Fault injection at t = 1760 sec).

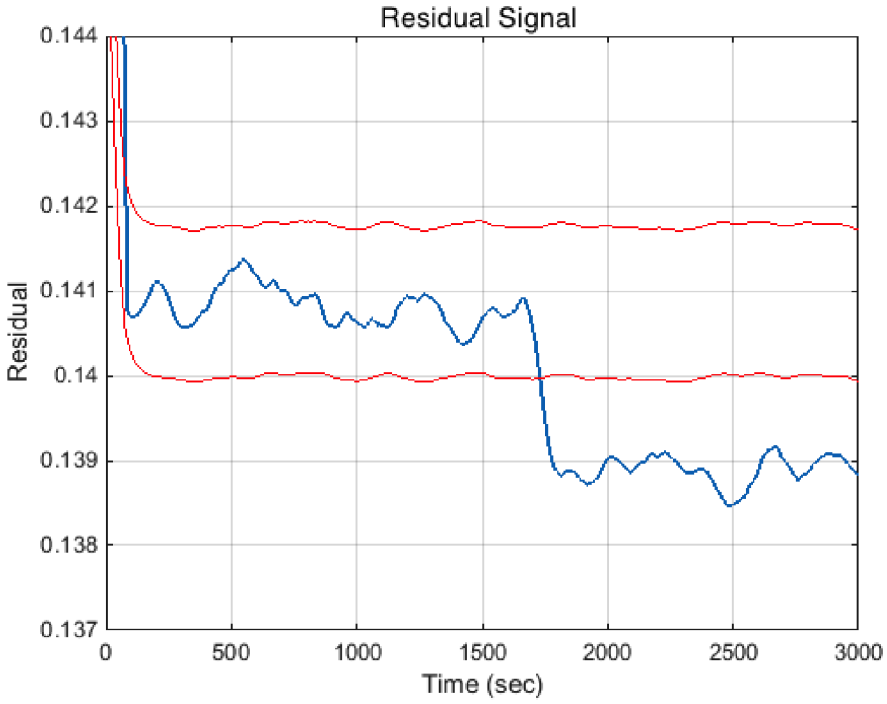


(a) Residual in AUV_1 with respect to AUV_2

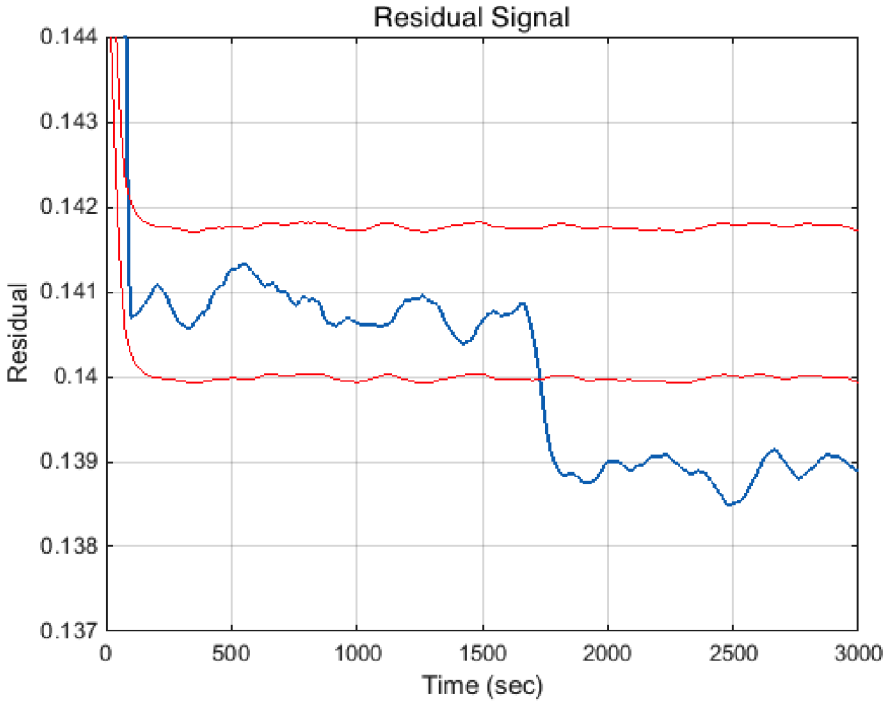


(b) Residual in AUV_1 with respect to AUV_3

Figure 3.78: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 7% drop in rotation velocity in loss of effectiveness in rotor fault scenario in FLFD scheme (Fault injection at t = 1760 sec).



(a) Residual in AUV_1 with respect to AUV_2



(b) Residual in AUV_1 with respect to AUV_3

Figure 3.79: Two residuals in AUV_1 with respect to its neighbors AUV_2 and AUV_3 for 8% drop in rotation velocity in loss of effectiveness in rotor fault scenario in FLFD scheme (Fault injection at t = 1760 sec).

Table 3.15: Fault injection time and detection time in loss of effectiveness in rotor fault scenario for FLFD scheme.

Percentage drop in rotation velocity	Fault injection time(sec)	Fault detection time(sec)
1%	1760	Not detected
2%	1760	Not detected
3%	1760	2478
4%	1760	1873
5%	1760	1769
6%	1760	1765
7%	1760	1765
8%	1760	1765

time period. In addition, it has been observed that the more sever loss of effectiveness in rotor faults leads to the larger residuals and smaller fault detection delays.

3.8 Confusion Matrix Analysis for Fault Detection

In order to evaluate the performance of the proposed fault detection algorithm, the confusion matrix approach is used. A confusion matrix consists of four elements, namely the true positive, the true negative, the false positive, and the false negative which are depicted in Table 3.16.

Table 3.16: Table corresponding to confusion matrix [172].

		Predicted	
		Faulty	Healthy
Actual	Faulty	TP	FN
	Healthy	FP	TN

The aforementioned elements of the confusion matrix are defined as follows:

- True positive (TP): The number of samples detected as faulty while the AUV is operating in the faulty mode.

- True negative (TN): The number of samples detected as healthy while the AUV is operating in the healthy mode.
- False negative (FN): The number of samples detected as healthy while the AUV is operating in the faulty mode.
- False positive (FP): The number of samples detected as faulty while the AUV is operating in the healthy mode.

For each fault scenario a confusion matrix is calculated and the parameters named as accuracy, precision, detection rate (True faulty rate) and false alarm rate (False faulty rate) are calculated to evaluate the performance of the fault detection scheme. These four parameters are defined as follows:

- Accuracy: It is the percentage of the prediction that are correct and it indicates the overall effectiveness of the fault detection scheme. The accuracy is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: It is the percentage of the positive (faulty mode) prediction that are correct. The precision is computed as follows:

$$Precision = \frac{TP}{TP + FP}$$

- Detection rate (True faulty rate): It measures the portion of actual faulty modes of the AUV which are correctly identified as faulty. The detection rate shows the effectiveness of the fault detection scheme to identify the faulty mode of the AUV and it is expressed as:

$$Detection\ rate\ (True\ faulty\ rate) = \frac{TP}{TP + FN}$$

- False alarm rate (False faulty rate): It reports the portion of the healthy samples which are misclassified as the faulty. The false alarm rate is calculated according to the following formula:

$$False\ alarm\ rate\ (False\ faulty\ rate) = \frac{FP}{TN + FP}$$

The confusion matrix for each of the fault scenarios in ALFD and FLFD schemes are calculated and the performances of these two methods are compared.

3.8.1 Confusion Matrix Analysis for Thruster Blocking Fault Scenario

The confusion matrix for thruster blocking fault scenarios in ALFD and FLFD schemes are illustrated in Tables 3.17 and 3.18 respectively.

Table 3.17: Confusion matrix for thruster blocking fault scenario in ALFD scheme

Percentage drop in thruster torque	TP	TN	FN	FP
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	N/A	N/A	N/A
4%	N/A	N/A	N/A	N/A
5%	1468	45999	12533	0
6%	2055	45999	11946	0
7%	2801	45999	11200	0
8%	4241	45999	9760	0
9%	5881	45999	8120	0
10%	8468	45999	5533	0
11%	10995	45999	3006	0
12%	13941	45999	60	0
13%	13961	45999	40	0
14%	13961	45999	40	0
15%	13961	45999	40	0
16%	13961	45999	40	0

According to the confusion matrix elements, the accuracy, precision, detection

Table 3.18: Confusion matrix for thruster blocking fault scenario in FLFD scheme

Percentage drop in thruster torque	TP	TN	FN	FP
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	12469	34699	12831	0
4%	20033	34699	5267	0
5%	25200	34699	100	0
6%	25200	34699	100	0
7%	25200	34699	100	0
8%	25200	34699	100	0
9%	25220	34699	80	0
10%	25220	34699	80	0
11%	25240	34699	60	0
12%	25240	34699	60	0
13%	25240	34699	60	0
14%	25260	34699	40	0
15%	25260	34699	40	0
16%	25260	34699	40	0

rate and false alarm rate for ALFD and FLFD schemes are calculated in Tables 3.19 and 3.20.

As the results in Tables 3.19 and 3.20 show the proposed ALFD scheme is capable of detecting 10% drop in the thruster torque with 90.77% accuracy, 100% precision, 78.53% detection rate and 0% false alarm rate, while the FLFD scheme is capable of detecting 5% drop in the thruster torque with 99.83% accuracy, 100% precision, 99.60% detection rate and 0% of false alarm rate. According to theses parameters it can be concluded that the our FLFD scheme is capable of detecting lower severity thruster blocking faults in AUV's thruster and it has higher accuracy and detection rate comparing to agent-level fault detection scheme.

Table 3.19: Accuracy and Precision for thruster blocking fault scenario.

	Accuracy		Precision	
Percentage drop in thruster torque	ALFD	FLFD	ALFD	FLFD
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	78.61%	N/A	100%
4%	N/A	91.22%	100%	100%
5%	79.11%	99.83%	100%	100%
6%	80.09%	99.83%	100%	100%
7%	81.33%	99.83%	100%	100%
8%	83.73%	99.83%	100%	100%
9%	86.46%	99.86%	100%	100%
10%	90.77%	99.86%	100%	100%
11%	94.99%	99.89%	100%	100%
12%	99.90%	99.89%	100%	100%
13%	99.93%	99.89%	100%	100%
14%	99.93%	99.93%	100%	100%
15%	99.93%	99.93%	100%	100%
16%	99.93%	99.93%	100%	100%

Table 3.20: Detection rate and False alarm rate for thruster blocking fault scenario.

	Detection rate		False alarm rate	
Percentage drop in thruster torque	ALFD	FLFD	ALFD	FLFD
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	49.28%	N/A	0%
4%	N/A	79.18%	N/A	0%
5%	10.48%	99.60%	0%	0%
6%	14.67%	99.60%	0%	0%
7%	20%	99.60%	0%	0%
8%	30.29%	99.60%	0%	0%
9%	60.48%	99.68%	0%	0%
10%	78.53%	99.68%	0%	0%
11%	99.57%	99.76%	0%	0%
12%	99.71%	99.76%	0%	0%
13%	99.71%	99.76%	0%	0%
14%	99.71%	99.84%	0%	0%
15%	99.71%	99.84%	0%	0%
16%	99.71%	99.84%	0%	0%

3.8.2 Confusion Matrix Analysis for Flooded Thruster Fault Scenario

The confusion matrix for flooded thruster fault scenario in ALFD and FLFD scheme are illustrated in Tables 3.21 and 3.22 respectively.

Table 3.21: Confusion matrix for flooded thruster fault scenario in ALFD scheme

Percentage increase in rotation velocity	TP	TN	FN	FP
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	N/A	N/A	N/A
4%	N/A	N/A	N/A	N/A
5%	N/A	N/A	N/A	N/A
6%	N/A	N/A	N/A	N/A
7%	N/A	N/A	N/A	N/A
8%	N/A	N/A	N/A	N/A
9%	2628	51499	5873	0
10%	3915	51499	4586	0
11%	5921	51499	2580	0
12%	6828	51499	1673	0
13%	7881	51499	620	0
14%	7961	51499	540	0
15%	8081	51499	420	0
16%	8241	51499	260	0

According to confusion matrix elements, the accuracy, precision, detection rate and false alarm rate for ALFD and FLFD scheme are calculated in Tables 3.23 and 3.24.

As the results in Tables 3.23 and 3.24 show the proposed ALFD scheme is capable of detecting 12% increase in rotation velocity with 97.21% accuracy, 100% precision, 80.31% detection rate and 0% false alarm rate, while the FLFD scheme is capable of detecting 5% increase in rotation velocity with 96.41% accuracy, 100% precision, 92.30% detection rate and 0% false alarm rate. According to theses parameters it can be concluded that the our FLFD scheme is capable of detecting lower severity

Table 3.22: Confusion matrix for flooded thruster fault scenario in FLFD scheme.

Percentage increase in rotation velocity	TP	TN	FN	FP
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	N/A	N/A	N/A
4%	17132	31999	10869	0
5%	25847	31999	2154	0
6%	27921	31999	80	0
7%	27921	31999	80	0
8%	27921	31999	80	0
9%	27921	31999	80	0
10%	27941	31999	60	0
11%	27941	31999	60	0
12%	27941	31999	60	0
13%	27961	31999	40	0
14%	27961	31999	40	0
15%	27961	31999	40	0
16%	27961	31999	40	0

Table 3.23: Accuracy and Precision for flooded thruster fault scenario.

	Accuracy		Precision	
Percentage increase in rotation velocity	ALFD	FLFD	ALFD	FLFD
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	N/A	N/A	N/A
4%	N/A	81.88%	N/A	100%
5%	N/A	96.41%	N/A	100%
6%	N/A	99.86%	N/A	100%
7%	N/A	99.86%	N/A	100%
8%	N/A	99.86%	100%	100%
9%	90.21%	99.86%	100%	100%
10%	92.35%	99.90%	100%	100%
11%	95.70%	99.90%	100%	100%
12%	97.21%	99.90%	100%	100%
13%	98.96%	99.93%	100%	100%
14%	99.10%	99.93%	100%	100%
15%	99.30%	99.93%	100%	100%
16%	99.56%	99.93%	100%	100%

Table 3.24: Detection rate and false alarm rate for flooded thruster fault scenario.

	Detection rate		False alarm rate	
Percentage increase in rotation velocity	ALFD	FLFD	ALFD	FLFD
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	N/A	N/A	N/A
4%	N/A	61.18%	N/A	0%
5%	N/A	92.30%	N/A	0%
6%	N/A	99.71%	N/A	0%
7%	N/A	99.71%	N/A	0%
8%	N/A	99.71%	0%	0%
9%	30.91%	99.71%	0%	0%
10%	46.05%	99.78%	0%	0%
11%	69.65%	99.78%	0%	0%
12%	80.31%	99.78%	0%	0%
13%	92.70%	99.85%	0%	0%
14%	93.64%	99.85%	0%	0%
15%	95.05%	99.85%	0%	0%
16%	96.94%	99.85%	0%	0%

flooded thruster faults in AUV's thruster and it has higher accuracy and detection rate comparing to agent-level fault detection scheme. .

3.8.3 Confusion Matrix Analysis for Loss of Effectiveness in Rotor Fault Scenario

The confusion matrix for loss of effectiveness in ALFD and FLFD scheme are illustrated in Tables 3.25 and 3.26 respectively.

According to confusion matrix elements, the accuracy, precision, detection rate and false alarm rate for ALFD and FLFD scheme are calculated in Tables 3.27 and 3.28.

As the results in Tables 3.27 and 3.28 show the proposed ALFD scheme is capable of detecting 13% decrease in rotation velocity with 99.39% accuracy, 100% precision

Table 3.25: Confusion matrix for loss of effectiveness in rotor fault scenario in ALFD scheme

Percentage drop in rotation velocity	TP	TN	FN	FP
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	N/A	N/A	N/A
4%	N/A	N/A	N/A	N/A
5%	1241	47799	10960	0
6%	1708	47799	10493	0
7%	2255	47799	9946	0
8%	2935	47799	9266	0
9%	4428	47799	7773	0
10%	5955	47799	6246	0
11%	7161	47799	5040	0
12%	8615	47799	3586	0
13%	11781	47799	420	0
14%	11921	47799	280	0
15%	12001	47799	200	0
16%	12121	47799	80	0

Table 3.26: Confusion matrix for loss of effectiveness in rotor fault scenario in FLFD scheme

Percentage drop in rotation velocity	TP	TN	FN	FP
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	1372	35199	23429	0
4%	4825	35199	19976	0
5%	23001	35199	1800	0
6%	24701	35199	100	0
7%	24701	35199	100	0
8%	24701	35199	100	0
9%	24721	35199	80	0
10%	24721	35199	80	0
11%	24721	35199	80	0
12%	24741	35199	60	0
13%	24741	35199	60	0
14%	24741	35199	60	0
15%	24761	35199	40	0
16%	24761	35199	40	0

Table 3.27: Accuracy and Precision for loss of effectiveness in rotor fault scenario.

	Accuracy		Precision	
Percentage drop in rotation velocity	ALFD	FLFD	ALFD	FLFD
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	60.95%	N/A	20%
4%	N/A	66.70%	N/A	56.67%
5%	81.73%	97%	26.67%	100%
6%	82.51%	99.83%	43.34%	100%
7%	83.42%	99.83%	76.67%	100%
8%	84.55%	99.83%	93.34%	100%
9%	87.04%	99.86%	100%	100%
10%	89.59%	99.86%	100%	100%
11%	91.60%	99.86%	100%	100%
12%	94.02%	99.90%	100%	100%
13%	99.30%	99.90%	100%	100%
14%	99.53%	99.90%	100%	100%
15%	99.60%	99.93%	100%	100%
16%	99.86%	99.93%	100%	100%

Table 3.28: Detection rate and False alarm rate for loss of effectiveness in rotor fault scenario.

	Detection rate		False alarm rate	
Percentage drop in rotation velocity	ALFD	FLFD	ALFD	FLFD
1%	N/A	N/A	N/A	N/A
2%	N/A	N/A	N/A	N/A
3%	N/A	5.53%	N/A	0%
4%	N/A	19.45%	N/A	0%
5%	10.17%	92.74%	0%	0%
6%	13.99%	99.59%	0%	0%
7%	18.48%	99.59%	0%	0%
8%	24.05%	99.59%	0%	0%
9%	36.29%	99.67%	0%	0%
10%	48.80%	99.67%	0%	0%
11%	58.69%	99.67%	0%	0%
12%	70.60%	99.75%	0%	0%
13%	96.55%	99.75%	0%	0%
14%	97.70%	99.75%	0%	0%
15%	98.36%	99.83%	0%	0%
16%	99.34%	99.83%	0%	0%

, 96.55% detection rate and 0% false alarm rate, while the FLFD scheme is capable of detecting 5% decrease in rotation velocity with 97% accuracy, 100% precision, 92.74% detection rate and 0% false alarm rate. According to theses parameters it can be concluded that the our FLFD scheme is capable of detecting lower severity loss of effectiveness in rotor faults in AUV's thruster and it has higher accuracy and detection rate comparing to agent-level fault detection scheme. .

3.9 Conclusion

In this chapter, the agent-level fault detection (ALFD) and formation-level fault detection (FLFD) schemes for detecting the faults in the thruster of the AUV is proposed and explained. Our fault detection schemes are based on dynamic neural network (DNN) that is trained with genetic algorithm (GA). In the proposed ALFD scheme only absolute measurements are considered to use in DNN as a historic data to train the DNN. However, in our FLFD we also use relative measurements between each AUV and its neighbors to train the DNNs. The confusion matrix analysis is performed for all of the fault scenarios. In thruster blocking fault scenario, the the ALFD scheme is capable of detecting 10% drop in thruster torque with 90% accuracy and 78.30% detection rate, while the FLFD scheme is capable of detecting 5% drop in thruster torque with 99.83% accuracy and 99.60% detection rate. In flooded thruster fault scenario, the ALFD scheme is capable of detecting 12% increase in rotation velocity with 97.21% accuracy and 80.31% detection rate, while the FLFD scheme is capable of detecting 5% increase in rotation velocity with 96.41% accuracy and 92.30% detection rate. In loss of effectiveness in rotor fault scenario, the ALFD scheme is capable of detecting 13% decrease in rotation velocity with 99.39% accuracy and 96.55% detection rate, while the FLFD scheme is capable of detecting 5%

decrease in rotation velocity with 97% accuracy and 92.74% detection rate. According to the aforementioned results it can be concluded that the FLFD can detect the lower severity faults in thruster of the AUV with high level of accuracy and precision and has better performance in comparison with ALFD scheme. Hence, utilizing the relative information in each AUV to train the DNN, yields the better results in fault detection comparing to ALFD scheme. In addition, our proposed FLFD scheme is capable of detecting the faulty AUV in a formation by means of the fault detection unit in the AUV.

The performance of the extended dynamic back-propagation (EDBP) and genetic algorithm (GA) in training of the dynamic neural network is compared. The presented results indicate that the GA approach converge faster in comparison with the EDBP, in other word it provides better results in less number of iterations. In addition, based on several experiments it has been seen that the GA requires less data for training the DNN comparing to EDBP. Thus, this method can be applied for other applications that the availability of data for training is a concern.

In next chapter, the fault isolation and identification problem will be discussed and the results for fault isolation and identification will be provided as well.

Chapter 4

Agent-Level and Formation-Level Fault Isolation and Identification Schemes

In Chapter 3, a fault detection scheme for formation of AUVs was developed. A fault detection unit in each AUV including two dynamic neural networks is employed to accomplish the fault detection. In the proposed method when a fault occurs in a thruster of one of the AUVs in the formation, the fault detection unit in the faulty AUV can detect the fault after a delay. When the fault is detected, in the next step the type and severity of the fault should be determined. The objective of this chapter is isolation and identification of the thruster faults. In this chapter the process of determining the type of the fault in the thruster is considered as fault isolation and indicating the severity of the fault is considered as the fault identification.

4.1 Fault Isolation Schemes

As shown in previous chapter, various fault scenarios were injected into the thruster of the AUV and the detection of the faults in both agent-level and formation-level schemes has been successfully fulfilled. In the proposed fault isolation scheme, a multi-layer perceptron neural network (MLPNN) is employed in each of the AUVs after the neural observer to categorize the type of the fault into thruster blocking, flooded thruster and loss of effectiveness in rotor. In this section the proposed fault isolation schemes in agent-level and formation-level for determining the type of the occurred thruster fault of the AUV are presented.

4.1.1 Agent-Level Fault Isolation Scheme

In proposed agent-level fault isolation scheme, the residual signals which are generated in the agent-level fault detection scheme are processed and applied as an input to the MLPNN. In this work processing of the residual signal is calculating the magnitudes of the residual signal before and after the occurrence of the fault which are applied as the two inputs for the MLPNN. According to several experiments, it has been observed that the aforementioned inputs are valuable to fulfill the fault isolation. The output of the proposed MLPNN is the fault label corresponding to the type of the occurred fault in the thruster of the AUV. The structure of the proposed agent-level fault isolation scheme is depicted in Figure 4.1.

As it is shown in Figure 4.1, in the first step the preprocessing unit evaluate the residual signals and provides the numerical values for using as inputs for the MLPNN. The output of the MLPNN indicates the class of the occurred fault. The assigned classes for fault types in agent-level fault isolation scheme is expressed in Table 4.1.

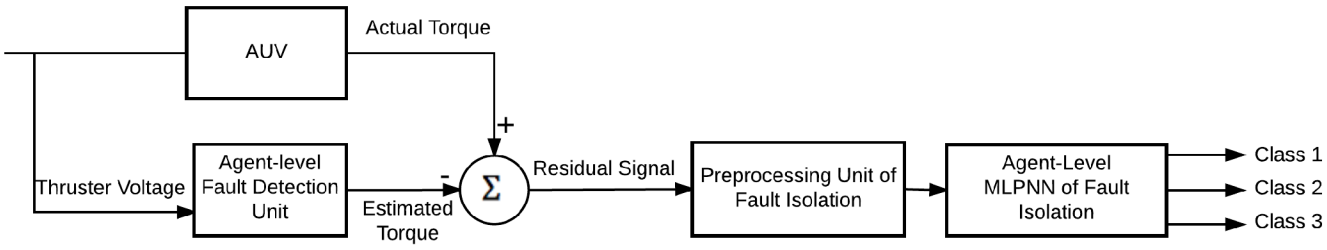


Figure 4.1: Structure of the agent-level fault isolation scheme.

Table 4.1: Assigned classes for fault types in agent-level fault isolation scheme.

Fault Class (Type)	Assigned Label
Class 1 (Thruster blocking)	0 0 1
Class 2 (Flooded thruster)	0 1 0
Class 3 (Loss of effectiveness in rotor)	1 0 0

4.1.2 Formation-Level Fault Isolation Scheme

In proposed formation-level fault isolation scheme, the residual signals which are generated in the formation-level fault detection scheme are processed and applied as an input to the MLPNN. Since it is assumed that the two adjacent neighbors of the faulty AUV are identical and working in healthy mode, there is no significant difference between the two generated residual signals, therefore only one of them is used for fault isolation purpose. In this work processing of the residual signal is calculating the magnitudes of the residual signal before and after the occurrence of the fault which are applied as the two inputs for the MLPNN. According to several experiments, it has been observed that the aforementioned inputs are valuable to fulfill the fault isolation. The output of the proposed MLPNN is the fault label corresponding to the type of the occurred fault in the thruster of the AUV. The structure of the proposed formation-level fault isolation scheme is depicted in Figure 4.2.

As it is shown in Figure 4.2, in the first step the preprocessing unit evaluates the

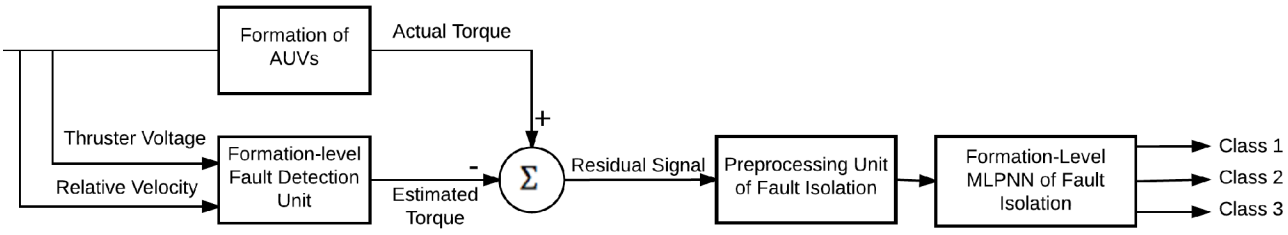


Figure 4.2: Structure of the formation-level fault isolation scheme.

residual signals and provides the numerical values as inputs for the MLPNN. The output of the MLPNN indicates the class of the occurred fault. The assigned classes for fault types in formation-level fault isolation scheme is expressed in Table 4.2.

Table 4.2: Assigned classes for fault types in formation-level fault isolation scheme..

Fault Class (Type)	Assigned Label
Class 1 (Thruster blocking)	0 0 1
Class 2 (Flooded thruster)	0 1 0
Class 3 (Loss of effectiveness in rotor)	1 0 0

4.1.3 Data Preprocessing

In agent-level and formation-level fault isolation schemes, providing the data for training the MLPNNs is divided into different steps. Firstly, the residual signals which are generated through the agent-level and formation-level fault detection schemes under various faulty operating condition of the AUV are collected. Secondly, these residuals are evaluated in the preprocessing unit and the magnitudes of them before and after the fault occurrence is calculated and they are considered as the inputs of the MLPNNs. Finally, these magnitudes which are the inputs for the MLPNNs are normalized in the range of [0,1] before the training process begins. In order to have a reliable performance, 10 different input data sets are provided. Each data sets includes 3600 samples including 1200 thruster blocking fault scenarios, 1200 flooded

thruster fault scenarios and 1200 loss of effectiveness in rotor fault scenarios. The input data set is randomly divided into three parts namely as, training set, validation set and testing set. In this work, the 50% of the input data set is randomly chosen as training set, the 25% of it is selected at random as validation set and the remaining 25% of it is the testing set. The training set is used to find the optimal weights for the MLPNN, the validation set is utilized to determine the stopping point for training and avoid over training and the testing set is used to estimate the error rate after the final model of the MLPNN is selected. It has been observed that by using 3600 samples as input data set an acceptable result for fault isolation task can be achieved.

4.1.4 Training Phase

The MLPNNs in both agent-level and formation-level fault isolation schemes are trained with genetic algorithm (GA). The GA is applied to adjust the connection weights of the MLPNNs. The training procedure of the MLPNN with GA is fully explained in the following steps.

- First step (Initial Population): In this step the initial population consists of N_{pop} chromosomes are chosen randomly. It should be considered that no replication is allowed in the population, in other words all of the chromosomes should be distinct. Each of the chromosomes includes N_{var} which represent all the connection weights of the MLPNN. If we assume that the MLPNN has the total of N connection weights, then the l^{th} chromosome of the population can be written as:

$$Chromosome_l = [w_{1l}, w_{2l}, \cdots, w_{Nl}] \tag{4.1.1}$$

  In equation (4.1.1) the w denotes to the connection weight of the MLPNN.

- Second step (Fitness Function): In this step the training set is applied to the

MLPNN and the value of the fitness function for each of the chromosomes in the population is calculated. In this thesis the root mean squared error is considered as the fitness function which is expressed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2} \tag{4.1.2}$$

The proposed fitness function is the error between the desired output and the current output. In equation (4.1.2), $\widehat{y_i}$ and y_i denote the estimated output and desired output respectively and the n is the number of input data.

- Third step (Termination criterion): In this stage the termination criteria should be checked, if it reaches the desired level the training stops and returns the chromosomes which includes the optimal weights of the MLPNN and if it dose not reach the desired level, the next step begins. In this work, the termination criteria is related to the RMSE value. If the $RMSE < Desired\ Value$ then the training stops and the optimal connection weights of the MLPNN returns.

- Forth step (Selection): In this step, firstly the chromosomes are ranked from highest to lowest based on their RMSE value. Then only the 50% of the chromosomes with highest rank are survived to the next generation and the remaining are eliminated. It should be considered that the size of the population is fixed, therefore the eliminated chromosomes are replaced with the offsprings that are produced in the next steps.

- Fifth step (Crossover): In this step, firstly two chromosomes from the population are selected to produce the offsprings. It is assumed that the i^{th} and j^{th} chromosome are selected. Secondly, a connection weight in both chromosomes is selected to be the crossover point. The crossover point selection process is

expressed as follows:

$$\alpha = roundup\left\{random * N\right\} \tag{4.1.3}$$

The α in equation (4.1.3) denotes the randomly selected connection weight in both chromosomes. Therefore, the two chromosomes that are chosen for generating new offsprings can be written as:

$$\begin{aligned} Chromosome_i &= [w_{1i}, w_{2i} \cdots, w_{\alpha i}, \cdots, w_{Ni}] \\ Chromosome_j &= [w_{1j}, w_{2j} \cdots, w_{\alpha j}, \cdots, w_{Nj}] \end{aligned} \tag{4.1.4}$$

In the next step the selected connection weights in each chromosomes (i.e. $w_{\alpha i}$ and $w_{\alpha j}$) are replaced with the new values based on following equation.

$$\begin{aligned} w_{newi} &= w_{\alpha i} - \beta[w_{\alpha i} - w_{\alpha j}] \\ w_{newj} &= w_{\alpha j} - \beta[w_{\alpha i} - w_{\alpha j}] \end{aligned} \tag{4.1.5}$$

where β is also a random value between 0 and 1. The final step is to produce the offsprings for the population by replacing the new connection weights in the chromosomes and swapping the connection weights of the two selected chromosomes at the crossover point. The generated offsprings are expressed as follows:

$$\begin{aligned} offspring_1 &= [w_{1i}, w_{2i}, \cdots, w_{newi}, \cdots w_{Nj}] \\ offspring_2 &= [w_{1j}, w_{2j}, \cdots, w_{newj}, \cdots w_{Ni}] \end{aligned} \tag{4.1.6}$$

- Sixth step (Mutation): In order to avoid trapping into the local minima areas, the mutation is applied. For mutation at the first step the mutation rate is chosen which is a number between 0 and 1. Secondly, at random the weight connection of chromosomes are selected to be mutated. Finally, the chosen

connection weights are replaced by a new random value.

- Seventh step: In this stage the new population is generated and the algorithm returns to third step.

The flowchart of training the MLPNN with genetic algorithm is illustrated in Figure 4.3. As it is shown in the Figure 4.3, before the training process of the MLPNN with GA starts, the structure of the MLPNN including the number of layers and the number of neurons in each layer should be determined.

4.1.4.1 Training Phase Results in Agent-Level

The MLPNN in agent-level fault isolation scheme is trained with 10 training sets which are obtained through different faulty operating condition of the AUV. Each of the training sets includes 1800 samples. The process of generating training sets is fully explained in section 4.1.3. As it is mentioned, in the first step the structure of the MLPNN has been selected. Structure of the MLPNN includes, the number of layers and the number of neurons in each layer. In order to determine the structure of the MLPNN, we start with a small network structure, then the number of neurons and hidden layers are increased till the optimum structure of the network is achieved with respect to the MLPNN performance. In this work, according to several experiments it has been observed that utilizing two hidden layer for the MLPNN in agent-level fault isolation scheme, provides an acceptable performance for the network. The optimum structure of the MLPNN and its specifications for agent-level fault isolation scheme is shown in Table 4.3.

In Table 4.3 the four successive numbers in the structure of the network are the number of neurons in the input layer, first hidden layer, second hidden layer and output layer respectively.

As it is mentioned in training phase the variables of each chromosomes in the

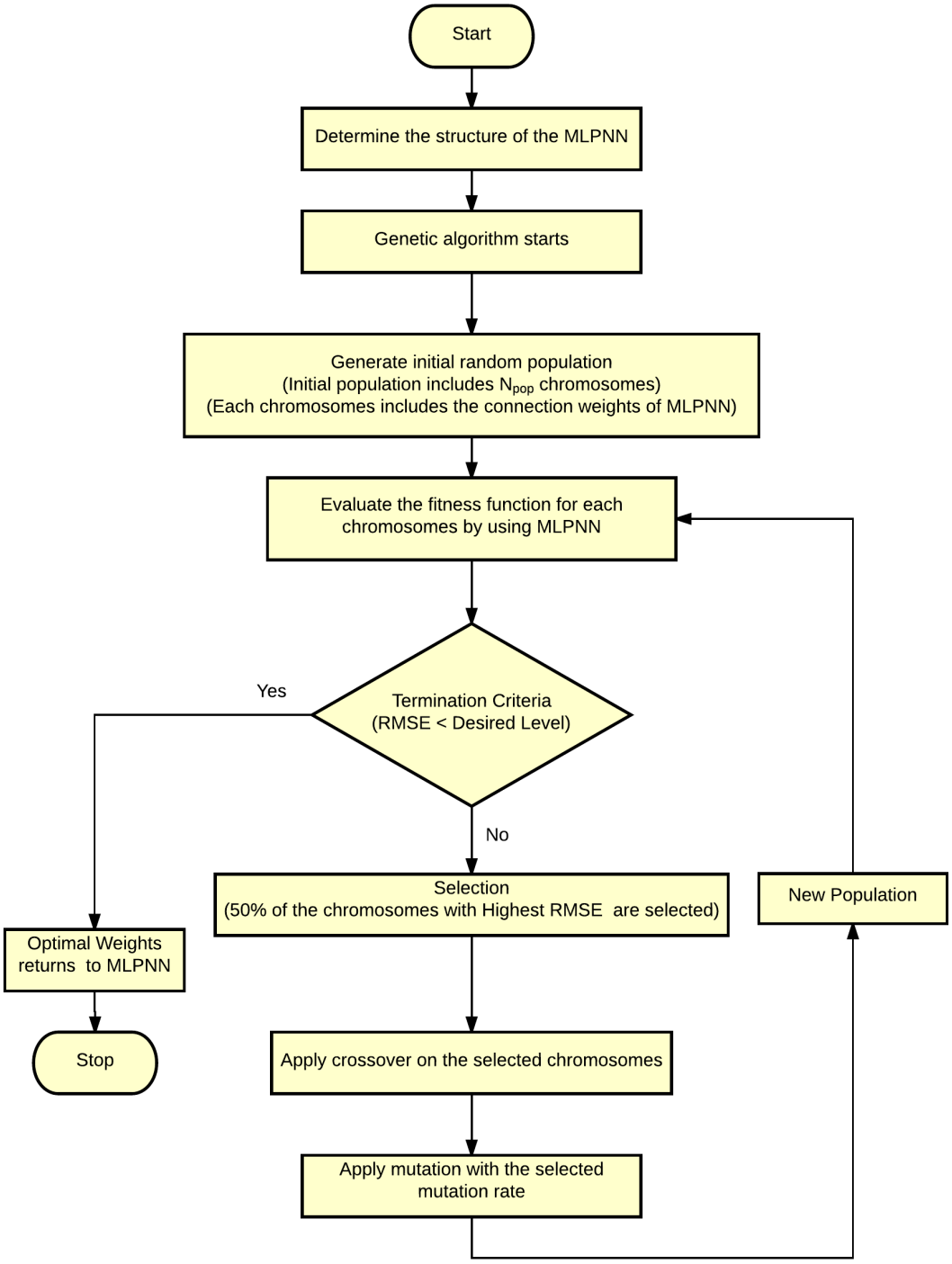


Figure 4.3: Flow chart of training MLPNN with GA.

Table 4.3: MLPNN specifications in agent-level fault isolation scheme.

Structure of the Network	2-7-1-3
$F(.)$ First Hidden Layer	Hyperbolic Tangent Sigmoid
$F(.)$ Second Hidden Layer	Hyperbolic Tangent Sigmoid
$F(.)$ Output Layer	Linear

population includes the weights of the MLPNN. The total number of parameters for the MLPNN in agent-level fault isolation scheme which are optimized by GA is 24 and are expressed in Table 4.4.

Table 4.4: Parameters of MLPNN in agent-level fault isolation scheme that are optimized with GA.

Parameters	Number of Parameters
Weights of the first hidden layer	$W_1(N(1), K) = W_1(7, 2) = 14$
Weights of the second hidden layer	$W_1(N(2), N(1)) = W_1(1, 7) = 7$
Weights of the output layer	$W_3(N(3), N(2)) = W_3(3, 1) = 3$

K = Number of inputs
N(1) = Number of neurons in the first hidden layer 1
N(2) = Number of neurons in the second hidden layer 1
N(3) = Number of neurons in output layer

In addition, the genetic algorithm parameters which is applied in agent-level fault isolation scheme are expressed in Table 4.5.

Table 4.5: Genetic algorithm parameters in agent-level fault isolation scheme.

Population size	15
Termination criterion	$RMSE < 0.04$
Mutation rate	0.15
Selection type	Natural Selection

The performance of the MLPNN in training phase for one of the training sets is depicted in Figure 4.4.

The RMSE of the MLPNN in agent-level fault isolation scheme for 10 different training sets are expressed in Table 4.6.

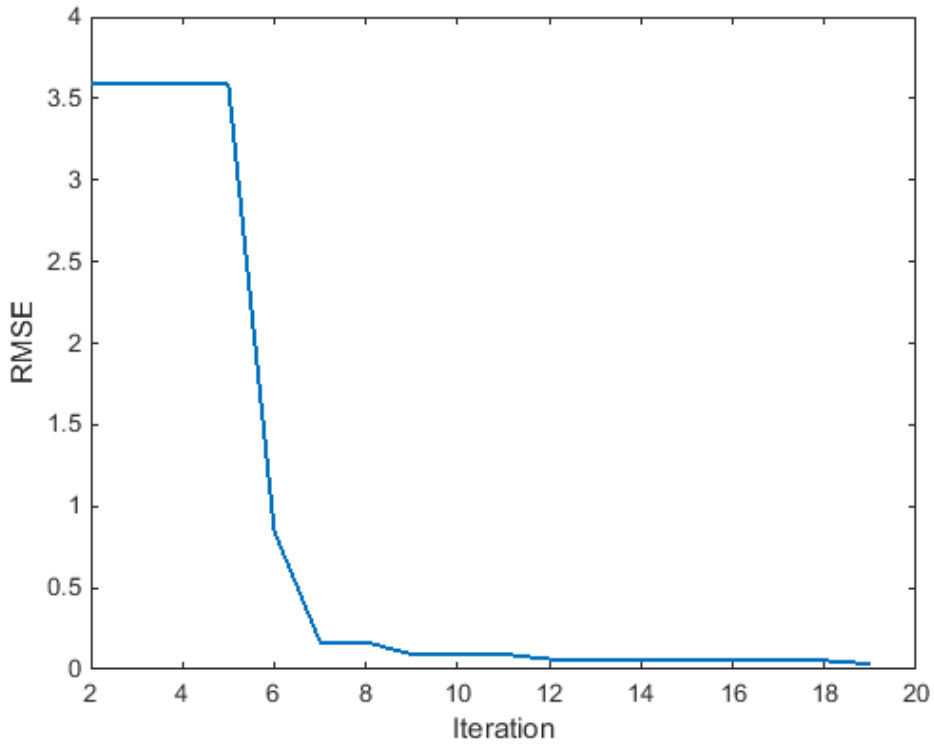


Figure 4.4: Performance of the MLPNN in training phase of agent-level fault isolation scheme.

Table 4.6: Training phase RMSE of the MLPNN in agent-level fault isolation scheme for 10 different training sets.

Training set	RMSE
1	0.039979
2	0.039366
3	0.038845
4	0.036954
5	0.039875
6	0.039432
7	0.039758
8	0.037412
9	0.038944
10	0.038236

According to Table 4.6, the average RMSE and its standard deviation are 0.03888 and 0.00104 respectively, which means the performance of the MLPNN in agent-level fault isolation scheme is quite acceptable.

4.1.4.2 Training Phase Results in Formation-Level

The MLPNN in formation-level fault isolation scheme is trained with 10 training sets which are obtained through different faulty operating condition of the AUV. Each of the training sets includes 1800 samples. The process of generating training sets is fully explained in section 4.1.3. As it is mentioned, in the first step the structure of the MLPNN has been selected. Structure of the MLPNN includes, the number of layers and the number of neurons in each layer. In this work, according to several experiments it has been observed that utilizing two hidden layer for the MLPNN in formation-level fault isolation scheme, provides an acceptable performance for the network. The optimum structure of the MLPNN its specifications for formation-level fault isolation scheme is shown in Table 4.7.

Table 4.7: MLPNN specifications in formation-level fault isolation scheme.

Structure of the Network	2-5-1-3
$F(.)$ First Hidden Layer	Hyperbolic Tangent Sigmoid
$F(.)$ Second Hidden Layer	Hyperbolic Tangent Sigmoid
$F(.)$ Output Layer	Linear

In Table 4.7 the four successive numbers in the structure of the network are the number of neurons in the input layer, first hidden layer, second hidden layer and output layer respectively.

As it is mentioned in training phase the variables of each chromosomes in the population includes the weights of the MLPNN. The total number of parameters for the MLPNN in formation-level fault isolation scheme which are optimized by GA is 18 and are expressed in Table 4.8.

Table 4.8: Parameters of MLP network in formation-level fault isolation scheme that are optimized with GA.

Parameters	Number of Parameters
Weights of the first hidden layer	$W_1(N(1), K) = W_1(5, 2) = 10$
Weights of the second hidden layer	$W_1(N(2), N(1)) = W_1(1, 5) = 5$
Weights of the output layer	$W_3(N(3), N(2)) = W_3(3, 1) = 3$

K = Number of inputs
N(1) = Number of neurons in the first hidden layer
N(2) = Number of neurons in the second hidden layer
N(3) = Number of neurons in output layer

In addition, the genetic algorithm parameters which is applied in formation-level fault isolation scheme are expressed in Table 4.9.

Table 4.9: Genetic algorithm parameters in formation-level fault isolation scheme.

Population size	15
Termination criterion	$RMSE < 0.04$
Mutation rate	0.15
Selection type	Natural Selection

The performance of the MLPNN in training phase for one of the training sets is depicted in Figure 4.5.

The RMSE of the MLPNN in formation-level fault isolation scheme for 10 different training sets are expressed in Table 4.10.

According to Table 4.10, the average RMSE and its standard deviation are 0.03717 and 0.00104 respectively, which means the performance of the MLPNN in formation-level fault isolation scheme is quite acceptable.

4.1.5 Cross-Validation Phase

The cross-validation method is from statistics which is used to ensure the generalization and to avoid overtraining. In this thesis the hold-out cross-validation method is used. In this method, the MLPNN is trained with training set and meanwhile

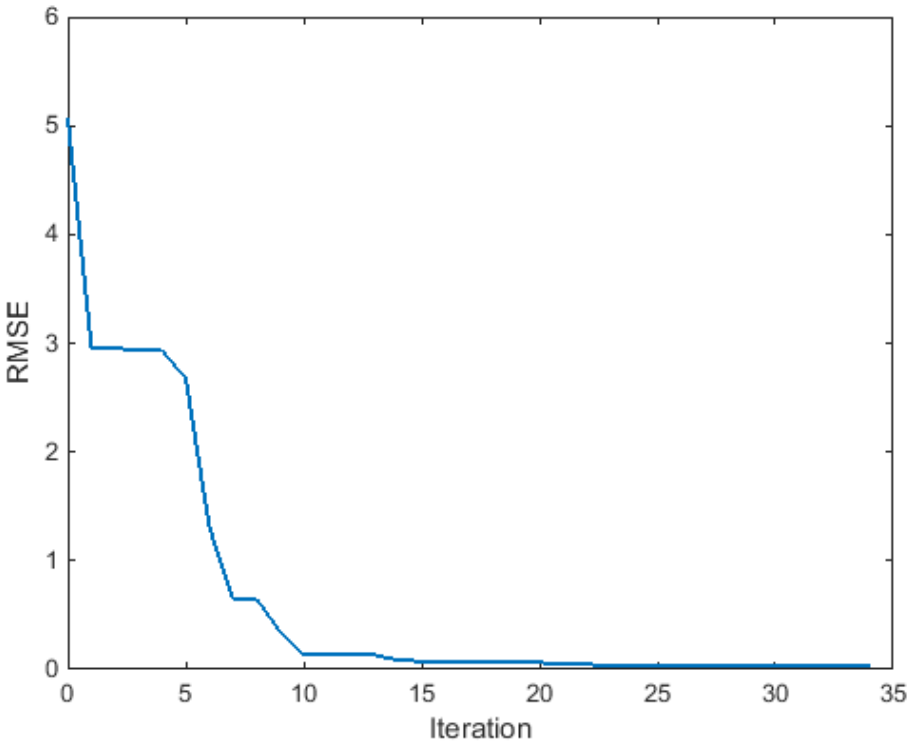


Figure 4.5: Performance of the MLPNN in training phase of formation-level fault isolation scheme.

Table 4.10: Training phase RMSE of the MLPNN in formation-level fault isolation scheme for 10 different training sets.

Training set	RMSE
1	0.038919
2	0.036423
3	0.038110
4	0.035439
5	0.037126
6	0.038432
7	0.036721
8	0.037119
9	0.036542
10	0.036922

the RMSE is checked on the validation set to avoid overtraining. This procedure is repeated and the training is stopped when the RMSE in cross-validation stops improving. If the training continues, it result in overtraining on the training set and it indicates that the MLPNN loses it generalization ability.

4.1.5.1 Cross-Validation Phase Results in Agent-Level

The RMSE of the MLPNN in agent-level fault isolation scheme for 10 different validation sets are expressed in Table 4.11

Table 4.11: Cross-validation phase RMSE of the MLPNN in agent-level fault isolation scheme for 10 different cross-validation sets.

Validation set	RMSE
1	0.039999
2	0.039741
3	0.038975
4	0.037931
5	0.039091
6	0.039532
7	0.039822
8	0.037920
9	0.038981
10	0.038440

According to Table 4.11, the average RMSE and its standard deviation are 0.03904 and 0.00075 respectively, which is quite acceptable.

4.1.5.2 Cross-Validation Results in Formation-Level

The RMSE of the MLPNN in formation-level fault isolation scheme for 10 different validation sets are expressed in Table 4.12

According to Table 4.12, the average RMSE and its standard deviation are 0.03780 and 0.001 respectively, which is quite acceptable.

Table 4.12: cross-validation phase RMSE of the MLPNN in formation-level fault isolation scheme for 10 different cross-validation sets.

Validation set	RMSE
1	0.039811
2	0.037113
3	0.038760
4	0.036837
5	0.038222
6	0.038672
7	0.036952
8	0.037437
9	0.037001
10	0.037213

4.1.6 Testing Phase

In order to present the capability of the MLPNNs in agent-level and formation-level fault isolation schemes, they are evaluated with 10 testing sets which are not formerly seen by the networks. As it is mentioned before, each of the testing sets contains 1200 samples.

4.1.6.1 Testing Phase Results in Agent-Level

The RMSE of the MLPNN in agent-level fault isolation scheme for 10 different testing sets are expressed in Table 4.13

According to Table 4.13, the average RMSE and its standard deviation are 0.03952 and 0.00083 respectively, which is quite acceptable.

4.1.6.2 Testing Phase Results in Formation-Level

The RMSE of the MLPNN in formation-level fault isolation scheme for 10 different testing sets are expressed in Table 4.14

According to Table 4.14, the average RMSE and its standard deviation are 0.03881 and 0.00077 respectively, which is quite acceptable.

Table 4.13: Testing phase RMSE of the MLPNN in agent-level fault isolation scheme for 10 different testing sets.

Testing set	RMSE
1	0.041002
2	0.040111
3	0.039522
4	0.038318
5	0.039621
6	0.039892
7	0.040221
8	0.038441
9	0.039112
10	0.038993

Table 4.14: Testing phase RMSE of the MLPNN in formation-level fault isolation scheme for 10 different testing sets.

Testing set	RMSE
1	0.040111
2	0.038541
3	0.039642
4	0.038531
5	0.039101
6	0.038992
7	0.037503
8	0.039198
9	0.037867
10	0.038655

4.1.7 Analysis of Fault Isolation Scheme

In order to investigate the performance of the proposed fault isolation algorithm, the confusion matrix approach for multi-class classification is used. A confusion matrix for classification of three classes (i.e. Thruster blocking, Flooded thruster and Loss of effectiveness in rotor) is is shown in Table 4.15.

Table 4.15: Table corresponding to confusion matrix [174].

		Predicted		
		C_1	C_2	C_3
Actual	C_1	$N_{1,1}$	$N_{1,2}$	$N_{1,3}$
	C_2	$N_{2,1}$	$N_{2,2}$	$N_{2,3}$
	C_3	$N_{3,1}$	$N_{3,2}$	$N_{3,3}$

In Table 4.15, C_1, C_2 and C_3 denote to thruster blocking, flooded thruster and loss of effectiveness in rotor respectively and the elements of the confusion matrix are defined as follows:

- $N_{1,1}$ is the true positive (TP) value for the thruster blocking class. This value indicates the number of samples that are correctly classified as thruster blocking fault.

- $N_{2,1}$ is the number of samples that are misclassified as thruster blocking fault while the type of the occurred fault is flooded thruster.

- $N_{3,1}$ is the number of samples that are misclassified as thruster blocking fault while the type of the occurred fault is loss of effectiveness in rotor.

- The sum of $N_{2,1}$ and $N_{3,1}$ values is the false negative (FN) value for the thruster blocking class and named as FN_1. This value indicates the number of samples that are incorrectly classified as thruster blocking fault.

- $N_{2,2}$ is the true positive (TP) value for the flooded thruster class. This value indicates the number of samples that are correctly classified as flooded thruster fault.

- $N_{1,2}$ is the number of samples that are misclassified as flooded thruster fault while the type of the occurred fault is thruster blocking.

- $N_{3,2}$ is the number of samples that are misclassified as flooded thruster fault while the type of the occurred fault is loss of effectiveness in rotor.

- The sum of $N_{1,2}$ and $N_{3,2}$ values is the false negative (FN) value for the flooded thruster class and named as FN_2. This value indicates the number of samples that are incorrectly classified as flooded thruster fault.

- $N_{3,3}$ is the true positive (TP) value for the loss of effectiveness in rotor class. This value indicates the number of samples that are correctly classified as loss of effectiveness in rotor fault.

- $N_{1,3}$ is the number of samples that are misclassified as loss of effectiveness in rotor fault while the type of the occurred fault is thruster blocking.

- $N_{2,3}$ is the number of samples that are misclassified as loss of effectiveness in rotor fault while the type of the occurred fault is flooded thruster.

- The sum of $N_{1,3}$ and $N_{2,3}$ values is the false negative (FN) value for the loss of effectiveness in rotor class and named as FN_3. This value indicates the number of samples that are incorrectly classified as loss of effectiveness in rotor fault.

According to the elements of the confusion matrix the accuracy and the error rate which is considered as the complement of the accuracy are calculated in following

equations:

$$Accuracy = \frac{N_{1,1} + N_{2,2} + N_{3,3}}{\sum_{i=1}^{3} \sum_{j=1}^{3} N_{i,j}} \tag{4.1.7}$$

$$Error\ rate = (1 - Accuracy) \tag{4.1.8}$$

The accuracy presents the overall correctness of the proposed method.

In order to measure the accuracy for each of the classes, the precision is defined for each of the aforementioned classes as follows:

$$Precision_{C_1} = \frac{N_{1,1}}{N_{1,1} + FN_1} \tag{4.1.9}$$

$$Precision_{C_2} = \frac{N_{2,2}}{N_{2,2} + FN_2} \tag{4.1.10}$$

$$Precision_{C_3} = \frac{N_{3,3}}{N_{3,3} + FN_3} \tag{4.1.11}$$

In this thesis the confusion matrix is obtained for a total of 270 faulty samples, including 90 samples corresponding to thruster blocking fault scenarios, 90 samples corresponding to flooded thruster fault scenarios and 90 samples corresponding to loss of effectiveness in rotor fault scenarios.

4.1.7.1 Confusion Matrix Analysis for Agent-Level Fault Isolation scheme

The confusion matrix for the agent-level fault isolation scheme indicated in Table 4.16 and the accuracy, error rate and precision for each classes are mentioned in Table 4.17.

According to the measurements in Table 4.17, it can be seen that the performance of the proposed fault isolation is quite well.

Table 4.16: Table corresponding to confusion matrix for agent-level fault isolation scheme.

		Predicted		
		C_1	C_2	C_3
Actual	C_1	65	0	25
	C_2	0	90	0
	C_3	27	0	63

Table 4.17: Accuracy, error rate and precision of each class for agent-level fault isolation scheme.

Class	Precision	Accuracy	Error rate
Thruster blocking C_1	70.65%	80.74%	19.26%
Flooded Thruster C_2	100%		
Loss of effectiveness in rotor C_3	71.59%		

4.1.7.2 Confusion Matrix Analysis for Formation-Level Fault Isolation Scheme

The confusion matrix for the formation-level fault isolation scheme indicated in Table 4.18 and the accuracy, error rate and precision for each classes are mentioned in Table 4.19.

Table 4.18: Table corresponding to confusion matrix for formation-level fault isolation scheme.

		Predicted		
		C_1	C_2	C_3
Actual	C_1	70	0	20
	C_2	0	90	0
	C_3	22	0	68

According to the measurements in Table 4.19, it can be seen that the performance of the proposed fault isolation is quite well.

As the Tables 4.17 and 4.19 indicate, the formation-level fault isolation scheme has a higher accuracy and precision, in other words the performance of the formation-level

Table 4.19: Accuracy, error rate and precision of each class for formation-level fault isolation scheme.

Class	Precision	Accuracy	Error rate
Thruster blocking C_1	76.08%	84.44%	15.56%
Flooded Thruster C_2	100%		
Loss of effectiveness in rotor C_3	77.27%		

fault isolation scheme is better comparing to agent-level fault isolation scheme.

4.2 Fault Identification Schemes

In the proposed fault identification scheme, a multi-layer perceptron neural network (MLPNN) is employed in each of the AUVs to categorize the severity of the occurred thruster fault into low, medium and high. In this section the proposed fault identification schemes in agent-level and formation-level for determining the severity of the occurred thruster fault of the AUV are presented.

The fault percentage intervals for different severity of faults corresponding to low, medium and high severity levels are determined based on the results from the fault detection scheme. These intervals are expressed in Table 4.20.

Table 4.20: Fault percentage intervals for different type of faults corresponding to low, medium and high severity levels.

Fault Type	Fault Percentage Interval
Thruster blocking low severity	Thruster torque is dropped by 1% to 7%
Thruster blocking medium severity	Thruster torque is dropped by 7% to 15%
Thruster blocking high severity	Thruster torque is dropped by 16% and larger
Flooded thruster low severity	Rotation velocity is increased by 1% to 7%
Flooded thruster medium severity	Rotation velocity is increased by 7% to 15%
Flooded thruster high severity	Rotation velocity is increased by 16% and larger
Loss of effectiveness in rotor low severity	Rotation velocity is dropped by 1% to 7%
Loss of effectiveness in rotor medium severity	Rotation velocity is dropped by 7% to 15%
Loss of effectiveness in rotor high severity	Rotation velocity is dropped by 16% and larger

4.2.1 Agent-Level Fault Identification Scheme

In proposed agent-level fault identification scheme, the residual signals which are generated in the agent-level fault detection scheme are processed and applied as an input to the MLPNN. In this work processing of the residual signal is calculating the ratio of the magnitudes of the residual signal before and after the occurrence of the fault which is applied as the input for the MLPNN. According to several experiments, it has been observed that the aforementioned input is valuable to fulfill the fault identification. The output of the proposed MLPNN is the fault label corresponding to the severity of the occurred fault in the thruster of the AUV. The structure of the proposed agent-level fault identification scheme is depicted in Figure 4.6.

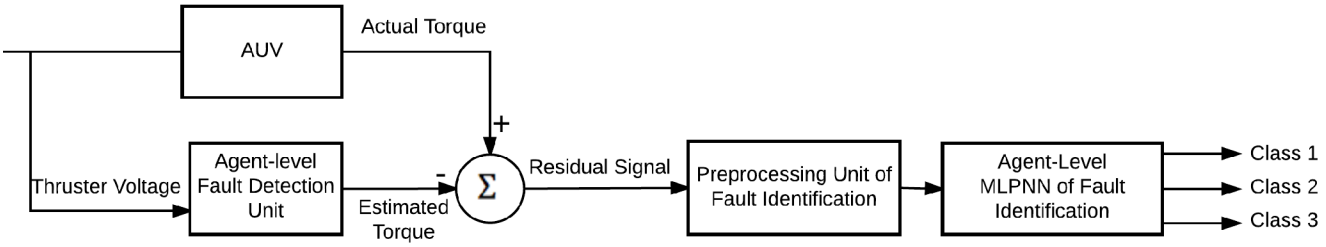


Figure 4.6: Structure of the agent-level fault identification scheme.

As it is shown in Figure 4.6, in the first step the preprocessing unit provides the input for the MLPNN and then the MLPNN indicate the class of the occurred fault. The assigned classes for fault severities in agent-level fault identification scheme is expressed in Table 4.21.

Table 4.21: Assigned classes for fault severities in agent-level fault identification scheme.

Fault Class (Severity)	Assigned Label
Class 1 (Low severity)	0 0 1
Class 2 (Medium severity)	0 1 0
Class 3 (High severity)	1 0 0

4.2.2 Formation-Level Fault Identification Scheme

In proposed formation-level fault identification scheme, the residual signals which are generated in the formation-level fault detection scheme are processed and applied as an input to the MLPNN. Since it is assumed that the two adjacent neighbors of the faulty AUV are identical and working in healthy mode, there is no significant difference between the two generated residual signals, therefore only one of them is used for fault identification purpose. In this work processing of the residual signal is calculating the ratio of the magnitudes of the residual signal before and after the occurrence of the fault which is applied as the input for the MLPNN. According to several experiments, it has been observed that the aforementioned input is valuable to fulfill the fault identification. The output of the proposed MLPNN is the fault label corresponding to the severity of the occurred fault in the thruster of the AUV. The structure of the proposed formation-level fault identification is depicted in Figure 4.7.

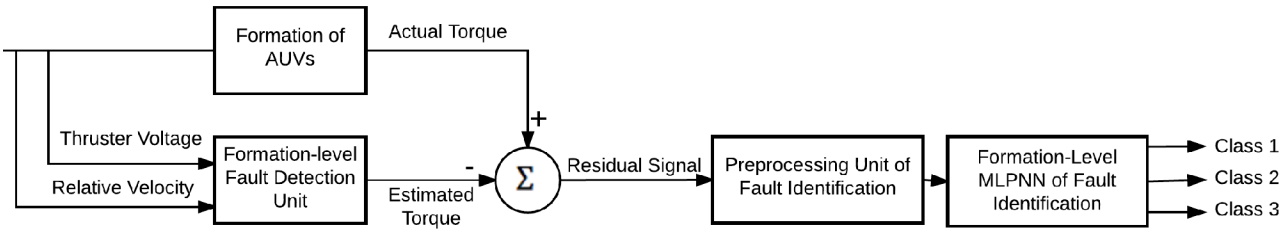


Figure 4.7: Structure of the formation-level fault identification scheme.

As it is shown in Figure 4.7, in the first step the preprocessing unit provides the input for the MLPNN and then the MLPNN indicate the class of the occurred fault. The assigned classes for fault severities in formation-level fault identification scheme is expressed in Table 4.22.

Table 4.22: Assigned classes for fault severities in formation-level fault identification scheme.

Fault Class (Severity)	Assigned Label
Class 1 (Low severity)	0 0 1
Class 2 (Medium severity)	0 1 0
Class 3 (High severity)	1 0 0

4.2.3 Data Preprocessing

In agent-level and formation-level fault identification schemes, providing the data for training the MLPNNs is divided into different steps. Firstly, the residual signals which are generated through the agent-level and formation-level fault detection schemes under various faulty operating condition of the AUV are collected. Secondly, these residuals are processed and the magnitudes of them before and after the fault occurrence is calculated and the ratio of them is considered as the input of the MLPNNs. Finally, these magnitudes which are the inputs for the MLPNN are normalized in the range of [0,1] before the training process begins. In order to have a reliable performance, 10 different input data sets are provided. Each data sets includes 2700 samples including 900 thruster blocking fault scenarios, 900 flooded thruster fault scenarios and 900 loss of effectiveness in rotor fault scenarios. The input data set is randomly divided into three parts namely as, training set, validation set and testing set. In this work, the 50% of the input data set is randomly chosen as training set, the 25% of it is selected at random as validation set and the remaining 25% of it is the testing set. It has been observed that by using 2700 samples as input data set an acceptable result for fault identification task can be accomplished.

4.2.4 Training Phase

The MLPNNs in both agent-level and formation-level fault identification schemes are trained with genetic algorithm (GA). The GA is applied to adjust the connection

weights of the MLPNNs. The training procedure of the MLPNNs in the fault identification schemes is same as the fault isolation scheme which is fully explained in section 4.1.4.

4.2.4.1 Training Phase Results in Agent-Level

The MLPNN in agent-level fault identification scheme is trained with 10 training sets which are obtained through different faulty operating condition of the AUV. Each of the training sets includes 900 samples. The process of generating training sets is fully explained in section 4.2.3. As it is mentioned, in the first step the structure of the MLPNN has been selected. Structure of the MLPNN includes, the number of layers and the number of neurons in each layer. In order to determine the structure of the MLPNN, we start with a small network structure, then the number of neurons and hidden layers are increased till the optimum structure of the network is achieved with respect to the MLPNN performance. In this work, according to several experiments it has been observed that utilizing one hidden layer for the MLPNN in agent-level fault identification scheme, provides an acceptable performance for the network. The optimum structure of the MLPNN and its specifications for agent-level fault identification scheme is shown in Table 4.23.

Table 4.23: MLPNN specifications in agent-level fault identification scheme.

Structure of the Network	1-6-3
$F(.)$ Hidden Layer	Hyperbolic Tangent Sigmoid
$F(.)$ Output Layer	Linear

In Table 4.23 the three successive numbers in the structure of the network are the number of neurons in the input layer, hidden layer and output layer respectively.

As it is mentioned in training phase the variables of each chromosomes in the population includes the weights of the MLPNN. The total number of parameters for

the MLPNN in agent-level fault identification scheme which are optimized by GA is 9 and are expressed in Table 4.24.

Table 4.24: Parameters of MLPNN in agent-level fault identification scheme that are optimized with GA.

Parameters	Number of Parameters
Weights of the hidden layer 1	$W_1(N(1), K) = W_1(6, 1) = 6$
Weights of the output layer	$W_3(N(3), N(2)) = W_3(3, 1) = 3$
K = Number of inputs N(1) = Number of neurons in the hidden layer 1 N(3) = Number of neurons in output layer	

In addition, the genetic algorithm parameters which is applied in agent-level fault identification scheme are expressed in Table 4.25.

Table 4.25: Genetic algorithm parameters in agent-level fault identification scheme.

Population size	20
Termination criterion	$RMSE < 0.04$
Mutation rate	0.25
Selection type	Natural Selection

The performance of the MLPNN in training phase for one of the training sets is depicted in Figure 4.8.

The RMSE of the MLPNN in agent-level fault identification scheme for 10 different training sets are expressed in Table 4.26.

According to Table 4.26, the average RMSE and its standard deviation are 0.03856 and 0.00095 respectively, which means the performance of the MLPNN in agent-level fault identification scheme is quite acceptable.

4.2.4.2 Training Phase Results in Formation-Level

The MLPNN in formation-level fault identification scheme is trained with 10 training sets which are obtained through different faulty operating condition of the AUV. Each

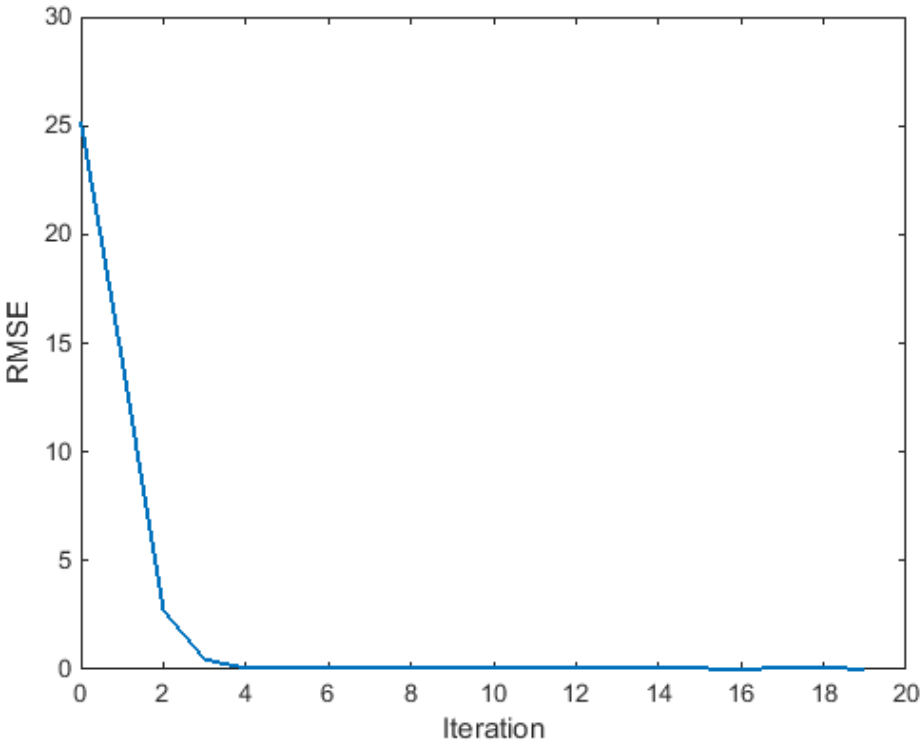


Figure 4.8: Performance of the MLPNN in training phase of agent-level fault identification scheme.

Table 4.26: Training phase RMSE of the MLPNN in agent-level fault identification scheme for 10 different training sets.

Training set	RMSE
1	0.038771
2	0.039101
3	0.036705
4	0.038342
5	0.039743
6	0.038521
7	0.037654
8	0.039967
9	0.038201
10	0.038640

of the training sets includes 900 samples. The process of generating training sets is fully explained in section 4.2.3. As it is mentioned, in the first step the structure of the MLPNN has been selected. Structure of the MLPNN includes, the number of layers and the number of neurons in each layer. In this work, according to several experiments it has been observed that utilizing one hidden layer for the MLPNN in formation-level fault identification scheme, provides an acceptable performance for the network. The optimum structure of the MLPNN and its specifications for formation-level fault identification scheme is shown in Table 4.27. In Table 4.27 the

Table 4.27: MLPNN specifications in formation-level fault identification scheme.

Structure of the Network	1-4-3
$F(.)$ Hidden Layer	Hyperbolic Tangent Sigmoid
$F(.)$ Output Layer	Linear

three successive numbers in the structure of the network are the number of neurons in the input layer, hidden layer and output layer respectively.

As it is mentioned in training phase the variables of each chromosomes in the population includes the weights of the MLPNN. The total number of parameters for the MLPNN in formation-level fault identification scheme which are optimized by GA is 7 and are expressed in Table 4.28.

Table 4.28: Parameters of MLPNN in formation-level fault identification scheme that are optimized with GA.

Parameters	Number of Parameters
Weights of the hidden layer 1	$W_1(N(1), K) = W_1(4, 1) = 4$
Weights of the output layer	$W_3(N(3), N(2)) = W_3(3, 1) = 3$

K = Number of inputs
N(1) = Number of neurons in the hidden layer 1
N(3) = Number of neurons in output layer

In addition, the genetic algorithm parameters which is applied in formation-level fault identification scheme are expressed in Table 4.29.

Table 4.29: Genetic algorithm parameters in formation-level fault identification scheme.

Population size	20
Termination criterion	$RMSE < 0.04$
Mutation rate	0.25
Selection type	Natural Selection

The performance of the MLPNN in training phase for one of the training sets is depicted in Figure 4.9.

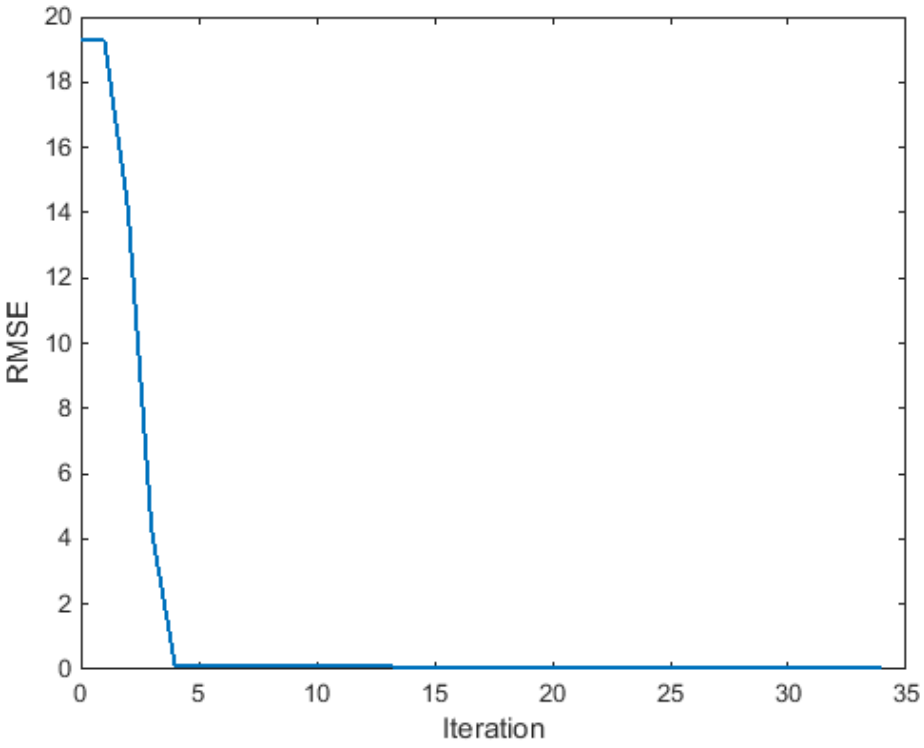


Figure 4.9: Performance of the MLPNN in training phase of formation-level fault identification scheme.

The RMSE of the MLPNN in formation-level fault identification scheme for 10 different training sets are expressed in Table 4.30.

According to Table 4.30 the average RMSE and its standard deviation are 0.03810 and 0.00096 respectively, which means the performance of the MLPNN in formation-level fault identification scheme is quite acceptable.

Table 4.30: Training phase RMSE of the MLPNN in formation-level fault identification scheme for 10 different training sets.

Training set	RMSE
1	0.038231
2	0.038795
3	0.036004
4	0.037987
5	0.038965
6	0.038108
7	0.037321
8	0.039409
9	0.037679
10	0.038519

4.2.5 Cross-Validation Phase

The process of cross-validation phase for the MLPNNs in both agent-level and formation-level fault identification schemes are same as the fault isolation scheme which if fully explained in section 4.1.5.

4.2.5.1 Cross-Validation Phase Results in Agent-Level

The RMSE of the MLPNN in agent-level fault identification scheme for 10 different validation sets are expressed in Table 4.31.

Table 4.31: Cross-validation phase RMSE of the MLPNN in agent-level fault identification scheme for 10 different validation sets.

Validation set	RMSE
1	0.039322
2	0.039765
3	0.037312
4	0.038899
5	0.039934
6	0.039012
7	0.037989
8	0.039990
9	0.038456
10	0.039103

According to Table 4.31, the average RMSE and its standard deviation are 0.03897 and 0.00086 respectively, which is quite acceptable.

4.2.5.2 Cross-Validation Phase Results in Formation-Level

The RMSE of the MLPNN in formation-level fault identification scheme for 10 different validation sets are expressed in Table 4.32.

Table 4.32: Cross-validation phase RMSE of the MLPNN in formation-level fault identification scheme for 10 different validation sets.

Validation set	RMSE
1	0.038446
2	0.038905
3	0.037211
4	0.038031
5	0.039021
6	0.038865
7	0.037981
8	0.039807
9	0.038006
10	0.038834

According to Table 4.32 the average RMSE and its standard deviation are 0.03852 and 0.00072 respectively, which is quite acceptable.

4.2.6 Testing Phase

In order to present the capability of the MLPNNs in agent-level and formation-level fault identification schemes, they are evaluated with 10 testing sets which are not formerly seen by the networks. As it is mentioned before, each of the testing sets contains 900 samples.

4.2.6.1 Testing Phase Results in Agent-Level

The RMSE of the MLPNN in agent-level fault identification scheme for 10 different testing sets are expressed in Table 4.33

Table 4.33: Testing phase RMSE of the MLPNN in agent-level fault identification scheme for 10 different testing sets.

Testing set	RMSE
1	0.042102
2	0.039989
3	0.038401
4	0.039219
5	0.040005
6	0.039631
7	0.038043
8	0.039995
9	0.038567
10	0.039789

According to Table 4.33 the average RMSE and its standard deviation are 0.03957 and 0.0011 respectively. Thus, it can be concluded that the performance of the proposed MLPNN is quite acceptable.

4.2.6.2 Testing Phase Results in Formation-Level

The RMSE of the MLPNN in formation-level fault identification scheme for 10 different testing sets are expressed in Table 4.34

According to Table 4.34 the average RMSE and its standard deviation are 0.0388 and 0.00064 respectively. Thus, it can be concluded that the performance of the proposed MLPNN is quite acceptable.

Table 4.34: Testing phase RMSE of the MLPNN in formation-level fault identification scheme for 10 different testing sets.

Testing set	RMSE
1	0.038897
2	0.039102
3	0.037865
4	0.038432
5	0.039512
6	0.039309
7	0.038308
8	0.039934
9	0.038398
10	0.038987

4.2.7 Analysis of Fault Identification Scheme

In order to investigate the performance of the proposed fault identification algorithm, the confusion matrix approach for multi-class classification is used. A confusion matrix for classification of three classes (i.e. Low severity, Medium severity and High severity) is shown in Table 4.35.

Table 4.35: Table corresponding to confusion matrix [174].

		Predicted		
		C_1	C_2	C_3
Actual	C_1	$N_{1,1}$	$N_{1,2}$	$N_{1,3}$
	C_2	$N_{2,1}$	$N_{2,2}$	$N_{2,3}$
	C_3	$N_{3,1}$	$N_{3,2}$	$N_{3,3}$

In Table 4.35, C_1, C_2 and C_3 denote to low severity, medium severity and high severity respectively and the elements of the confusion matrix are defined as follows:

- $N_{1,1}$ is the true positive (TP) value for the low severity class. This value indicates the number of samples that are correctly classified as low severity fault.
- $N_{2,1}$ is the number of samples that are misclassified as low severity fault while the severity of the occurred fault is medium.

- $N_{3,1}$ is the number of samples that are misclassified as low severity fault while the severity of the occurred fault is high.

- The sum of $N_{2,1}$ and $N_{3,1}$ values is the false negative (FN) value for the low severity class and named as FN_1. This value indicates the number of samples that are incorrectly classified as low severity fault.

- $N_{2,2}$ is the true positive (TP) value for the medium severity class. This value indicates the number of samples that are correctly classified as medium severity fault.

- $N_{1,2}$ is the number of samples that are misclassified as medium severity fault while the severity of the occurred fault is low.

- $N_{3,2}$ is the number of samples that are misclassified as medium severity fault while the severity of the occurred fault is high.

- The sum of $N_{1,2}$ and $N_{3,2}$ values is the false negative (FN) value for the medium severity class and named as FN_2. This value indicates the number of samples that are incorrectly classified as medium severity fault.

- $N_{3,3}$ is the true positive (TP) value for the high severity class. This value indicates the number of samples that are correctly classified as high severity fault.

- $N_{1,3}$ is the number of samples that are misclassified as high severity fault while the severity of the occurred fault is low.

- $N_{2,3}$ is the number of samples that are misclassified as high severity fault while the severity of the occurred fault is medium.

- The sum of $N_{1,3}$ and $N_{2,3}$ values is the false negative (FN) value for high severity class and named as FN_3. This value indicates the number of samples that are incorrectly classified as high severity fault.

The accuracy, error rate and precision for each of the classes in fault identification scheme are calculated based on equations (4.1.7) to (4.1.11) respectively.

In this work the confusion matrix is obtained for 270 faulty samples, including 90 samples corresponding to thruster blocking fault scenarios, 90 samples corresponding to flooded thruster fault scenarios and 90 samples corresponding to loss of effectiveness in rotor fault scenarios.

4.2.7.1 Confusion Matrix Analysis for Agent-Level Fault Identification Scheme

The confusion matrix for the agent-level fault identification scheme is indicated in Table 4.36 and the accuracy, error rate and precision for each classes are mentioned in Table 4.37.

Table 4.36: Table corresponding to confusion matrix for agent-level fault identification scheme.

		Predicted		
		C_1	C_2	C_3
Actual	C_1	53	37	0
	C_2	29	51	10
	C_3	0	27	63

According to the accuracy measurement in Table 4.37 it can be concluded that the over all performance of the proposed agent-level fault identification scheme is not quite acceptable.

Table 4.37: Accuracy, error rate and precision of each class for agent-level fault identification scheme.

Class	Precision	Accuracy	Error rate
Low severity C_1	58.88%	61.85%	38.15%
Medium severity C_2	56.66%		
High severity C_3	70%		

4.2.7.2 Confusion Matrix Analysis for Formation-Level Fault Identification Scheme

The confusion matrix for the formation-level fault identification scheme is indicated in Table 4.38 and the accuracy, error rate and precision for each classes are mentioned in Table 4.39.

Table 4.38: Table corresponding to confusion matrix for formation-level fault identification scheme.

		Predicted		
		C_1	C_2	C_3
Actual	C_1	74	16	0
	C_2	8	82	0
	C_3	0	14	76

Table 4.39: Accuracy, error rate and precision of each class for formation-level fault identification scheme.

Class	Precision	Accuracy	Error rate
Low severity C_1	82.22%	85.92%	14.08%
Medium severity C_2	91.11%		
High severity C_3	84.44%		

According to the measurements in Table 4.39 it can be concluded that the performance of the proposed formation-level fault identification scheme is quit well.

As the Tables 4.37 and 4.39 indicate, the formation-level fault identification scheme has a higher accuracy and precision comparing to agent-level fault identification, in other words the performance of the formation-level fault identification scheme is better in comparison with the agent-level fault identification scheme.

4.3 Conclusion

In this chapter, the agent-level and formation-level fault isolation and identification schemes based on neural networks are proposed. In agent-level fault isolation scheme and formation-level fault isolation scheme, the residual signals generated in the agent-level fault detection scheme are processed and their magnitudes before and after the occurrence of the fault are considered as the inputs to the MLPNN. The proposed MLPNNs in agent-level and formation-level fault isolation schemes are trained based on genetic algorithm in order to classify the type of the fault into thruster blocking, flooded thruster and loss of effectiveness in rotor. In order to investigate the performance of the proposed fault isolation schemes the confusion matrix analysis is applied. According to the confusion matrix elements the accuracy and precision of the proposed methods are calculated. The results indicate that the agent-level fault isolation scheme has the 80.74% accuracy while the formation-level fault isolation scheme has the 84.44% accuracy. Thus, it can be concluded the formation-level performance is better comparing to agent-level.

In agent-level fault identification scheme and formation-level fault identification scheme, the residual signals generated in the agent-level fault detection scheme are processed and the ratio of their magnitudes before and after the occurrence of the fault is considered as the input to the MLPNN. The proposed MLPNNs in agent-level and formation-level fault identification schemes are trained based on genetic algorithm in order to classify the severity of the fault into low, medium and high severities. In

order to investigate the performance of the proposed fault identification schemes the confusion matrix analysis is applied. According to the confusion matrix elements the accuracy and precision of the proposed methods are calculated. The results indicate that the agent-level fault identification scheme has the 61.85% accuracy while the formation-level fault identification scheme has the 85.92% accuracy. Thus, it can be concluded the formation-level fault identification scheme has a better and more reliable performance comparing to agent-level fault identification scheme.

Chapter 5

Conclusions and Future Work

5.1 Thesis Summary

In this thesis the problem of fault detection, isolation and identification (FDII) scheme for formation of AUVs is investigated. The proposed intelligent fault diagnosis scheme for formation of AUVs is based on dynamic neural networks (DNNs) training with genetic algorithm.

For detecting the faults, two levels of fault detection are proposed named as: agent-level fault detection (ALFD) and formation-level fault detection (FLFD). In the ALFD scheme the absolute measurements are used to train the DNN while in the FLFD both absolute measurements and relative measurements of the AUV with respects to its adjacent neighbors are utilized for training of the network.

The performance of the ALFD and FLFD scheme are evaluated and compared for 3 faulty scenarios (i.e. Thruster blocking, Flooded thruster, Loss of effectiveness in rotor) with different severities through confusion matrix analysis. The confusion matrix analysis indicates that the FLFD scheme is capable of detecting the low, medium and high severity faults with high accuracy and precision while the ALFD scheme detects only the medium and high severity faults and is not able to detect

low severity faults.

In addition, the performances of the dynamic network during the training phase based on two different methods named as genetic algorithm and extended dynamic back-propagation are compared. The results indicated that the GA approach converge faster, in other words it provides better results in less number of iterations.

In order to isolate the occurred fault, two different approaches namely as, agent-level and formation-level fault isolation schemes are developed. In both agent-level fault isolation scheme a MLPNN is employed which is trained based on GA. The residual signals are generated in the agent-level and formation-level fault detection scheme are processed and applied as an input to the MLPNNs respectively. In this work processing of the residual signal is calculating the magnitudes of the residual signal before and after the occurrence of the fault which are applied as the two inputs for the MLPNNs. The output of the proposed MLPNNs is the label corresponding to the type of the fault. The confusion matrix analysis indicate that the performance of the proposed formation-level fault isolation scheme in determining the type of the occurred fault is better in comparison with the agent-level fault isolation scheme.

In order to identify the severity of the occurred fault, two different fault identification approaches are applied namely as, agent-level and formation-level fault identification schemes. In both agent-level fault identification scheme a MLPNN is employed which is trained based on GA. The residual signals are generated in the agent-level and formation-level fault detection scheme are processed and applied as an input to the MLPNNs respectively. In this work processing of the residual signal is calculating the magnitudes of the residual signal before and after the occurrence of the fault and the ratio of the magnitudes is applied as an input for the MLPNNs. The output of the proposed MLPNNs is the label corresponding to the severity of the fault. The

confusion matrix analysis indicate that the performance of the proposed formation-level fault identification scheme is better comparing to agent-level fault identification scheme and it has higher accuracy and precision in determining the severity of the occurred fault.

5.2 Suggestions for Future Work

- Firstly, in this work the genetic algorithm is used to train the dynamic neural network. Further research studies can be done on determining the optimal architecture for the DNN by utilizing the genetic algorithm.

- Secondly, development and testing a hybrid method which utilize a combination of genetic algorithm (GA) and extended dynamic back-propagation(EDBP) can be consider as a future study.

- Thirdly, in this thesis the problem of fault detection, isolation and identification for the thruster of the AUV has been addressed. Developing a fault diagnosis system for the sensors of the AUV can be investigated in future studies.

- Fourthly, another suggestion for future work can be focused on development of the more advanced FDII system for the faulty situation that contains more than one type of fault or combination of faults occur.

Bibliography

[1] Fossen, T. I. (1994). Guidance and control of ocean vehicles (Vol. 199, No. 4). New York: Wiley.

[2] Schoenwald, D. (2000). AUVs: In space, air, water, and on the ground. IEEE Control Systems Magazine, 20(6), 15-18.

[3] Stilwell, D. J., & Bishop, B. E. (2000). Platoons of underwater vehicles. Control Systems, IEEE, 20(6), 45-52.

[4] Rajala, A. G., Edwards, D. B., & O'Rourke, M. (2005, January). Collaborative behavior for vehicle replacement in AUV formations. In ASME 2005 International Mechanical Engineering Congress and Exposition (pp. 141-149). American Society of Mechanical Engineers.

[5] Stilwell, D. J., Gadre, A. S., Sylvester, C. A., & Cannell, C. J. (2004, June). Design elements of a small low-cost autonomous underwater vehicle for field experiments in multi-vehicle coordination. In Autonomous Underwater Vehicles, 2004 IEEE/OES (pp. 1-6). IEEE.

[6] Fiorelli, E., Bhatta, P., Leonard, N. E., & Shulman, I. (2003, August). Adaptive sampling using feedback control of an autonomous

underwater glider fleet. In Proc. 13th Int. Symp. on Unmanned Untethered Submersible Technology (UUST).

[7] Chen, X., Serrani, A., & Ozbay, H. (2003, December). Control of leader-follower formations of terrestrial UAVs. In Decision and Control, 2003. Proceedings. 42nd IEEE Conference on (Vol. 1, pp. 498-503). IEEE.

[8] Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. Robotics and Automation, IEEE Transactions on, 14(6), 926-939.

[9] Ren, W., & Beard, R. (2004). Decentralized scheme for spacecraft formation flying via the virtual structure approach. Journal of Guidance, Control, and Dynamics, 27(1), 73-82.

[10] Leonard, N. E., & Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In Decision and Control, 2001. Proceedings of the 40th IEEE Conference on (Vol. 3, pp. 2968-2973). IEEE.

[11] Tanner, H. G., Jadbabaie, A., & Pappas, G. J. (2003, December). Stable flocking of mobile agents, Part I: Fixed topology. In Decision and Control, 2003. Proceedings. 42nd IEEE Conference on (Vol. 2, pp. 2010-2015). IEEE.

[12] Olfati-Saber, R., & Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. Automatic Control, IEEE Transactions on, 49(9), 1520-1533.

[13] Jadbabaie, A., Lin, J., & Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. Automatic Control, IEEE Transactions on, 48(6), 988-1001.

[14] Lawton, J. R., Beard, R. W., & Young, B. J. (2003). A decentralized approach to formation maneuvers. Robotics and Automation, IEEE Transactions on, 19(6), 933-941.

[15] Ayoubi, M. (1994). Fault diagnosis with dynamic neural structure and application to a turbo-charger. In International Symposium on Fault Detection, Supervision and Safety for Technical Proceesses (SAFEPROCESS94), Espoo, Finland, June (pp. 13-16).

[16] Ayoubi, M., Schäffer, M., & Sinsel, S. (1995). Dynamic neural units for nonlinear dynamic systems identification. In From Natural to Artificial Neural Computation (pp. 1045-1051). Springer Berlin Heidelberg.

[17] Mohammadi, R., Naderi, E., Khorasani, K., & Hashtrudi-Zad, S. (2010, October). Fault diagnosis of gas turbine engines by using dynamic neural networks. In ASME Turbo Expo 2010: Power for Land, Sea, and Air (pp. 365-376). American Society of Mechanical Engineers.

[18] Valdes, A., Khorasani, K., & Ma, L. (2009). Dynamic neural network-based fault detection and isolation for thrusters in formation flying of satellites. In Advances in Neural Networks-ISNN 2009 (pp. 780-793). Springer Berlin Heidelberg.

[19] Al-Zyoud, I. D., & Khorasani, K. (2006, July). Neural Network-based Actuator Fault Diagnosis for Attitude Control Subsystem of a Satellite. In Automation Congress, 2006. WAC'06. World (pp. 1-6). IEEE.

[20] Al-Zyoud, I. D., & Khorasani, K. (2005). Detection of actuator faults using a dynamic neural network for the attitude control subsystem of a satellite. In Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on (Vol. 3, pp. 1746-1751). IEEE.

[21] Al-Zyoud, I. A. D., & Khorasani, K. (2006, July). Neural Network-based Actuator Fault Diagnosis for Attitude Control Subsystem of an Unmanned Space Vehicle. In IJCNN (pp. 3686-3693).

[22] Patan, K., & Parisini, T. (2005). Identification of neural dynamic models for fault detection and isolation: the case of a real sugar evaporation process. Journal of Process Control, 15(1), 67-79.

[23] Yazdizadeh, A., & Khorasani, K. (2002). Adaptive time delay neural network structures for nonlinear system identification. Neurocomputing, 47(1), 207-240.

[24] Yazdizadeh, A., & Khorasani, K. (1997, June). Identification of a class of nonlinear systems using dynamic neural network structures. In Neural Networks, 1997., International Conference on (Vol. 1, pp. 194-198). IEEE.

[25] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks.

Acoustics, Speech and Signal Processing, IEEE Transactions on, 37(3), 328-339.

[26] Talebi, H. A., Khorasani, K., & Tafazoli, S. (2009). A recurrent neural-network-based sensor and actuator fault detection and isolation for nonlinear systems with application to the satellite's attitude control subsystem. Neural Networks, IEEE Transactions on, 20(1), 45-60.

[27] Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. Neural Networks, IEEE Transactions on, 1(1), 4-27.

[28] Li, Z. Q., Ma, L., & Khorasani, K. (2006, July). A dynamic neural network-based reaction wheel fault diagnosis for satellites. In Neural Networks, 2006. IJCNN'06. International Joint Conference on (pp. 3714-3721). IEEE.

[29] Li, L., Ma, L., & Khorasani, K. (2005). A dynamic recurrent neural network fault diagnosis and isolation architecture for satellites Actuator/thruster failures. In Advances in Neural Networks-ISNN 2005 (pp. 574-583). Springer Berlin Heidelberg.

[30] Korbicz, J., Patan, K., & Obuchowicz, A. (1999). Dynamic neural networks for process modelling in fault detection and isolation systems. International Journal of Applied Mathematics and Computer Science, 9, 519-546.

[31] Yazdizadeh, A., Khorasani, K., & Patel, R. V. (2000). Identification of a two-link flexible manipulator using adaptive time delay

neural networks. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 30(1), 165-172.

[32] Satoh, S., Yakuwa, F., & Dote, Y. (2003, October). Combination of radial basis function (RBF) and time delayed neural networks (TDNN) for fault diagnosis of automobile transmission gears using general parameter learning and adaptation. In Systems, Man and Cybernetics, 2003. IEEE International Conference on (Vol. 2, pp. 1457-1462). IEEE.

[33] Barai, S. V., & Pandey, P. C. (1997). Time-delay neural networks in damage detection of railway bridges. Advances in Engineering Software, 28(1), 1-10.

[34] Wöhler, C., & Anlauf, J. K. (1999). An adaptable time-delay neural-network algorithm for image sequence analysis. IEEE Transactions on Neural Networks, 10(6), 1531-1536.

[35] Edwards, D. B., Bean, T. A., Odell, D. L., & Anderson, M. J. (2004). A leader-follower algorithm for multiple AUV formations (pp. 40-46). IEEE.

[36] Cui, R., Ge, S. S., Voon Ee How, B., & Choo, Y. S. (2009, May). Leader-follower formation control of underactuated auvs with leader position measurement. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on (pp. 979-984). IEEE.

[37] Wang, Y., Yan, W., & Yan, W. (2009, August). A leader-follower formation control strategy for AUVs based on line-of-sight guidance. In Mechatronics and Automation, 2009. ICMA 2009. International Conference on (pp. 4863-4867). IEEE.

[38] Emrani, S., Dirafzoon, A., Talebi, H. A., Nikravesh, S. Y., & Menhaj, M. B. (2010, November). An adaptive leader-follower formation controller for multiple AUVs in spatial motions. In IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society (pp. 59-64). IEEE.

[39] Emrani, S., Dirafzoon, A., & Talebi, H. A. (2011, September). Adaptive distributed formation control of multiple autonomous underwater vehicles. In Control Applications (CCA), 2011 IEEE International Conference on (pp. 693-698). IEEE.

[40] Zhong-Hai, Z., Jian, Y., Wen-Xia, Z., & Jin-Ping, Z. (2012, May). Virtual-leader-follower structure and finite-time controller based cooperative control of multiple autonomous underwater vehicles. In Control and Decision Conference (CCDC), 2012 24th Chinese (pp. 3670-3675). IEEE.

[41] Cui, R., Xu, D., & Yan, W. (2007, May). Formation control of autonomous underwater Vehicles under fixed topology. In Control and Automation, 2007. ICCA 2007. IEEE International Conference on (pp. 2913-2918). IEEE.

[42] Montana, D. J. (1995). Neural network weight selection using genetic algorithms. Intelligent Hybrid Systems, 8(6), 12-19.

[43] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling.

[44] Montana, D. J., & Davis, L. (1989, August). Training Feedforward Neural Networks Using Genetic Algorithms. In IJCAI (Vol. 89, pp. 762-767).

[45] Whitley, D., & Kauth, J. (1988). GENITOR: A different genetic algorithm. Colorado State University, Department of Computer Science.

[46] Schaffer, J. D., Whitley, D., & Eshelman, L. J. (1992, June). Combinations of genetic algorithms and neural networks: A survey of the state of the art. In Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on (pp. 1-37). IEEE.

[47] Whitley, D., & Hanson, T. Optimizing neural networks using faster, more accurate genetic search. 3rd Intern. In Conference on Genetic Algorithms, Washington DC (pp. 391-396).

[48] Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. Parallel computing, 14(3), 347-361.

[49] Goldberg, D. E. (1990). E.(1989). Genetic algorithms in search, optimization and machine learning. Reading: Addison-Wesley.

[50] Yang, B., Su, X. H., & Wang, Y. D. (2002). BP neural network optimization based on an improved genetic algorithm. In Machine

Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on (Vol. 1, pp. 64-68). IEEE.

[51] Smith, J., & Fogarty, T. C. (1996, May). Self adaptation of mutation rates in a steady state genetic algorithm. In Evolutionary Computation, 1996., Proceedings of IEEE International Conference on (pp. 318-323). IEEE.

[52] Mordaunt, P., & Zalzala, A. M. S. (2002). Towards an evolutionary neural network for gait analysis. In Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on (Vol. 2, pp. 1922-1927). IEEE.

[53] Seiffert, U. (2001, April). Multiple Layer Perceptron training using genetic algorithms. In ESANN (pp. 159-164).

[54] Pai, G. V. (2004, December). A fast converging evolutionary neural network for the prediction of uplift capacity of Suction Caissons. In Cybernetics and Intelligent Systems, 2004 IEEE Conference on (Vol. 1, pp. 654-659). IEEE.

[55] Fu, Z., Mo, J., Chen, L., & Chen, W. (2010). Using genetic algorithm-back propagation neural network prediction and finite-element model simulation to optimize the process of multiple-step incremental air-bending forming of sheet metal. Materials & design, 31(1), 267-277.

[56] Suresh, S., Saraswathi, S., & Sundararajan, N. (2010). Performance enhancement of extreme learning machine for multi-category sparse

data classification problems. Engineering Applications of Artificial Intelligence, 23(7), 1149-1157.

[57] Ho, W. H., & Chang, C. S. (2011). Genetic-algorithm-based artificial neural network modeling for platelet transfusion requirements on acute myeloblastic leukemia patients. Expert Systems with Applications, 38(5), 6319-6323.

[58] Su, C. L., Yang, S. M., & Huang, W. L. (2011). A two-stage algorithm integrating genetic algorithm and modified Newton method for neural network training in engineering systems. Expert Systems with Applications, 38(10), 12189-12194.

[59] Karegowda, A. G., Manjunath, A. S., & Jayaram, M. A. (2011). Application of genetic algorithm optimized neural network connection weights for medical diagnosis of pima Indians diabetes. International Journal on Soft Computing, 2(2), 15-23.

[60] Ding, S., Xu, L., Su, C., & Jin, F. (2012). An optimizing method of RBF neural network based on genetic algorithm. Neural Computing and Applications, 21(2), 333-336.

[61] Ding, S., Su, C., & Yu, J. (2011). An optimizing BP neural network algorithm based on genetic algorithm. Artificial Intelligence Review, 36(2), 153-162.

[62] Elveren, E., & Yumuşak, N. (2011). Tuberculosis disease diagnosis using artificial neural network trained with genetic algorithm. Journal of medical systems, 35(3), 329-332.

[63] Che, Z. G., Chiang, T. A., & Che, Z. H. (2011). Feed-forward neural networks training: A comparison between genetic algorithm and back-propagation learning algorithm. Int. J. Innov. Comp. Inf. Control, 7(10), 5839-5851.

[64] Irani, R., & Nasimi, R. (2011). Evolving neural network using real coded genetic algorithm for permeability estimation of the reservoir. Expert Systems with Applications, 38(8), 9862-9866.

[65] Ali Ahmadi, M., Zendehboudi, S., Lohi, A., Elkamel, A., & Chatzis, I. (2013). Reservoir permeability prediction by neural networks combined with hybrid genetic algorithm and particle swarm optimization. Geophysical Prospecting, 61(3), 582-598.

[66] Belciug, S., & Gorunescu, F. (2013). A hybrid neural network/genetic algorithm applied to breast cancer detection and recurrence. Expert Systems, 30(3), 243-254.

[67] Qiu, F., & Li, Y. (2014, June). Air traffic flow of genetic algorithm to optimize wavelet neural network prediction. In Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on (pp. 1162-1165). IEEE.

[68] Kalderstam, J., Edén, P., Bendahl, P. O., Strand, C., Fernö, M., & Ohlsson, M. (2013). Training artificial neural networks directly on the concordance index for censored data using genetic algorithms. Artificial intelligence in medicine, 58(2), 125-132.

[69] Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part

I: Quantitative model-based methods. Computers & chemical engineering, 27(3), 293-311.

[70] Venkatasubramanian, V., Rengaswamy, R., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. Computers & Chemical Engineering, 27(3), 313-326.

[71] Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., & Yin, K. (2003). A review of process fault detection and diagnosis: Part III: Process history based methods. Computers & chemical engineering, 27(3), 327-346.

[72] Simani, S., Fantuzzi, C., & Patton, R. J. (2003). Model-Based Fault Diagnosis Techniques. In Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques (pp. 19-60). Springer London.

[73] Patton, R. J., Frank, P. M., & Clarke, R. N. (1989). Fault diagnosis in dynamic systems: theory and application. Prentice-Hall, Inc..

[74] Patton, R. J., Clark, R. N., & Frank, P. M. (Eds.). (2000). Issues of fault diagnosis for dynamic systems. Springer.

[75] Chen, J., & Patton, R. J. (2012). Robust model-based fault diagnosis for dynamic systems. Springer Publishing Company, Incorporated.

[76] Isermann, R. (1993). Fault diagnosis of machines via parameter estimation and knowledge processingtutorial paper. Automatica, 29(4), 815-835.

[77] Frank, P. M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results. Automatica, 26(3), 459-474.

[78] Patton, R. J., & Chen, J. (1994). Review of parity space approaches to fault diagnosis for aerospace systems. Journal of Guidance, Control, and Dynamics, 17(2), 278-285.

[79] Chen, J., & Zhang, H. Y. (1990). Parity vector approach for detecting failures in dynamic systems. International Journal of Systems Science, 21(4), 765-770.

[80] Gertler, J. (1997). Fault detection and isolation using parity relations. Control engineering practice, 5(5), 653-661.

[81] Gertler, J. (1998). Fault detection and diagnosis in engineering systems. CRC press.

[82] Magni, J. F., & Mouyon, P. (1994). On residual generation by observer and parity space approaches. Automatic Control, IEEE Transactions on, 39(2), 441-447.

[83] Massoumnia, M. A. (1986). A geometric approach to the synthesis of failure detection filters. Automatic Control, IEEE Transactions on, 31(9), 839-846.

[84] Douglas, R. K., & Speyer, J. L. (1996). Robust fault detection filter design. Journal of guidance, control, and dynamics, 19(1), 214-218.

[85] Douglas, R. K., & Speyer, J. L. (1999). H bounded fault detection filter. Journal of guidance, control, and dynamics, 22(1), 129-138.

[86] Chen, R. H., Mingori, D. L., & Speyer, J. L. (2003). Optimal stochastic fault detection filter. Automatica, 39(3), 377-390.

[87] Chen, R. H., & Speyer, J. L. (2000). A generalized least squares fault detection filter. International Journal of Adaptive Control and Signal Processing, 14(7), 747-757.

[88] Chen, R. H., & Speyer, J. L. (2002). Robust multiplefault detection filter. International Journal of Robust and Nonlinear Control, 12(8), 675-696.

[89] Park, J., & Rizzoni, G. (1994). A new interpretation of the fault detection filter part 1: Closed-form algorithm. International Journal of Control, 60(5), 767-787.

[90] Park, J., Rizzoni, G., & Ribbens, W. B. (1994). On the representation of sensor faults in fault detection filters. Automatica, 30(11), 1793-1795.

[91] Basseville, M., & Nikiforov, I. V. (1993). Detection of abrupt changes: theory and application (Vol. 104). Englewood Cliffs: Prentice Hall.

[92] Berec, L. (1998). A multimodel method to fault detection and diagnosis: Bayesian solution. An introductory treatise. International journal of adaptive control and signal processing, 12(1), 81-92.

[93] Tzafestas, S., & Watanabe, K. (1990). Modern approaches to system/sensor fault detection and diagnosis. Journal A, 31(4), 42-57.

[94] Frank, P. M., & Koppen-Seliger, B. (1997). Fuzzy logic and neural network applications to fault diagnosis. International Journal of Approximate Reasoning, 16(1), 67-88.

[95] Patton, R. J., Uppal, F. J., & Lopez-Toribio, C. J. (2000, June). Soft computing approaches to fault diagnosis for dynamic systems: a survey. In 4th IFAC Symposium on Fault Detection supervision and Safety for Technical Processes (pp. 198-211).

[96] Sorsa, T., & Koivo, H. N. (1993). Application of artificial neural networks in process fault diagnosis. Automatica, 29(4), 843-849.

[97] Ayoubi, M. (1994, October). Nonlinear dynamic systems identification with dynamic neural networks for fault diagnosis in technical processes. In Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on (Vol. 3, pp. 2120-2125). IEEE.

[98] Sobhani-Tehrani, E., & Khorasani, K. (2009). Fault diagnosis of nonlinear systems using a hybrid approach (Vol. 383). Heidelberg: Springer.

[99] Alessandri, A. (2003). Fault diagnosis for nonlinear systems using a bank of neural estimators. Computers in Industry, 52(3), 271-289.

[100] Zhang, X., Polycarpou, M. M., & Parisini, T. (2002). A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems. Automatic Control, IEEE Transactions on, 47(4), 576-593.

[101] Antonelli, G. (2014). Underwater robots. Springer.

[102] Fossen, T. I. (2002). Marine control systems: guidance, navigation, and control of ships, rigs and underwater vehicles (No. 28).

[103] Dearden, R., & Ernits, J. (2013). Automated fault diagnosis for an autonomous underwater vehicle. IEEE J. Oceanic Eng, 58, 11-21.

[104] Caccia, M., Bono, R., Bruzzone, G., Bruzzone, G., Spirandelli, E., & Veruggio, G. (2001). Experiences on actuator fault detection, diagnosis and accomodation for ROVs. International Symposiyum of Unmanned Untethered Sub-mersible Technol.

[105] Healey, A. J. (2005). Analytical redundancy and fuzzy inference in AUV fault detection and compensation. Naval Postgraduate School Monterey CA Center for Autonomous Underwater Vehicle Research.

[106] Yang, K. C., Yuh, J., & Choi, S. K. (1998, May). Experimental study of fault-tolerant system design for underwater robots. In Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on (Vol. 2, pp. 1051-1056). IEEE.

[107] Yang, K. C. H., Yuh, J., & Choi, S. K. (1999). Fault-tolerant system design of an autonomous underwater vehicle ODIN: an experimental study. International Journal of Systems Science, 30(9), 1011-1019.

[108] Podder, T. K., Antonelli, G., & Sarkar, N. (2000). Fault tolerant control of an autonomous underwater vehicle under thruster redundancy: simulations and experiments. In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference

on (Vol. 2, pp. 1251-1256). IEEE.

[109] Podder, T. K., Antonelli, G., & Sarkar, N. (2001). An experimental investigation into the fault-tolerant control of an autonomous underwater vehicle. Advanced Robotics, 15(5), 501-520.

[110] Hamilton, K., Lane, D. M., Taylor, N. K., & Brown, K. (2001, May). Fault diagnosis on autonomous robotic vehicles with recovery: an integrated heterogeneous-knowledge approach. In ICRA (pp. 3232-3237).

[111] Alessandri, A., Caccia, M., & Veruggio, G. (1999). Fault detection of actuator faults in unmanned underwater vehicles. Control Engineering Practice, 7(3), 357-368.

[112] Barnett, D., McClaran, S., Nelson, E., McDermott, M., & Williams, G. (1996, June). Architecture of the Texas A&M autonomous underwater vehicle controller. In Autonomous Underwater Vehicle Technology, 1996. AUV'96., Proceedings of the 1996 Symposium on (pp. 231-237). IEEE.

[113] Isermann, R., & Balle, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. Control engineering practice, 5(5), 709-719.

[114] Alessandri, A., Caccia, M., & Veruggio, G. (1998, September). A model-based approach to fault diagnosis in unmanned underwater vehicles. In OCEANS'98 Conference Proceedings (Vol. 2, pp. 825-829). IEEE.

[115] Alessandri, A., Hawkinson, T., Healey, A. J., & Veruggio, G. (1999). Robust model-based fault diagnosis for unmanned underwater vehicles using sliding mode-observers. Naval Postgraduate School Monterey CA Center for Autonomous Underwater Vehicle Research.

[116] Bono, R., Bruzzone, G., & Caccia, M. (1999). ROV actuator fault diagnosis through servo-amplifiers' monitoring: an operational experience. In OCEANS'99 MTS/IEEE. Riding the Crest into the 21st Century (Vol. 3, pp. 1318-1324). IEEE.

[117] Zheng, X. (1992, June). Layered control of a practical AUV. In Autonomous Underwater Vehicle Technology, 1992. AUV'92., Proceedings of the 1992 Symposium on (pp. 142-147). IEEE.

[118] Alekseev, Y. K., Kostenko, V. V., & Shumsky, A. Y. (1994, September). Use of identification and fault diagnostic methods for underwater robotics. In OCEANS'94.'Oceans Engineering for Today's Technology and Tomorrow's Preservation.'Proceedings (Vol. 2, pp. II-489). IEEE.

[119] Mangoubi, R. S., Appleby, B. D., Verghese, G. C., & Vander Velde, W. E. (1995, December). A robust failure detection and isolation algorithm. In Decision and Control, 1995., Proceedings of the 34th IEEE Conference on (Vol. 3, pp. 2377-2382). IEEE.

[120] Ranganathan, N., Patel, M. I., & Sathyamurthy, R. (2001). An intelligent system for failure detection and control in an autonomous underwater vehicle. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 31(6), 762-767.

[121] Beale, G. O., & Kim, J. (2000, August). A robust approach to reconfigurable control. In 5th IFAC Conference on Manoeuvring and Control of Marine Craft, Aalborg, DK (pp. 197-202).

[122] Kim, J. H., & Beale, G. O. (2001, June). Fault detection and classification in underwater vehicles using the T 2 statistic. In 9th Mediterranean Conference on Control and Automation, Dubrovnik, Kr.

[123] Ni, L., & Fuller, C. R. (2003). Control reconfiguration based on hierarchical fault detection and identification for unmanned underwater vehicles. Journal of Vibration and Control, 9(7), 735-748.

[124] Antonelli, G., Caccavale, F., Sansone, C., & Villani, L. (2004). Fault diagnosis for auvs using support vector machines. In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on (Vol. 5, pp. 4486-4491). IEEE.

[125] Omerdic, E., & Roberts, G. (2004). Thruster fault diagnosis and accommodation for open-frame underwater vehicles. Control Engineering Practice, 12(12), 1575-1598.

[126] Mišković, N., & Barišić, M. (2005). Fault detection and localization on underwater vehicle propulsion systems using principal component analysis. In IEEE ISIE 2005 Conference.

[127] Shahid, M. Z., Soucacos, P. P., & Beale, G. O. (2005, June). Fault recovery in underwater vehicles using fully automatic reconfigurable control. In Industrial Electronics, 2005. ISIE 2005.

Proceedings of the IEEE International Symposium on (Vol. 1, pp. 87-94). IEEE.

[128] Soucacos, P. P., & Beale, G. O. (2006, June). Design of a robust reconfigurable control algorithm for underwater vehicles experiencing stern plane jams. In American Control Conference, 2006 (pp. 7-pp). IEEE.

[129] Zhu, D., Liu, Q., & Yang, Y. (2008). An Active Fault-Tolerant Control Method Of unmanned Underwater Vehicles with Continuous and Uncertain Faults. International Journal of Advanced Robotic Systems, 5(4).

[130] Wang, Y., Zhang, M., & Guo, Y. (2009, August). Fault detection and data restoration based on PCA for sensors of autonomous underwater vehicle. In Mechatronics and Automation, 2009. ICMA 2009. International Conference on (pp. 4801-4805). IEEE.

[131] Hanai, A. M., Choi, S., Marani, G., & Rosa, K. H. (2009, May). Experimental validation of model-based thruster fault detection for underwater vehicles. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on (pp. 194-199). IEEE.

[132] Bian, X., Chen, T., Yan, Z., Zhao, D., & Yu, G. (2009, February). Fault diagnosis based on Grey Dynamic Prediction for AUV sensor. In Industrial Technology, 2009. ICIT 2009. IEEE International Conference on (pp. 1-6). IEEE.

[133] Liang, X., Li, W., Su, L., Yin, H., & Zhao, J. (2010, January). Thruster Fault Diagnosis of Autonomous Underwater Vehicles Based on Least Disturbance Wavelet Neural Network. In Computer Modeling and Simulation, 2010. ICCMS'10. Second International Conference on (Vol. 1, pp. 78-82). IEEE.

[134] Shumsky, A., Zhirabok, A., & Hajiyev, C. (2010, October). Observer based fault diagnosis in thrusters of autonomous underwater vehicle. In Control and Fault-Tolerant Systems (SysTol), 2010 Conference on (pp. 11-16). IEEE.

[135] Corradini, M. L., Monteriu, A., & Orlando, G. (2011). An actuator failure tolerant control scheme for an underwater remotely operated vehicle. Control Systems Technology, IEEE Transactions on, 19(5), 1036-1046.

[136] Corradini, M. L., Monteriu, A., Orlando, G., & Pettinari, S. (2011, December). An actuator failure tolerant robust control approach for an underwater remotely operated vehicle. In Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on (pp. 3934-3939). IEEE.

[137] Zhang, M., Wu, J., & Wang, Y. (2011, March). Simultaneous Faults Detection and Location of Thrusters and Sensors for Autonomous Underwater Vehicle. In Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on (Vol. 1, pp. 504-507). IEEE.

[138] Jianguo, W., Lei, W., Chunmeng, J., Yushan, S., Bin, H., & Jiqing, L. (2011, July). Wavelet neural network applied to fault diagnosis

of underwater vehicle. In Control Conference (CCC), 2011 30th Chinese (pp. 4301-4306). IEEE.

[139] Liu, Q., Zhu, D., & Yang, S. X. (2012). Unmanned Underwater Vehicles Fault Identification and Fault-Tolerant Control Method Based on FCA-CMAC Neural Networks, Applied on an Actuated Vehicle. Journal of Intelligent & Robotic Systems, 66(4), 463-475.

[140] Davoodi, M. R., Meskin, N., & Khorasani, K. (2013, November). Simultaneous fault detection and control design for an autonomous unmanned underwater vehicle. In GCC Conference and Exhibition (GCC), 2013 7th IEEE (pp. 529-534). IEEE.

[141] Jia, Q., Xu, J., & Wang, G. (2013, November). Fault diagnosis based on grey correlation analysis for autonomous underwater vehicle sensor. In Chinese Automation Congress (CAC), 2013 (pp. 656-659). IEEE.

[142] Jia, Q., Xu, J., & Wang, G. (2013, November). Fault diagnosis based on second-order Taylor series dynamic prediction for autonomous underwater vehicle sensor. In Chinese Automation Congress (CAC), 2013 (pp. 651-655). IEEE.

[143] Davoodi, M. R., Meskin, N., & Khorasani, K. (2014, June). Simultaneous fault detection, isolation and control tracking design using a single observer-based module. In American Control Conference (ACC), 2014 (pp. 3047-3052). IEEE.

[144] Zhang, M. J., Wu, J., & Chu, Z. Z. (2014). Multi-fault diagnosis for autonomous underwater vehicle based on fuzzy weighted support vector domain description. China Ocean Engineering, 28, 599-616.

[145] Tettamanzi, A., & Tomassini, M. (2001). Soft computing: integrating evolutionary, neural, and fuzzy systems (Vol. 86). Springer.

[146] Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 1, p. 740). New York: springer.

[147] Callan, R. (1998). Essence of neural networks. Prentice Hall PTR.

[148] Haykin, S., & Network, N. (2004). A comprehensive foundation. Neural Networks, 2(2004).

[149] Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.). (2000). Evolutionary computation 1: Basic algorithms and operators (Vol. 1). CRC Press.

[150] Fogel, D. B., & Michalewicz, Z. (2000). Evolutionary Computation 2: Advanced Algorithms and Operators.

[151] Eiben, A. E., & Smith, J. E. (2003). Introduction to evolutionary computing. springer.

[152] De Jong, K. A. (2006). Evolutionary computation: a unified approach. MIT press.

[153] Yao, X., & Xu, Y. (2006). Recent advances in evolutionary computation. Journal of Computer Science and Technology, 21(1), 1-18.

[154] Fogel, D. B. (1997, January). The Advantages of Evolutionary Computation. In BCEC (pp. 1-11).

[155] Fogel, D. B. (1998). Evolutionary computation: the fossil record. Wiley-IEEE Press.

[156] Fogel, D. B. (1993). Applying evolutionary programming to selected traveling salesman problems. Cybernetics and systems, 24(1), 27-36.

[157] Yao, X. (1999). Evolving artificial neural networks. Proceedings of the IEEE, 87(9), 1423-1447.

[158] Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

[159] Haupt, R. L., & Haupt, S. E. (2004). Practical genetic algorithms. John Wiley & Sons.

[160] Sivanandam, S. N., & Deepa, S. N. (2008). Genetic Algorithm Optimization Problems (pp. 165-209). Springer Berlin Heidelberg.

[161] LeSage, J. R. (2010). Electromechanical retrofit of submarine control surface actuators (Doctoral dissertation, University of Texas at Austin).

[162] Adams, J. M., & Rattan, K. S. (2001). Intelligent control of a direct-drive robot using multi-stage fuzzy logic. In Circuits and Systems, 2001. MWSCAS 2001. Proceedings of the 44th IEEE 2001 Midwest Symposium on (Vol. 2, pp. 543-546). IEEE.

[163] Höfling, T., & Isermann, R. (1996). Fault detection based on adaptive parity equations and single-parameter tracking. Control Engineering Practice, 4(10), 1361-1369.

[164] Ren, W., & Beard, R. W. (2008). Distributed consensus in multi-vehicle cooperative control. Springer-Verlag, London.

[165] Pivano, L. (2008). Thrust estimation and control of marine propellers in four-quadrant operations.

[166] Tayarani-Bathaie, S. S., Vanini, Z. S., & Khorasani, K. (2014). Dynamic neural network-based fault diagnosis of gas turbine engines. Neurocomputing, 125, 153-165.

[167] Roberts, G. N., & Sutton, R. (Eds.). (2006). Advances in unmanned marine vehicles (Vol. 69). Iet.

[168] Ourdighi, A., & Benyettou, A. (2010). An Adaptive Time delay Neural Network Training using Parallel Genetic Algorithms in Time series Prediction and Classication.

[169] Valeriano-Medina, Y., Martínez, A., Hernández, L., Sahli, H., Rodríguez, Y., & Cañizares, J. R. (2013). Dynamic model for an autonomous underwater vehicle based on experimental data. Mathematical and Computer Modelling of Dynamical Systems, 19(2), 175-200.

[170] Silpa-Anan, C. (2001). Autonomous underwater robot: Vision and control. Australian National University.

[171] Fossen, T. I., & Blanke, M. (2000). Nonlinear output feedback control of underwater vehicle propellers using feedback form estimated axial flow velocity. Oceanic Engineering, IEEE Journal of, 25(2), 241-255.

[172] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427-437.

[173] Aksman, L. M. (2006). Force estimation based compliance control of a two link harmonically driven robotic manipulator (Doctoral dissertation).

[174] Makhtar, M., Neagu, D. C., & Ridley, M. J. (2011). Comparing multi-class classifiers: on the similarity of confusion matrices for predictive toxicology applications. In Intelligent Data Engineering and Automated Learning-IDEAL 2011 (pp. 252-261). Springer Berlin Heidelberg.

[175] McEwen, R., & Streitlien, K. (2001). Modeling and control of a variable-length auv. Proc 12th UUST.

[176] Sisli, U. Y., & Temeltas, H. (2008, January). Formation Graphs and Decentralized Formation Control of Multi Vehicles with Kinematics Constraints. In European Robotics Symposium 2008 (pp. 93-101). Springer Berlin Heidelberg.

[177] Carelli, R., De la Cruz, C., & Roberti, F. (2006). Centralized formation control of non-holonomic mobile robots. Latin American applied research, 36(2), 63-69.

[178] Cao, Y. (2010). Decentralized coordination of multiple autonomous vehicles (Doctoral dissertation, Utah State University).

[179] Antonelli, G. (2014). Fault Detection/Tolerance Strategies for AUVs and ROVs. In Underwater Robots (pp. 101-116). Springer International Publishing.

[180] Rao, D., & Williams, S. B. (2009, December). Large-scale path planning for underwater gliders in ocean currents. In Australasian Conference on Robotics and Automation (ACRA), Sydney.

[181] Schultz, J. A. (2009). Autonomous Underwater Vehicle (AUV) Propulsion System Analysis and Optimization (Doctoral dissertation, Virginia Polytechnic Institute and State University).

[182] Jamali, N., Kormushev, P., Carrera, A., Carreras, M., & Caldwell, D. G. Underwater Robot-Object Contact Perception using Machine Learning on Force/Torque Sensor Feedback. In Proceedings of IEEE ICRA (Vol. 15).