Neural Network-based Actuator Fault Detection and Isolation for the Attitude Control Subsystem of a Satellite

Iz Al-Dein Al-Zyoud

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 in
Electrical and Computer Engineering at
Concordia University
Montréal, Québec, Canada

September 2005

Your file Votre référence
ISBN: 978-0-494-16414-3
Our file Notre référence
ISBN: 978-0-494-16414-3

To my lovely parents and my dear brothers and sisters for their care, love and patience

Acknowledgment

First of all, I'm so grateful to Professor K. Khorsani for his guidance, encouragement and patience through my research work under his supervision. He had shown professionalism in his supervision and teaching attitude.

My special thanks for the examining committee members for their remarks and recommendations to my thesis work.

I would like to express my warm feelings for every body who have supported and helped me during my studentship. Also I would like to thank all academic and administrative staff of my respected university and particularly the department of electrical and computer engineering.

Finally, it's my pleasure to name some of my best friends who have supported me physically and emotionally every moment through various stages of my life: Falah, Murad, Amjad, Yousef, Eslam, Othman, Ashraf, Cliff, Hussein, Hussam, Amer, Hesham, Sa'ed, Ehsan, Ma'moun, Khaled, Ra'ed, Abduallah, Mohammad, Mas'oud and Mansour.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fault diagnosis and identification have been widely researched during the recent years due to the increasing demand on reliable operation of safety critical control systems, such as autonomous vehicles. The main tasks of fault diagnosis are to detect and isolate occurring faults in order to avoid overall failure of the monitored system and any catastrophes involving human fatalities and material damage [1].

The basic idea in fault detection system is to generate signals that reflect inconsistencies between the nominal and faulty operating conditions. Such signals known as residuals are usually calculated using a number of analytical methods [2].

The monitoring of faults in feedback control system components (sensors, actuators, etc) has to be known as fault detection and isolation (FDI). This can be achieved with two main approaches, namely model-based and process history-based using either qualitative or quantitative modeling [1], [2], [3]. The schematic diagram in Fig. 1.1, presents a detailed classification for the

diagnostic methods that are extensively researched during the last decades. Model-based approaches use prior mathematical information about the system to model the normal process. Based on some fundamental understanding of the physics of the monitored process, the model can be developed. This understanding is expressed in terms of mathematical functional relations between the inputs and the outputs of the system in quantitative models (differential equations, state space methods, transfer functions, etc) [3], [4], [5].

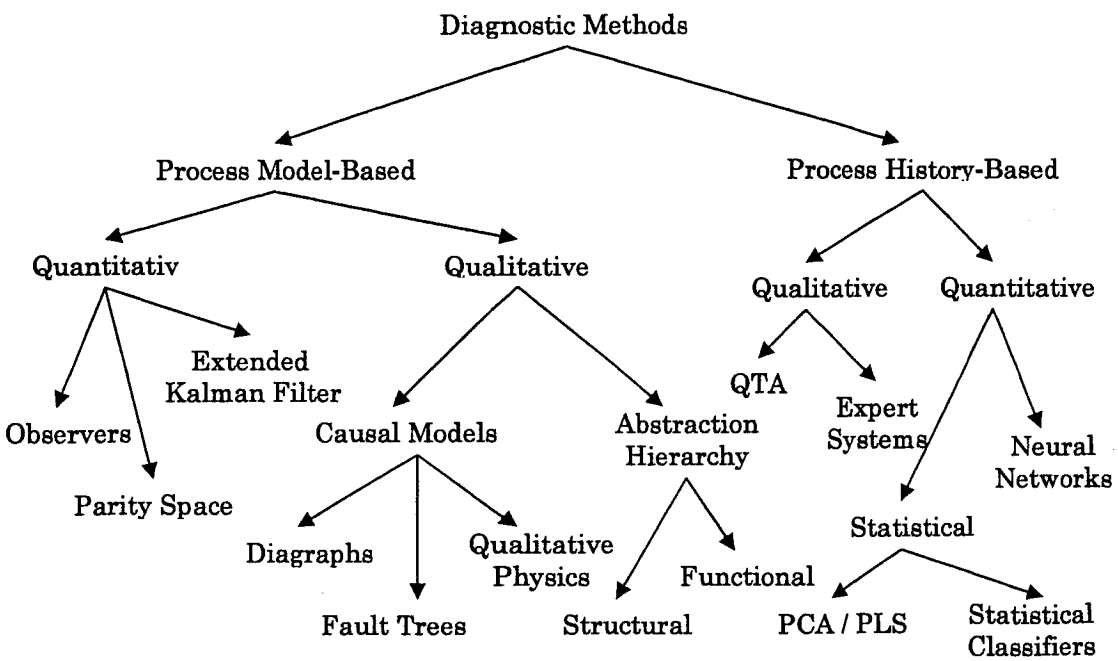


Fig. 1.1 Classification of diagnostic algorithms [3]

Quantitative model-based methods are based on parameter estimation, state estimation or parity space concepts; the philosophy behind these approaches are that a fault will cause changes to certain physical parameters and

measurements, which in turn will lead to a change in certain model parameters or states. The faults can be detected and isolated by monitoring the estimated parameters or states. A prior knowledge is assumed about the relationships between the faults and the model parameters (which parameters are likely to change and how they changed) or states (how many state variables and their possible physical significance) [5].

Patton et al. [5] summarized some special properties of the quantitative model-based FDI methods, based on their experience with real processes and simulations as follows:

Parameter Estimation

- Model structure must be known
- Suitable especially for multiplicative faults† and additive faults†† on the input and output signal.
- Uniquely detectable for various parameter changes
- Very small changes are detectable, which includes the detection of slowly developing as well as fast developing faults
- Possible deep fault diagnosis (physical coefficients)
- Possible on-line real-time application, if not very fast processes

State Estimation

- The model structure including parameters must be known rather

† These are changes (abrupt or gradual) in some plant parameters. They cause changes in the plant outputs which depend also on the magnitude of the known inputs [6].

†† These are unknown inputs acting on the plant, which are normally zero and which, when present, cause a change in the plant outputs independent of the known inputs [6].

accurately

- Suitable especially for additive faults, mostly multi-output signals required
- Very fast reaction after sudden faults
- Only some parameter changes detectable (depends on selection of state variables and lumping with other parameters)
- Possible on-line real-time application for fast processes, if not too many observers required
- No input signal changes required for additive faults (but then some parameter changes, e.g., time constants, not detectable)
- Mostly only relatively large faults detectable

Parity Equations

- Model structure and parameters must be known and must be fit the process well
- Suitable especially for additive faults
- Very fast reaction after sudden faults
- Possible on-line real-time application for fast processes
- No input signal changes required for additive faults (but then some parameter changes not detectable)
- Some faults can be small to be detected (e.g., additive faults and gains),and some must be large (e.g., time constants)

In contrast to the quantitative models, in qualitative models-based the

relationships among system variables and parameters are used to describe the system behavior in qualitative terms such as causalities and IF-THEN rules. The qualitative model can be developed either as qualitative causal models or abstraction hierarchies. The knowledge in the casual models can be represented qualitatively in various forms, such as diagraphs, fault trees or qualitative physics, more detailed regarding these methods can be found in [7].

Another diagnostic approach is based on the process history of the monitored system. In this approach, it is assumed that a large amount of historical process data is available. This data can be transformed and presented as a prior knowledge to a diagnostic system through different methods which are known as feature extraction. The extraction process can be either quantitative or qualitative in nature. Qualitative history information can be extracted mainly by expert systems and qualitative trend analysis (QTA). Trend analysis and prediction represent important parts of process monitoring and supervisory control. QTA often provides valuable information that facilitates reasoning about process behavior. Most malfunctions cases leave a distinct trend in the monitored actuator or sensor. These distinct trends can be utilized suitably in identifying the underlying abnormality in the process; so that, an early faults detection can be accomplished by a suitable classification and analysis of process trends which leads to a quick control [8].

Fault diagnosis using rule-based expert systems needs an extensive database of rules and the accuracy of diagnosis depends on these rules. Expert system tasks are time consuming due to the necessity for a huge amount and detailed database of rules and many process experts. Also, it suffers from the uniqueness of knowledge and the necessity for rules updating when large industrial plants are considered [4].

Fuzzy logic is now being investigated as powerful modelling and decision making tool for nonlinear FDI systems. Fuzzy logic methods for fault diagnosis problem belong to the sub-class of rule-based expert system; it can express expert knowledge in terms of natural language statements. It has the potential to formulate the qualitative relationships among the model variables of the process being monitored using IF-THEN rules. Fuzzy sets perform a smooth interface between the qualitative variables involved in the rules and the numerical data at the inputs and outputs of the model. The appealing feature of fuzzy logic is that its ability to deal with imprecise facts or noisy data and is therefore suited for applications where complete information about fault and system is not available to the FDI designer [1], [5], [4], [9].

Decision making stage in FDI system, is a logic decision process that transform the residual signals into qualitative statements. Therefore, the problem of decision making can be treated in a novel way with the aid of fuzzy logic [9]. The principle of residual evaluation using fuzzy logic can be

fulfilled in three steps .Firstly, the residuals have to be fuzzified, then they have to be evaluated by an inference mechanism using fuzzy IF-THEN rules, and finally they have to be defuzzified. The introduction of fuzzy logic in decision making stage can improve the reliability of FDI methods for real industrial processes [10], [11], [13].

Statistical and non-statistical classifiers represent the main methods used to extract quantitative history information, main statistical feature extraction methods are principle component analysis (PCA), partial least squares (PLS) and statistical pattern classifier. Quantitative feature extraction approaches essentially formulate the diagnostic problem-solving as a pattern recognition problem. The objective of pattern recognition is the classification of data vectors into predetermined classes, but the classification task in statistical methods can be done using knowledge of a prior class distribution [8]. For more details, authors in [3], [7], and [8] have reviewed fault diagnosis problem in three parts.

Of particular interest to us are neural networks which are an important class of non-statistical classifiers. Recently, neural networks have been researched extensively in literature, and they have been employed successfully in pattern recognition as well as system identification [12], [14], [15], [18] and proposed as powerful technique in fault diagnosis problem-solving [19], [22],

The appeal of neural networks and its application to fault diagnosis that has been studied in [16], [17], [23], are due to their capabilities to cope with

nonlinearities, complexities, uncertainties, and noisy and corrupted data. Neural network constitute suitable modeling tool for representing highly nonlinear processes.

Generally, it is more advantageous to develop a nonlinear neural network based model for a range of operating conditions rather than to develop a bank of linear models, each developed for a particular operating point, therefore rendering neural networks as ideal tools for generating residuals [3].

Alessandri [20] has proposed a fault diagnosis technique for nonlinear systems based on using a bank of neural estimators and applied it to diagnose actuator and sensor faults in a small unmanned underwater vehicle. The diagnosis algorithm was accomplished by means of neural network estimators, which provides estimates of parameters that describe actuator, plant, and sensors fault.

Karpenko et al. [21] has presented two methodologies to solve the FDI problem using feedforward neural networks. Firstly; the network assigns each vector of input data to a specific class of operating condition (either normal or faulty). The size of the output layer grows as the number of possible fault classes increases. But the neural network in the second methodology estimate the magnitude of each faulty condition, and only one neuron is required for each failure mode in the output layer of the trained network. As a result the size of the network can be reduced significantly.

Various developed FDI approaches show different properties with regard the

diagnosis of different faults in a process. The capabilities of FDI system can be enhanced by taking the advantages of combining these methods (quantitative and qualitative methods). This combination can also minimize the disadvantages of the two approaches; particularly, it is important to reduce or eliminate the ambiguity in qualitative reasoning. Hence, main future research directions are toward combining these methods together to provide highly reliable diagnostic algorithms. For example, fuzzy logic can be used together with state space models or neural networks [1].

A recent research activity has focused on the integration of neural networks and fuzzy logic through a neuro-fuzzy system for nonlinear FDI systems. The goal is to integrate the neural network learning ability with the explicit knowledge of fuzzy logic. The underlying concept is to structure a neural network, which can model the nonlinear systems efficiently, in a fuzzy logic format. Therefore, the network can be trained more rapidly and will provide explicit description about the causes of the faults. There are several possible approaches reported in the literature of combining fuzzy logic and neural network [9], [13].

Another form of integration can be obtained using a model-based method with fuzzy logic to formulate the so-called fuzzy observers. The main idea is to use the Takagi-Sugeno (T-S) fuzzy model which describes the nonlinear dynamic system by a number of locally-linearized models. According to the T-S model, a non-linear dynamic system can be linearized around a number of

operating points. Each linear model represents local system behaviour around the operating point and is described by fuzzy IF-THEN rules [9].

1.1 Desirable Characteristics of a Fault Diagnostic System

Fault diagnosis area has a variety of approaches than can fulfill the diagnostic requirements. So that it is useful to identify a set of desirable characteristics that a diagnostic system should possess in order to compare the proposed approaches against such a common set of requirements or standards.

Venkatasubramanian et al. [3] presented a set of desirable attributes that the diagnostic system should possess ideally, as following:

1. Quick detection and diagnosis

The reaction of the diagnostic technique should be fast enough to detect and diagnosis process malfunctions with reasonable small delay after their existence to prevent any serious consequences. The fulfillment of quick response goal in the face of various faults means that the diagnostic algorithm should be sensitive to noise, disturbances and modeling error, which could lead to a conflict with the tolerable performance goal under normal operating conditions.

2. Isolability

It can be defined as the ability of the diagnostic algorithm to differentiate between various faults. The isolation performance depends on the physical properties of the plant, on the size of faults, noise, disturbance and modeling uncertainties, and on the algorithm design. Further, some faults can not be isolated from other ones because they act on the physical plant in an undistinguishable way. There is a trade-off between the isolability and the rejection of the modeling uncertainties.

3. Robustness

It refers to the ability of the diagnostic algorithm to operate under the presence of noise, disturbance and modeling uncertainties, with minimal false alarms. Also, being robust means the performance of the diagnostic technique degrades gracefully instead of failing totally and abruptly.

4. Novelty identifiability

The Criterion behind this attribute can be defined as the ability of the diagnostic algorithm to decide under current conditions, whether the function of the monitored process is normal or not, and if not, whether the cause of abnormality is known malfunction, or an unknown (novel) malfunction. The designer expect that the diagnostic algorithm has the capability to recognize

the occurred novel faults and not misclassify them as one of the known malfunctions or as healthy operation.

5. Explanation facility

In addition to being capable to identify the source of malfunction, a diagnostic algorithm should also provide detailed information and explanation on how the fault originated and propagated to the current situation, which requires the ability to reason about the cause and effect relationships in the process. The operator decision and evaluation relies on the diagnostic system recommendations, so that the diagnostic system has to justify its recommendations. Moreover, a comprehensive explanation should be provided by diagnostic system such as, why certain hypotheses were proposed and why certain others were not.

6. Classification error estimation

A prior estimate on the classification error that can occur is a great characteristic that the diagnostic system could provide. Integrating the user's confidence on the reliability of the diagnostic algorithm is an important practical requirement. Such error estimations would be useful to project confidence level on the diagnostic decisions by the system which make the user feels more comfortable for the reliability of the system recommendations during its operation.

7. Adaptability

The diagnostic algorithm should be adaptable to various changes that could happen to the monitored process over the time, these changes could refereed to external inputs act on the process it self , structural changes, disturbances or even to changing in the environmental conditions. The diagnostic system should be possible to gradually develop its scope, whenever more information becomes available as new cases and problems emerge.

8. Multiple fault identifiability

The capability of the diagnostic system to identify multiple faults is an important and at the same time it is difficult task due to the interacting nature of most faults, especially for nonlinear systems, where the diagnostic system may not be able to use the individual fault patterns to model the combined effect of the faults. Moreover, for large and complex processes, it would be combinatorially forbidden to design separate diagnostic systems for various multiple fault combinations.

9. Modeling requirements

The amount of modeling requirements is an important factor in development of a diagnostic classifiers and it should be as minimal as possible, in order to perform fast and easy deployment of real-time diagnostic classifiers.

10. Storage and computational requirements

Fast real-time algorithms, usually, requires less complex computational implementations and might need high storage requirements. The diagnostic system should be able to compromise reasonably between the system performance and the above mentioned requirements.

1.2 Contributions of the Thesis

In this thesis, our goal is to develop a solution for the problem of fault detection and isolation in the actuator system of the ACS of a satellite based on neural networks approaches. The tasks of FDI have been fulfilled through two stages.

A neural residual generator is an essential part in order to generate residual errors that can reflect the real behavior of the monitored process with respect to it's working under normal conditions. This process is then followed by an adaptive neural classifier that is utilized to perform the isolation task by evaluating the generated residuals from the neural estimator in order to provide one with detailed information about occurred faults, such as the occurrence time and its location.

The contribution of this thesis in solving the above problems can be summarized as follows:

1. Developed a generic three-axis stabilized satellite control model based on the reaction wheels with no momentum under Matlab-SIMULINK

environment. Three separate PD controllers were designed and employed to command the three reaction wheels for control of each satellite axis.

2. Developed a dynamic neural network residual generator based on the Dynamic Multilayer Perceptron Network (DMLP). The developed neural observer is applied to the reaction wheel model that is commonly used as an actuator in the attitude control subsystem of a satellite. A linear model-based observer acting as a residual generator is also developed based on the linear model of the reaction wheel in order to serve as a benchmark for the comparative analysis.
3. Developed an adaptive neural network classifier based on Learning Vector Quantization (LVQ) network to be utilized as an isolation methodology.
4. The capabilities and advantages of our proposed dynamic neural network and adaptive neural classifier schemes are demonstrated under different fault scenarios and compared with a standard model-based approach.

1.3 Thesis Overview

The organization of this thesis is as follows. A neural FDI scheme is developed in Chapter 2 and a dynamic neural network structure and an adaptive neural classifier are introduced in details. An overview about the

ACS is presented in Chapter 3 in addition to modeling of a generic three-axis stabilized satellite control based on the reaction wheels and with no momentum bias using Matlab-SIMULINK. Various faulty scenarios and the corresponding simulation results for the application of our proposed dynamic network and adaptive neural classifier are presented in Chapter 4. In order to demonstrate and illustrate the capabilities of our proposed fault detection approach, a comparative evaluation of the results is performed using as a benchmark a linear model-based observer. Finally, conclusions and recommendations for future work are provided in Chapter 5.

Chapter 2

Neural Fault Detection and Isolation (FDI) Algorithm

Model-based FDI approaches rely on the mathematical models of the plant to identify the inconsistencies between the nominal and faulty operations. However, it is well known that construction of mathematical models for complex and nonlinear systems can be quite difficult and time consuming. Furthermore, a great deal of experimentation is generally required to validate the model-based approaches. Due to the limitations and difficulties with the model based approaches [4], [11], [21], in this thesis we will investigate an alternative approach that is based on using artificial neural networks. In this approach a neural FDI scheme, as depicted in Fig. 2.1., will be developed and employed to perform the detection and isolation tasks of the diagnostic system using two different structures of neural networks which can be summarized as follows:

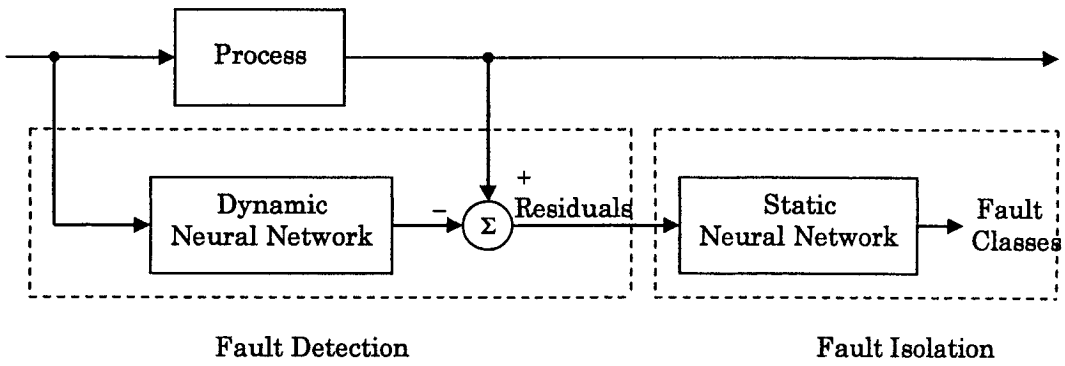


Fig. 2.1 General structure of neural FDI scheme

- Fault Detection

Fault detection or residual generation task represents the key part in FDI algorithm. The fulfillment of this task is depending mainly on the modelling of the dynamic behaviour of the monitored process, which can be performed by constructing a Dynamic Neural Network (DNN). The constructed neural networks, is called neural residual generator, should have the ability to identify all known modes of operations under healthy and faulty conditions. In other words, the generated residual error signals by the neural observer should have the ability to distinguish between the faulty and non-faulty mode of operation.

- Fault Isolation

Isolation task can be done by evaluating the generated residual signals from the neural observer, and each residual can be treated as pattern referred to a specific fault. So that the problem of fault isolation can be seen as pattern recognition or pattern classification; the idea beyond the residual evaluation

is to extract fault information from the residual itself in order to know which fault has happened and when. The neural classifier is constructed based on the static neural network. Dynamic properties are not necessary to perform the task of residual evaluation.

2.1 Dynamic Neural Networks in Literature

Recently, a great deal of attention has been paid to dynamic neural networks due to their capabilities in modeling, identification of nonlinear dynamical systems, control and filtering application [18], [24]], [25]. On the other hand, many engineering applications are still focused on using static neural networks as pattern classifiers or non-linear function approximators. The static mapping between the input-output is well suited for pattern recognition applications. Both of the input and output vectors are independent of time. However, these kind of neural networks suffer from the following drawbacks [26]:

(i) The information flows from neuron A to B, to C never comebacks to A in feedforward networks.

(ii) The structure of an artificial neuron is not dynamic in nature and performs a simple summation operation, and

(iii) The static neuron model doesn't take into account the time delays that affect the dynamics of the system. Time delays are the inherent characteristics of biological neurons.

Unlike static neural networks, dynamic neural networks employ extensive feedback between the neurons of a layer, and/or between the layers of the network. This feedback implies that the network has local memory characteristics. Because of feedback paths from their outputs to the inputs, the response of dynamic neural networks is recursive. That is, the weights are adjusted, the output is then recalculated, and the process is repeated. For stable networks, successive iterations produce smaller and smaller output changes until eventually the output become constant.

In addition to better representation of biological neural systems; DNN offers better computational capabilities compared to static ones [26].

Recently, several approaches are proposed to introduce dynamic properties to artificial neural networks. The obtained dynamic neural structures can be mainly classified into two categories: the first category encompasses dynamic neural structures which are developed based on the concept of single neuron dynamics as an extension of static neural networks, while the second category encompasses dynamic structure that developed based on the interaction between excitatory and inhibitory or antagonistic neural subpopulations [26], [29].

2.1.1 Dynamic Neuron Structures Based on Single Neuron Dynamics

Based on the concept of single neuron dynamics as an extension of static neural networks, there are four different structures that have been developed:

I. Recurrent Neural Network

Recurrent neural network, is one of the first dynamic neural network models which is introduced by Hopfield. This model consists of a single layer network included in a feedback configuration with a time delay as shown in Fig. 2.2. $y(k)$ and $y(k+1)$ represent the states at instant k and $k+1$, x_0 represents the initial value, $W(k)$ denotes the vector of the neural weights, $\Psi[.]$ is the nonlinear activation function and z^{-1} represents the time delay units which are used to learn the dynamics of the system. Thus the network is fed with current and delayed values of the process outputs. The order the system dynamics must be known beforehand in order to specify the number of time delay units [26], [28].

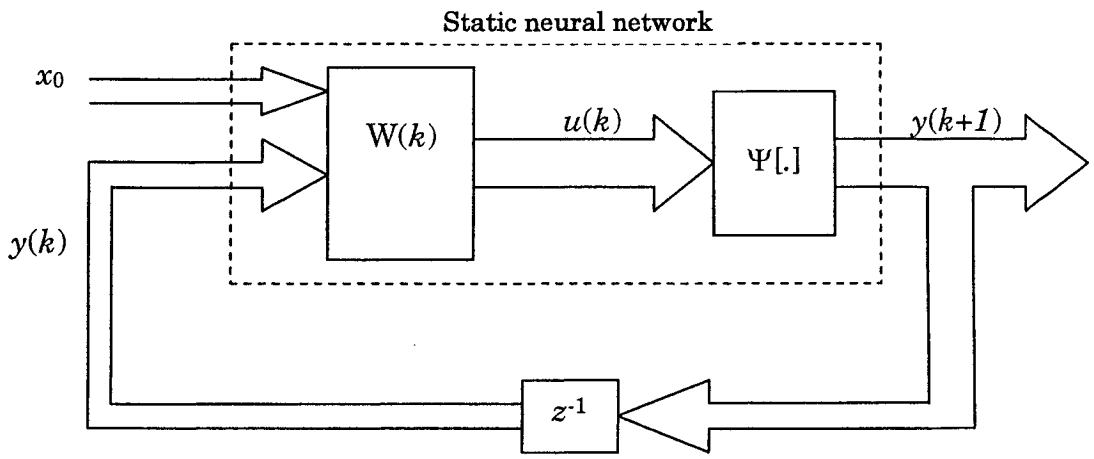


Fig. 2.2 State space model of the Hopfield neural structure [28]

Recently, however, multi-layered recurrent neural networks have been used for many applications. Hopfield neural model has been widely used in many applications such as system identification and control, robotics, machine vision, pattern recognition and associative memories. In spite of the

interesting applications that recurrent neural network have been used for, the basic architecture of the neuron is static; that is, the neuron simply provides a weighted integration of the synaptic inputs over a period of time. In other words, there are no dynamical elements within the structure of the neuron [26], [28].

II. Brain-State-in-a-Box (BSB) Neural Model

The BSB Neural model is a positive feedback system with amplitude limitation. It consists of highly interconnected set of neurons that fed back upon themselves. The BSB operates by using the built-in positive feedback to amplify an input pattern, until all neurons in the structure are driven into saturation. From the dynamic point of view, the BSB can be viewed as a discrete linear system in a saturated mode. The main difference between the BSB and usual discrete system is that the linear system is defined on $\Re^m$ while the BSB model is defined on closed n-dimensional hypercube.

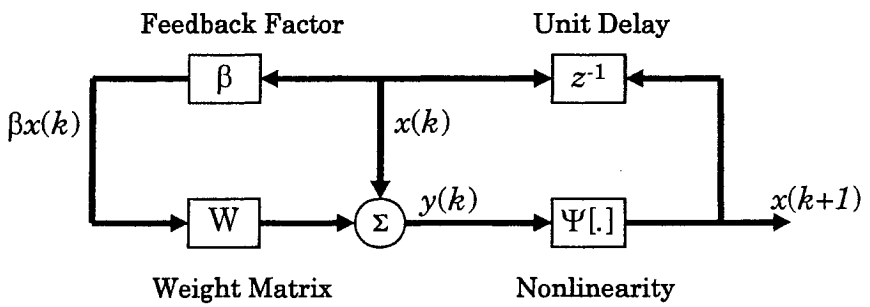


Fig. 2.3 Block diagram of the BSB model [26]

The BSB model is shown in Fig. 2.3, and is defined by equations (2.1) and (2.2),

$$y(k)=x(k) + \beta W\, x(k) \tag{2.1}$$

$$x(k+1) = \Psi[y(k)] \tag{2.2}$$

where W denotes weight matrix, $x(k)$ is the state vector of the model at discrete time k , $\Psi[.]$ is a piecewise linear function and β is a small positive constant called feedback factor.

A natural application for the BSB model is clustering. This follows from the fact that the stable corners of the unit hypercube act as point attractors with well-behaved basins of attractions. Consequently, the BSB model may be used as an unsupervised clustering algorithm, with each stable corner the hypercube representing a cluster of related data [26], [30].

III. Tim-Delay Neural Networks (TDNN)

It is possible to use a static network to process time series data by simply converting the temporal sequence into a static pattern by unfolding the sequence over time. From a practical point of view, it is possible to unfold the sequence over finite period of time. This can be accomplished by feeding the input sequence into a tapped delay line into static neural network architecture. Architecture like this is often referred to a Tim-Delay Neural Networks (TDNN). It should be noted, however, that the TDNN is a feedforward network; it attains dynamic behavior by virtue of the fact that each synapse of the network is designed as a Finite Impulse Response (FIR) filter. The TDNN neural structure has been used in many applications such as text-to-speech conversion, system identification and control of nonlinear

dynamical systems and phoneme recognition [26]. Authors in [31] have developed four different structures of the adaptive form of the time delay neural networks for identifying different classes of nonlinear system.

IV. Dynamic Neural Unit (DNU)

The DNU is a dynamic model of the biological neuron. It consists of a second order dynamics whose output constitutes the argument to a time-varying nonlinear activation function. Thus, the DNU performs two distinct operations: (i) the synaptic operation and (ii) the somatic operation. The synaptic operation involves the determination of the optimum feedforward and feedback weights, while the somatic operation determines the optimum gain (shape) of the nonlinear activation function for a given task [26].

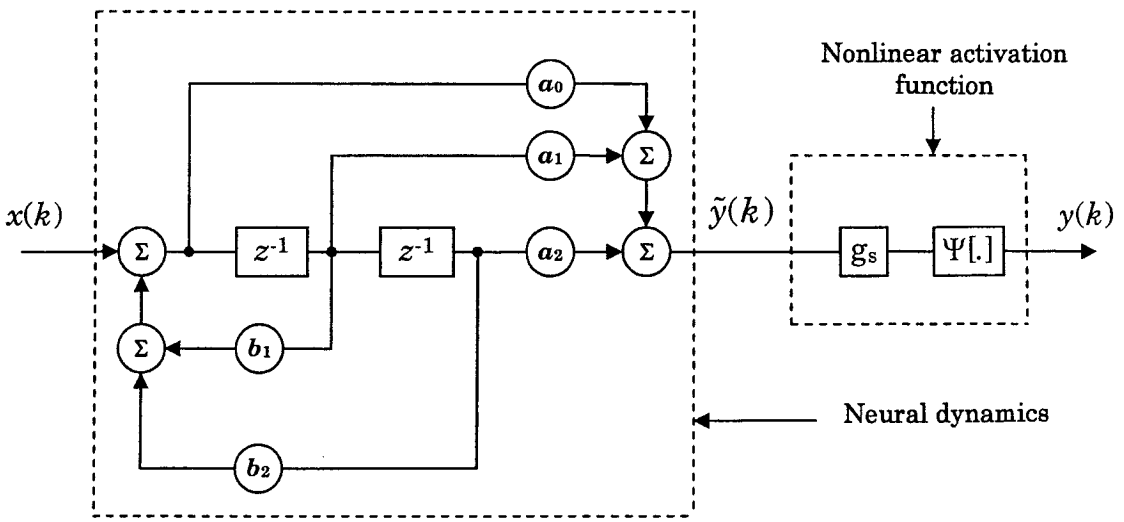


Fig. 2.4 Basic structure of DNU [34]

The DNU comprises of memory elements (delay operators), and feedforward and feedback synaptic weights representing a second-order dynamic

structure followed by a nonlinear activation function as shown in Fig. 2.4 which can be described by the following difference equation:

$$\tilde{y}(k) = a_0 x(k) + a_1 x(k-1) + a_2 x(k-2) - b_1 \tilde{y}(k-1) - b_2 \tilde{y}(k-2) \qquad (2.3)$$

where $x(k)$ denotes the neural input, $a = [a_0 \ a_1 \ a_2]$ and $b = [b_0 \ b_1]$ are the vector of adaptable feedforward and feedback weights. The nonlinear mapping operation on the output of the dynamic structure, $\tilde{y}(k)$, yields a neural output $y(k)$ given by:

$$y(k) = \Psi(g_s \cdot \tilde{y}(k)) \qquad (2.4)$$

where $\Psi[.]$ is some nonlinear activation function, and g_s is the somatic gain, the parameter which controls the slope, of the activation function [32], [33].

2.1.2 Dynamic Neuron Structures Based On Neural Subpopulations

The neural network structures described in the earlier section consider the behavior of a single neuron as the basic computing unit for describing neural information processing operations. Each computing unit in the network is based on an idealized neuron. An ideal neuron is assumed to respond optimally to the applied inputs. However, experimental studies in neuro-physiology show that the response of a biological neuron appears random, and only by averaging many observations it's possible to obtain predictable results. However, mathematical analysis has shown that these random cells

can transmit reliable information if they are sufficiently redundant in numbers [26].

The total neural activity generated within a tissue layer is a result of spatially localized assemblies of densely interconnected nerve cells called neural population, or neural mass. The neural population is comprised of neurons, and its properties have a generic resemblance to those of individual neurons. But it is not identical to them, and its properties can not be predicted from measurements on single neurons. This is due to the fact that the properties of neural population depend on various parameters of individual neurons and also depend upon the interconnections between neurons. The study of neural networks based on single-neuron analysis precludes the above two facets of biological neural structures. The conceptual gap between the functions of single neurons and those of numbers of neurons, a neural mass, is still very wide. Each neural population may be further divided into several coexisting subpopulations. A subpopulation contains a large class of similar neurons that lie in close spatial proximity. The most common neural mass is the mixture of excitatory (positive) and inhibitory (negative) subpopulations of neurons. The excitatory neural subpopulation increases the electro-chemical potential of the postsynaptic neuron, while the inhibitory subpopulation reduces the electro-chemical potential. The minimum topology of such a neural mass contains excitatory, inhibitory, excitatory-inhibitory (synaptic connection from excitatory to inhibitory), and

inhibitory-excitatory (synaptic connection from inhibitory to excitatory) feedback loops. Based on this hypothesis, a neural model named *P-N neural processor* was proposed for machine vision applications, and a *dynamic neural processor* (DNP) also was proposed for robotics and control applications [26], [35].

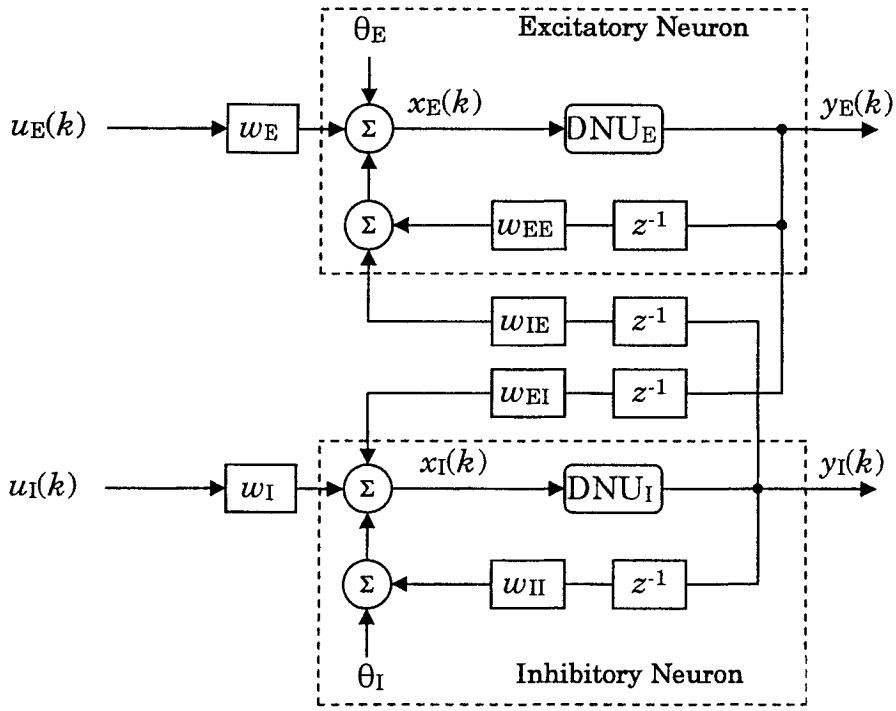


Fig. 2.5 Structure of Dynamic Neural Processor (DNP) [33]

The DNP consists of two DNUs coupled in excitatory and inhibitory modes as depicted in Fig. 2.5. The total inputs incident on the excitatory and inhibitory neural units are described by equations (2.5) and (2.6) respectively,

$$x_E(k) = w_E\, u_E(k) + w_{EE}\, y_E(k-1)\, w_{IE}\, y_I(k-1) - \theta_E \tag{2.5}$$

$$x_I(k) = w_I\, u_I(k) + w_{II}\, y_I(k-1)\, w_{EI}\, y_E(k-1) - \theta_I \tag{2.6}$$

where w_E and w_I, are the weights associated with the excitatory and inhibitory neural inputs respectively, w_{EE} and w_{II} represent the self-synaptic connection strengths, w_{IE} and w_{EI} represent the inter-neuron synaptic strengths, and θ_E and θ_I, represent the thresholds of excitatory and inhibitory neurons respectively [33].

2.2 Proposed Dynamic Neural Network for Fault Detection Problem

The main feature of a dynamic neural network is its capability of internally generating and embedding memory. This ensures that the network with its dynamic properties will be responsive to time varying signals. Dynamic properties can be introduced into a standard Multilayer Perceptron (MLP) network by two means. Since the neuron models in MLPs are static, introducing multiple recurrent connections with delays between layers provides one with a possible mechanism to change the static MLP network into a dynamic one. On the other hand, an alternative approach is to augment an Infinite Impulse Response (IIR) filter to the neuron structure in order to generate dynamics, resulting in a Dynamic Neuron Model (DNM), as obtained in [36], [37], [38] , and [39].

For our proposed dynamic neuron a linear IIR filter and a nonlinear activation module are incorporated. Taking into account the embedded dynamic characteristics of a neuron will therefore make it unnecessary to

introduce any global feedback structure. Consequently, simple feedforward architecture may be utilized for the development of the learning algorithm, and contrary to recurrent neural networks the stability analysis of the network is more straightforward.

As a matter of fact a dynamic neural network is often called locally recurrent globally feed-forward networks. In recent years, this kind of neural network has been successfully applied in process modeling and system identification [27], [34], [36], [37], [39], [40] and several fault diagnosis applications [29], [38], [41], [42], [43], [44]. One of the basic and fundamental training methods for the above specific dynamic network is the Extended Dynamic Back-propagation Algorithm (EDBA) [36]. Furthermore, starting from a relatively small structure we may develop an optimal architecture for the proposed dynamic network by incrementally increasing the number of hidden neurons until desired performance specifications are obtained [45], [46].

2.2.1 Dynamic Neuron Model (DNM)

A generalized structure of the dynamic neuron model that is proposed in [38] is considered here. The structure of the proposed dynamic neuron is a generalization of the conventional static model accomplished by adding an Infinite Impulse Response (IIR) filter to the neuron transfer function. Such a model with internal filter dynamics introduces appropriate dynamics for the neuron mapping and "transfer function".

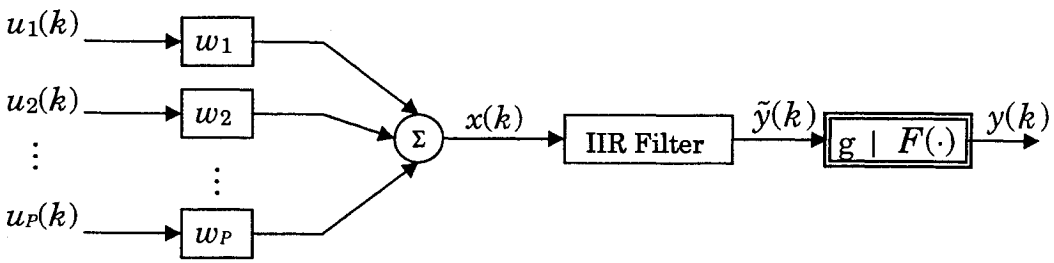


Fig. 2.6 General structure of the DNM with P inputs [43]

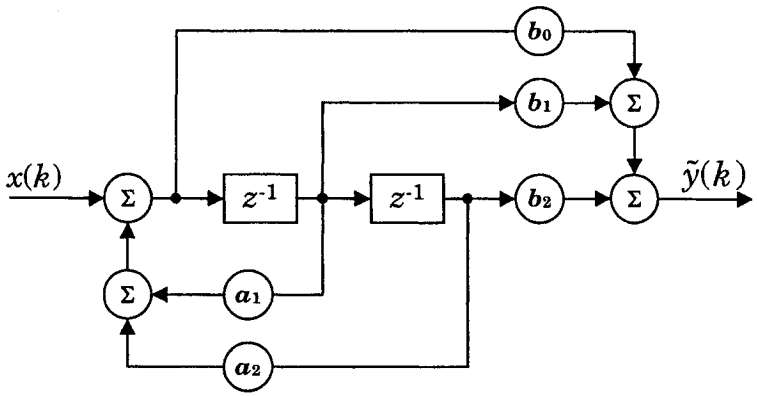


Fig. 2.7 Structure of second order IIR filter [34]

Due to the introduction of the internal filter, the general neuron activity will now depend on its internal states and therefore, the neuron does indeed process past values of its own activity $y(k)$ and its inputs $u_p(k)$ for p=1,2,...,P, where P is the number of the inputs and k is the discrete time steps (samples). Fig. 2.6 shows the structure of the proposed neuron that is designated as the Dynamic Neuron Model (DNM), with P inputs; also the three main operations are performed in this dynamic neuron structure. First, the weighted sum of the inputs is calculated according to the expression [43]:

$$x(k) = w^T u(k) = \sum_{p=1}^{P} w_p u_p(k) \tag{2.7}$$

where $w = [w_1\, w_2 \ldots w_P]^T$ denotes the input-weight vector, P is the number of inputs, and $u(k) = [u_1(k)\ \ u_2(k) \ldots u_P(k)]^T$ is the input vector (T denotes the transpose operator). Hence, the computed weighted sum of the inputs $x(k)$ is passed through the IIR filter. The corresponding characteristics of the filter can be described by the following difference equation:

$$\tilde{y}(k) = \sum_{i=0}^{n} b_i x(k-i) - \sum_{i=1}^{n} a_i \tilde{y}(k-i) \tag{2.8}$$

where $x(k)$ denotes the filter input, $a = [a_1\ a_2 \ldots a_n]^T$ and $b = [b_0\ b_1 \ldots b_n]^T$ are feedback and feedforward paths weighted by the vector weight, n denotes the filter order, and $\tilde{y}(k)$ denotes the filter output. In our proposed dynamic neural network, the filters under consideration are LTI dynamic systems of second order as shown in Fig. 2.7.

Consequently, the neuron output can be formulated as:

$$y(k) = F(g \cdot \tilde{y}(k)) \tag{2.9}$$

where $F(\cdot)$ is the nonlinear activation function that produces the neuron output $y(k)$ and g is parameter of the activation function defining its slope. Due to the adaptive nature of the parameter g, the dynamic neuron can better model the biological neuron. Thus, the proposed dynamic neural network is more elastic. Introduction of the slope parameter g to the activation function operation can be very helpful, particularly in the case of nonlinear squashing activation functions, i.e.: sigmoidal or hyperbolic tangent. In the case when the magnitude of the data is large, the activation function drives into its own saturation range and response of the neuron

could become a constant value or a signal with negligible small amplitude. This is a very undesirable effect, which can be compensated by application of the adjustable slope parameter g [40].

Owing to the neuron's internal dynamic properties the DMLP processes the modeled system measurements at the currant time instant k thereby reducing the input space of trained network in comparison with the Elman and other recurrent DML networks. The dynamic MLP under consideration doesn't require past values of the process measurements [36].

2.2.2 Extended Dynamic Backpropagation Algorithm

The proposed dynamic neural network can have the same structure as a standard feedforward backpropagation network. The calculated output error is propagated back to the input layer through the hidden layers containing dynamic filters, where the extended dynamic backpropagation algorithm can be defined and it can operate in both modes of training, that is, on-line or off-line [13].

Consider a M-layered network with the same dynamic neurons described by the differentiable activation function $F(\cdot)$. In Fig. 2.8, S_m denotes the number of the neurons in the m-th layer and $u_s^m(k)$is the output of the s-th neuron of the m-th layer at discrete time k ($m = 1...M$, $s = 1... S_m$). The activity of the s-th neuron in the m-th layer is defined by [36]:

$$u_s^m(k) = F(\tilde{y}_1(k)) = F(g_s^m \cdot \tilde{y}(k)) = F\left(g_s^m \left(\sum_{i=0}^{n} b_i x(k-i) - \sum_{i=1}^{n} a_i \tilde{y}(k-i) \right) \right)$$

$$= F\left(g_s^m \left(\sum_{i=0}^{n} b_{is}^m x(k) \sum_{p=1}^{S_{m-1}} w_{sp}^m u_p^m(k-i) - \sum_{i=1}^{n} a_{is}^m \tilde{y}_s^m(k-i) \right) \right) \qquad (2.10)$$

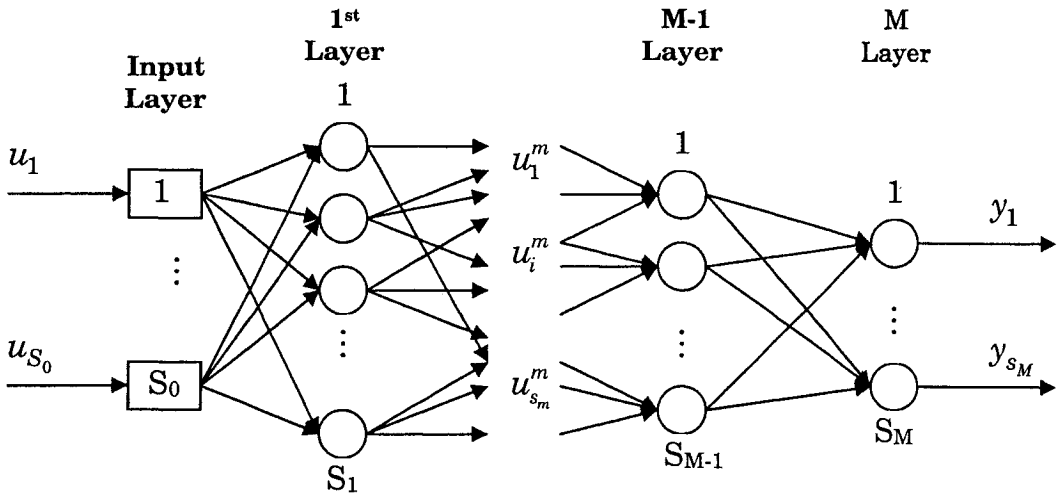


Fig. 2.8 The M-layered feedforward neural network [36]

Based upon a given set of input-output training pairs, the main objective of the learning process is to adjust all the unknown network parameters; which include the weight matrix (w), the filter parameters matrix (a, b), and the slope parameter matrix (g); where

- Weight matrix $[w_{sp}^m]$: $m = 1 \ldots M$, $s = 1 \ldots S_m$, $p = 1 \ldots S_{m-1}$
- Filter feedback parameter matrix $[a_{is}^m]$: $m = 1 \ldots M$, $s = 1 \ldots S_m$, $i = 1 \ldots n$
- Filter feedforward parameter matrix $[b_{is}^m]$: $m = 1 \ldots M$, $s = 1 \ldots S_m$, $i = 0 \ldots n$
- Slop parameter matrix $[g_s^m]$: $m = 1 \ldots M$, $s = 1 \ldots S_m$

In both static and dynamic neural networks, the objective is to determine an adaptive algorithm or a rule which adjust the parameters of the network

based on the network on a given set of input-output pairs. The idea of error backpropagation is widely applied for that purpose in static context with extension to dynamic ones. To define an EDBP algorithm, the standard approach can be applied. Assuming that unknown parameters vectors w, a, b and g are considered as elements of a parameter vector v, the learning process involves the determination of the vector v^* which optimizes a performance index J based upon the error function $e(k)$ which may be defined as [36]:

$$J = \frac{1}{2}\sum_{k=0}^{N} e(k)^2 = \frac{1}{2}\sum_{k=0}^{N} (y_d(k) - y(k))^2 \tag{2.11}$$

where $e(k)$ denotes the output error defined as a difference between the desired response $y_d(k)$ and the actual response $y(k)$.

The adjustment of the parameters of the s-th neuron in the m-th layer according to the EDBP algorithm has the form:

$$v_s^m(k+1) = v_s^m(k) + \eta \delta_s^m(k) S_{vs}^m(k) \tag{2.12}$$

where $v = [w, a, b, g]$ represents the unknown generalized parameter vector, η is the learning rate, δ_s^m is the generalized output error which is described below for both hidden and output layers, and $S_{vs}^m(k)$ denotes the sensitivity function for the elements of the unknown generalized parameter v.

The generalized output error is described as follows [36]:

- Hidden layers generalized output error:

$$\delta_s^m(k) = \sum_{z=1}^{S_{m+1}} (\delta_s^{m+1}(k) g_z^{m+1} b_{0z}^{m+1} w_{zs}^{m+1}) F'(\tilde{y}_{1s}^m(k)) \tag{2.13}$$

- Output layer generalized output error:

$$\delta_s^M(k) = e_s(k)F'(\tilde{y}_{1s}^M(k)) \tag{2.14}$$

In turn the sensitivity function $S_{vs}^m(k)$ is defined as follows:

- Sensitivity with respect to weight parameter w_{sp}^m:

$$S_{w_{ps}}^m(k) = g_s^m\left(\sum_{i=0}^{n} b_{is}^m u_p^m(k-i) - \sum_{i=1}^{n} a_{is}^m S_{w_{ps}}^m(k-i)\right) \tag{2.15}$$

- Sensitivity with respect to feedback parameter a_{is}^m:

$$S_{a_{is}}^m(k) = -g_s^m \tilde{y}_s^m(k-i) \tag{2.16}$$

- Sensitivity with respect to feedforward parameter b_{is}^m:

$$S_{b_{is}}^m(k) = g_s^m x_s^m(k-i) \tag{2.17}$$

- Sensitivity with respect to slop parameter g_s^m:

$$S_{g_s}^m(k) = \tilde{y}_s^m(k) \tag{2.18}$$

Based on equations (2.12) – (2.18) we may rewrite the updating laws for each network adaptable parameter as follows:

- **Hidden layers parameters:**
  - Weight parameter w_{sp}^m:

$$w_{sp}^m(k+1) = w_{sp}^m(k) + \eta\left(\sum_{z=1}^{S_{m+1}} (\delta_s^{m+1}(k) g_z^{m+1} b_{0z}^{m+1} w_{zs}^{m+1}) F'(\tilde{y}_{1s}^m(k))\right) g_s^m\left(\sum_{i=0}^{n} b_{is}^m u_p^m(k-i) - \sum_{i=1}^{n} a_{is}^m S_{w_{ps}}^m(k-i)\right) \tag{2.19}$$

– Filter feedback parameter a_{is}^{m}:

$$a_s^m(k+1) = a_s^m(k) - \eta\left(\sum_{z=1}^{S_{m+1}}(\delta_s^{m+1}(k) g_z^{m+1} b_{0z}^{m+1} w_{zs}^{m+1}) F'(\tilde{y}_{1s}^m(k))\right) g_s^m \tilde{y}_s^m(k-i) \tag{2.20}$$

– Filter feedforward parameter b_{is}^{m}:

$$b_s^m(k+1) = b_s^m(k) + \eta\left(\sum_{z=1}^{S_{m+1}}(\delta_s^{m+1}(k) g_z^{m+1} b_{0z}^{m+1} w_{zs}^{m+1}) F'(\tilde{y}_{1s}^m(k))\right) g_s^m x_s^m(k-i) \tag{2.21}$$

– Slop parameter g_s^m:

$$g_s^m(k+1) = g_s^m(k) + \eta\left(\sum_{z=1}^{S_{m+1}}(\delta_s^{m+1}(k) g_z^{m+1} b_{0z}^{m+1} w_{zs}^{m+1}) F'(\tilde{y}_{1s}^m(k))\right) \tilde{y}_s^m(k) \tag{2.22}$$

- **Output layer parameters:**

– Weight parameter w_{sp}^{m}:

$$w_{sp}^m(k+1) = w_{sp}^m(k) + \eta\left(e_s(k) F'(\tilde{y}_{1s}^M(k))\right) g_s^m \left(\sum_{i=0}^{n} b_{is}^m u_p^m(k-i) - \sum_{i=1}^{n} a_{is}^m S_{w_{ps}}^m(k-i)\right) \tag{2.23}$$

– Filter feedback parameter a_{is}^{m}:

$$a_s^m(k+1) = a_s^m(k) - \eta\left(e_s(k) F'(\tilde{y}_{1s}^M(k))\right) g_s^m \tilde{y}_s^m(k-i) \tag{2.24}$$

– Filter feedforward parameter b_{is}^{m}:

$$b_s^m(k+1) = b_s^m(k) + \eta\left(e_s(k) F'(\tilde{y}_{1s}^M(k))\right) g_s^m x_s^m(k-i) \tag{2.25}$$

– Slop parameter g_s^{m}:

$$g_s^m(k+1) = g_s^m(k) + \eta\left(e_s(k) F'(\tilde{y}_{1s}^M(k))\right) \tilde{y}_s^m(k) \tag{2.26}$$

2.3 Proposed Static Neural Network for Fault Isolation Problem

As mentioned at the beginning of this chapter the classification duties will be performed by a static neural network. The static neural classifier will treat the residual error signals generated from the neural observer as patterns and so that the pattern classification is achieved.

In pattern classification the first and most important stage is feature extraction. The extracted feature from the content of the input data, which will be classified, is obtained ordinarily in an unsupervised manner. The self-organizing map (SOM) is well suited for the purpose of feature extraction, particularly if the input data are generated by a nonlinear process.

The actual classification is the second stage of pattern classification, during this stage the extracted features from the input data are assigned to individual classes. While the SOM has the ability to perform the classification task too, the recommended procedure for best performance can be obtained by integrating it with a supervised learning scheme for the second step of classification. The mixed combination between a supervised and an unsupervised technique (SOM) yields the basis of an *adaptive pattern classification* that is hybrid in its nature [47].

An adaptive pattern classifier can be obtained through using a learning vector quantizer as a supervised learning scheme. As shown in Fig. 2.9 the second stage of adaptive pattern classification is provided by learning vector

quantizer, which provides in its turn, a mechanism for the final fine tuning of a feature map which is performed by a SOM algorithm.

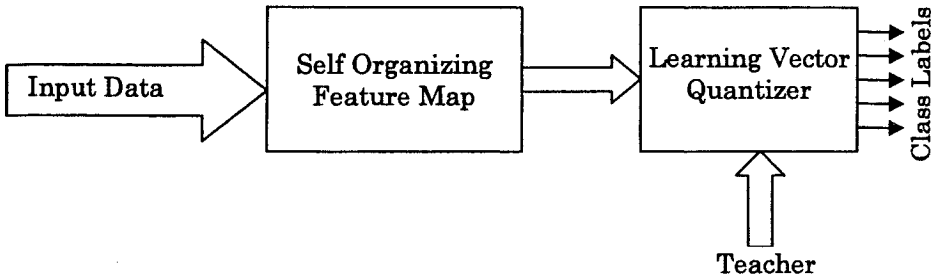


Fig. 2.9 Block diagram of adaptive pattern classification [47]

The structure of a Learning Vector Quantization (LVQ) network can be constructed using two layers as shown in Fig. 2.10; a first competitive layer and a second linear layer. The competitive layer learns to classify input vectors while the linear layer transforms the competitive layer's classes into predefined target classifications. We refer to the classes learned by the competitive layer as subclasses and the classes of the linear layer as target classes. Both the competitive and linear layers have one neuron per (sub or target) class. The constructed classifier has a simple structure and a non-complicated training algorithm [47, 48].

In the LVQ network, each neuron in the first layer is assigned to a class, with several neurons often assigned to the same class. Each class is then assigned to one neuron in the second layer. The number of neurons in the first layer, S^1, will therefore always be at least as large as the number of neurons in the second layer, S^2, and usually will be larger [30], [48].

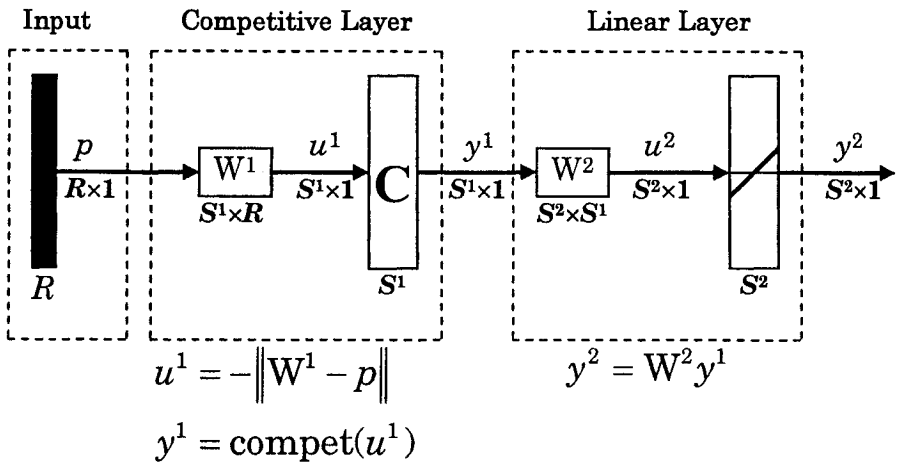


Fig. 2.10 The architecture of LVQ network [48]

The network input of the first (competitive) layer of the LVQ will be

$$u_i^1 = -\left\|\mathrm{W}_i^1 - p\right\| \tag{2.27}$$

or, in vector form,

$$u^1 = -\begin{bmatrix} \left\|w_1^1 - p\right\| \\ \left\|w_2^1 - p\right\| \\ \vdots \\ \left\|w_{S^1}^1 - p\right\| \end{bmatrix} \tag{2.28}$$

where W^1 represents the input weight matrix, i denotes the corresponded neuron, and p represents the input vector. The output of the first layer of the LVQ is given by

$$y^1 = \mathrm{compet}(u^1) \tag{2.29}$$

Therefore the neuron whose weight vector is closest to the input vector will output one, and the other neurons will output zero. Thus, the winning neuron

indicates a subclass, rather than a class as in competitive networks. There may be several different neurons (subclasses) that make up each class.

The second layer of the LVQ network is used to combine subclasses into a single class which is done by the weight matrix W^2. The columns of W^2 represent subclasses, and the rows represent classes. W^2 has a single 1 in each column, with the other elements set to zero. The row in which the 1 occurs indicates which class the appropriate subclass belongs to, in other words,

$$w_{ki}^2 = 1 \Rightarrow \text{Subclass } i \text{ is a part of class } k$$

The combining process of subclasses to form a class allows the LVQ network to create complex class boundaries which overcome the limitations of the standard competitive layers [30], [48].

2.3.1 LVQ learning

The learning in the LVQ network combines competitive learning with supervision. The learning in the competitive layer is based on a set of input/target pairs:

$$\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\}$$

Each target vector has a single 1. The rest of its elements are 0. The row in which the 1 is existed indicates the proper classification of the associated input.

Before learning occurs, each neuron in the second layer is assigned to an output neuron. This generates the matrix W^2. Typically, equal numbers of

hidden neurons are connected to each output neuron, so that each class can be made up of the same number of convex regions. All elements of W^2 are set to zero except for the following:

If hidden neuron i is to be assigned to class k, then set $w_{ki}^2 = 1$

Once W^2 is defined, it will never be altered. The hidden weights W^1 are trained with a variation of the Kohonen rule.

The LVQ learning rule proceeds as follows:

- At each iteration, an input vector p is presented to the network, and the distance from p to each prototype vector is computed.
- The hidden neurons compete, neuron i^* wins the competition, and the i^{*th} element of y^1 is set to 1.
- Then, y^1 is multiplied by W^2 to get the final output y^2, which also has only one nonzero element, k^*, indicating that p is being assigned to class k^*.
- We adjust the i^{*th} row of W^1 in such a way as to move this row closer to the input vector p if the assignment is correct, and to move the row away from p if the assignment is incorrect.
- if p is classified correctly, $(y_{k^*}^2 = t_{k^*} = 1)$, then we compute the new value of the i^{*th} row of W^1 as:

$$w_{i^*}^1(q) = w_{i^*}^1(q-1) + \alpha(p(q) - w_{i^*}^1(q-1)) \tag{2.30}$$

- if p is classified incorrectly, $(y_{k^*}^2 = 1 \neq t_{k^*} = 0)$, we compute the new value of the i^{*th} row

$$w_{i^*}^1(q) = w_{i^*}^1(q-1) - \alpha(p(q) - w_{i^*}^1(q-1)) \tag{2.31}$$

where α is the learning rate. The result will be that each hidden neuron moves towards vectors that fall into the class for which it forms a subclass, and away from vectors that fall into other classes [30], [48].

2.4 Conclusions

In this chapter, the neural fault detection and isolation scheme has been presented which is based on two different neural architectures, namely, static and dynamic neural network. A review on dynamic neural networks structures has been reported. Dynamic Multilayer Perceptron (DMLP) network has been proposed as a fault detection tool. Also, the learning vector quantization network has been proposed as fault isolation tool. In the next Chapter, a review on the Attitude Control Subsystem (ACS) of a satellite will be presented along with a detailed description of the Matlab-SIMULINK model of the ACS that has been developed in this thesis. Afterwards, in Chapter 4, simulation results will be presented under different actuator failures scenarios.

Chapter 3

Modeling of Attitude Control Subsystem (ACS) of a Satellite

Generally, satellites have provided various classes of service to mankind in both of military and civil missions; such services as: weather forecasting, providing communications to any point on the face of the earth and many other missions.

The satellite is traditionally subdivided into eight subsystems, and each subsystem is responsible to perform specific functions. A highly simplified functional satellite block diagram is shown Fig. 3.1. This diagram is a useful, concise way to visualize the relationships between the satellite subsystems. Rich details regarding satellite structure and subsystems can be found in [49], [51], [52], and [53].

In this chapter we will focus on modeling of one of the important spacecraft's subsystems which is the attitude control subsystem (ACS).

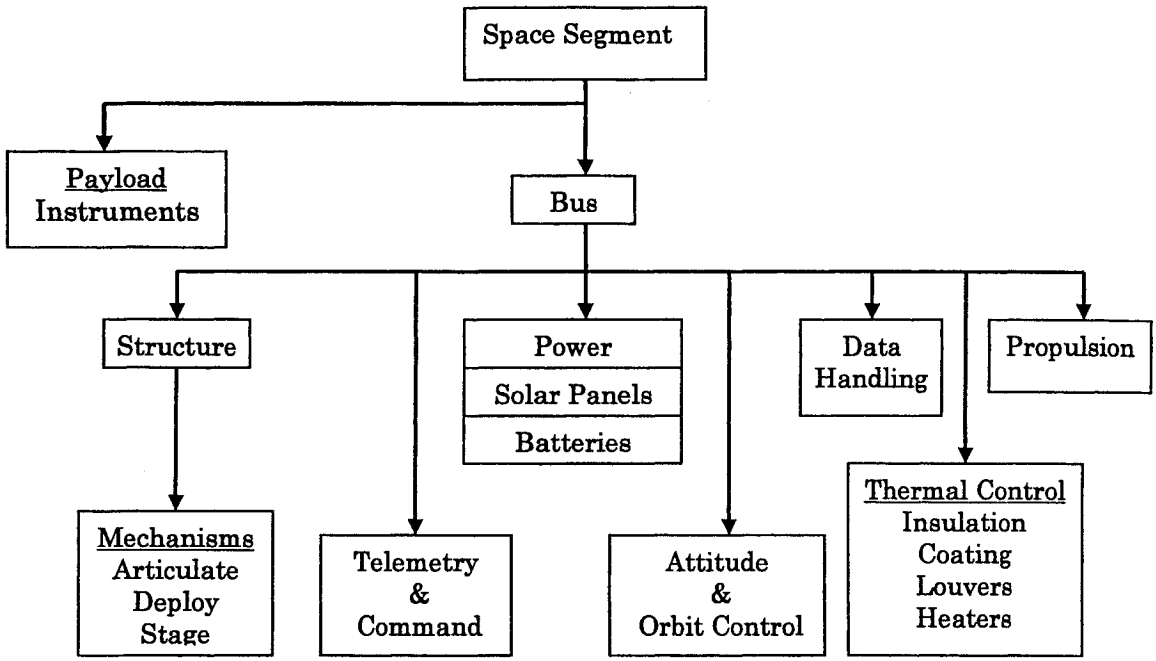


Fig. 3.1 Simplified spacecraft block diagram [49], [51]

3.1 ACS Overview

Attitude control deals with the orientation of the spacecraft axes with respect to an inertial reference frame. An instantaneous spacecraft attitude is commonly described by a *pitch* angle, a *roll* angle, and a *yaw* angle. Spacecraft attitude is measured as an angular deviation of the spacecraft body axes from the inertial coordinates. The attitude control subsystem controls the vehicle body axes such that the error in *pitch*, *roll* and *yaw* angles are within defined limits.

The attitude control task can be divided into three subtasks:

1- Measuring attitude, which is done by attitude sensors such as gyroscope;

2- Correcting attitude, which is done by torqures or actuators such as thrusters and reaction wheel; and

3- A control law, which is software that determines the magnitude and direction of torque in response to a given disturbance.

The ACS, conceptually simple is a classic feedback control system as shown in Fig. 3.2. If the spacecraft drifts off the desired attitude, the sensors detect the error; the control law determines the magnitude of the response and directs an actuator to correct it [49].

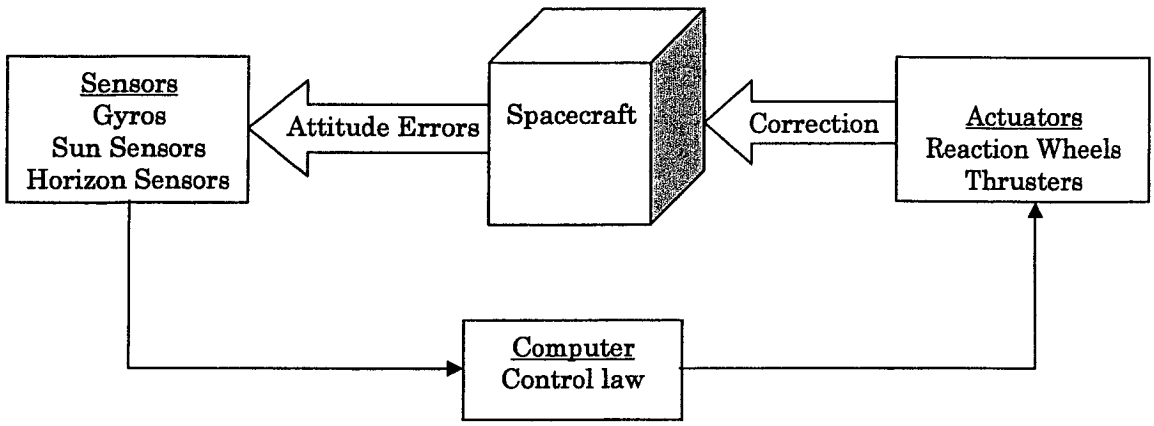


Fig. 3.2 Attitude Control Operation [49]

3.2 ACS Actuators

3.2.1 Reaction Wheels

The reaction wheel is in essence a momentum transfer and storage device which provides reaction torque to the vehicle and store angular momentum. It's simply small rotating flywheel driven by an internal DC motor, which

exchange momentum with the vehicle by changing wheel speed. While the dc motor reduces wheel speed, the reaction wheel momentum is dumped by holding the vehicle stationary with thrusters, or magnetic torquers. The use of reaction wheels permits fine and damped attitude control. A zero momentum configuration can be obtained when the reaction wheels are placed on each of the three principles axes, the control law for each axis will be linear, however, it is common practice to add a redundant wheel by placing a fourth wheel in a position oblique to all axes. The function of this arrangement is to accommodate any single wheel failure by the oblique wheel, which is more efficient than providing three redundant wheels [49], [50], [52].

3.2.2 Momentum Wheels

The main difference between the reaction wheels and momentum wheels is speed bias because the flywheels are designed to operate at biased and nonzero speed. Spacecraft can be stabilized using momentum bias system by placing a momentum wheel along pitch axis [49], [52].

3.2.3 Control Momentum Gyros (CMGs)

CMGs may be used instead of reaction wheels for high torque applications; they are single or double-gimbaled wheels spinning at a constant rate. A control torque on the output axis of the spacecraft can be obtained by

applying a commanded force to the input of the gyro. Its disadvantages comes from their heavy weights and large size, more power consuming with respect to reaction wheels, they are seldom used on smaller spacecraft. Necessity for complex control law and a desaturation mechanism, also, are considered as other disadvantages [49], [52].

3.2.4 Magnetic Torquers

Also it's called torque rods, a torque rod is simply a wire coil wrapped around a rod and it takes the advantage the magnetic filed of earth to generate a correcting force on a spacecraft. These devices perform desaturation for momentum-exchange systems and compensation for the spacecraft's residual magnetic filed or attitude drifts for minor disturbance torques. Because of their dependency on earth's natural magnetic filed, they are not usable for deep space missions [52], [53].

3.2.5 Thrusters

Most spacecraft use thrusters as an actuators; they provide momentum to the spacecraft by ejecting mass overboard in the form high velocity exhaust gas. Three types of thrusters are in use, which are cold gas, monopropellant hydrazine and bipropellant. Thrusters can be used directly to control the spacecraft attitude or used as momentum desaturation actuators for the reaction wheels [52], [53].

3.3 Attitude Control Techniques

The criteria of choosing the controlling technique of a spacecraft depending mainly on how we define the attitude control subsystem requirements. The most common ACS techniques, along with typical characteristics are discussed below [49], [52], [53].

3.3.1 Spin Stabilization

A basic passive technique is that of spin stabilization. In this mode the entire spacecraft spins around the axis with highest moment of inertia. A spin stabilized spacecraft takes the advantage of inherent resistance of a spinning body to disturbance torques. If no external disturbance torques are experienced, the angular momentum vector remains fixed in space, constant in both magnitude and direction. If a disturbance torque occurs that is parallel to the momentum vector, the spin rate will be affected but not the attitude. Thrusters are used to correct the spin rate. Disturbance torques perpendicular to the momentum vector will cause the spin axis to precess; thruster force can be used to remove precession.

Spin stabilization is useful in a number of special cases where reliability and simplicity are more important than operational flexibility. Also it takes the advantage of being low cost, modest pointing accuracy and minimal maneuvering. Sensors gyros, momentum exchange devices, and onboard computers are unnecessary on a spinner. Substantial cost and mass savings

result. While their disadvantage appears in the low accuracy 0.3 – 1°, in addition to the tight control of moments of inertia is required [49], [52], [53].

3.3.2 Dual Spin

Dual-spin stabilization is a compromise design where the spacecraft has two sections spinning at different rates about the same axis (Galileo is an example of dual-spin spacecraft). Normally one section of the spacecraft, the rotor, spins rapidly to provide angular momentum while de-spinning is performed by the second section, stator or platform, to keep one axis pointed for antenna and instruments toward the earth or sun. By combining inertially fixed and rotating sections, dual-spinners can accommodate a variety of payloads in a simple vehicle. Providing both scanning and pointing capabilities can be considered as advantage for dual spinning but it's suffer from the following disadvantages; sensitivity to mass properties, if the accuracy is required then the cost and complexity can equal or exceed three axis method (which will be discussed in the next section) [49], [52], [53].

3.3.3 Three-Axis Stabilization

Spacecraft stabilized in three-axes are more common today than those using spin or gravity gradient. This method of stabilizing maintains actively the vehicle axis system aligned with a reference system, usually inertial reference (gyros) or nadir reference. The control torques about the axes of

three-axis system come from combinations of ACS actuators. Many advantages can be obtained such as:

- Unlimited pointing capability in any direction – nadir, inertial, sun, scanning – and provides the high pointing accuracy, limited only by sensors accuracy, which reaches more than 0.001°,
- Ability to provide rapid maneuvering,
- Accommodating large power requirements,
- Unlimited payload pointing.
- Most adaptable to changing mission requirements

In spite of the mentioned advantages, three-axis method is the most expensive one, also its hardware are complex, heavy, consumes high power, expensive and failure sources. Spacecrafts that have used three-axis controlled include Magellan, Hubble telescope and Global Positioning System (GPS) [49], [52], [53].

3.3.4 Gravity-Gradient

Gravity-gradient technique utilizes the inertial properties of the spacecraft to keep it pointed towards the Earth, which is based on the fact that elongated object in a gravity field tends to align it's longitudinal axis through the Earth's centre. In order to work this technique, it is necessary that the gravity gradient torques are greater than any disturbance torque; this criteria can be done in orbits lower than 1000 Km. Also it is necessary for the

moment of inertia about x and y axis to be much greater than the moment of inertia about z axis. Note that gravity gradient doesn't stabilize *yaw* axis and it does only *pitch* and *roll* axes. It is common practice to use a momentum wheel with its axis perpendicular to orbit plane to provide stiffness in yaw. Gravity-gradient control torques are small at best. Slow oscillation caused by disturbances can be prevented by adding an active damping. Gravity-gradient stabilization is useful when long life and high reliability are required and pointing requirements are modest. GEOSAT is one of the spacecraft which is controlled by gravity gradient [49], [52], [53].

3.3.5 Momentum Bias

In this technique a momentum wheel is used to provide stiffness in two axes and control the wheel speed provides control in the third axis. Particularly, this system is useful for nadir pointing spacecraft using wheel speed to hold z axis on nadir. The advantages appears in simplicity and good ability for long-life mission, also its cheaper that a three-axis system and it offers good pointing in one axis (usually *pitch*) and poor pointing accuracy in the other wheel axes (usually *yaw* and *roll*). Compared to the three-axis control, momentum biases cannot achieve its pointing accuracy. Table 3.1 summarizes several different methods of spacecraft control [52].

TABLE 3.1
SUMMARY OF ATTITUDE CONTROL TECHNIQUES [52]

Method	Accuracy	Remarks
Spin Stabilization	0.3 – 1°	Passive, simple, low cost, inertially oriented
Gravity Gradient	1 – 3°	Passive, simple, low cost, central-body oriented
Reaction Wheels	0.01°	Quick, coastally, high precision
CMGs	0.1°	High authority, quick, heavy, costly
Magnetic Torquers	1 – 2°	Near-earth usage, slow, light weight, low cost

3.4 Space Vehicle Disturbance Torques

In outer space, unmanned space vehicles are expected to operate under significant and numerous disturbances in terms of relative and absolute magnitude which affect the space vehicle orientation, mass properties, and design symmetry. Estimation of the influences of these disturbances is an essential part of the ACS design. More precisely, disturbance torques affect actuator size and momentum storage requirements. Four types of worst case disturbance torques and the simplified equations of its estimation are provided in [52], these disturbances are:

- Gravity Gradient
- Solar Radiation
- Magnetic Field
- Aerodynamics

3.5 Developed ACS Model

As discussed earlier in this chapter, an attitude control subsystem is composed of three major parts:

- Attitude sensors, which provide direct measurements of spacecraft attitude.
- Feedback control system, which corrects measured attitude to desired attitude.
- Actuators, which provide a corrective control torques.

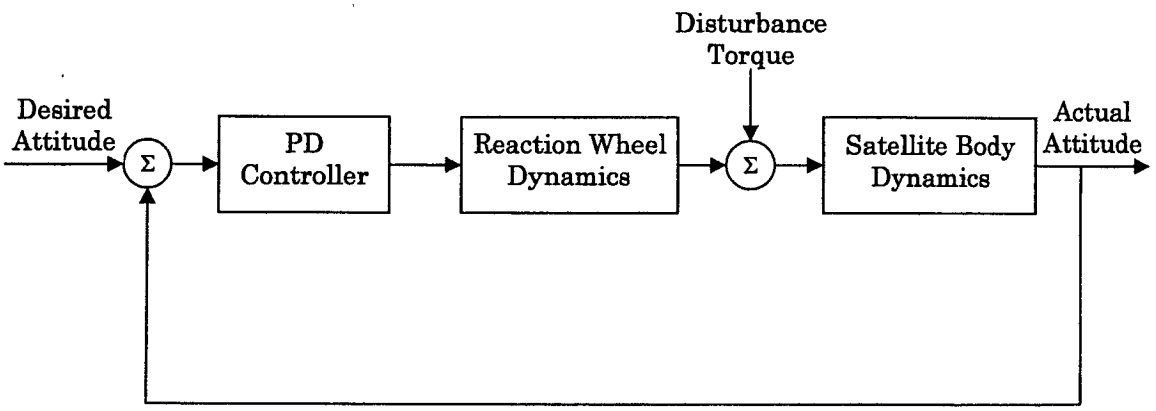


Fig. 3.3 Single axis attitude control feedback system [49]

For preliminary design phase, a single axis attitude control for *pitch* axis is considered as illustrated in Fig. 3.3. Each block in the attitude control feedback system is described mathematically by a transfer function. We assumed an ideal sensor without delays in the ACS, i.e. a unity feedback gain will be presented instead of the sensor dynamics in the feedback path. Unknown external disturbance torque is presumed to act on the spacecraft [52]. The transfer function which describes the dynamics of the reaction

wheel, as provided in equation (3.1), is calculated based on the linear model of the reaction wheel that is presented in [53].

$$R(s) = \frac{G_d K_t s}{s + \frac{G_d K_t K_e + \tau_v}{J}} \tag{3.1}$$

Note that all variables in equation (3.1) are defined and given in Table 3.2.

TABLE 3.2
CONSTANTS OF ITHACO'S (TYPE A) REACTION WHEEL [54]

Variable	Nomenclature	Unit	Value
J	Flywheel inertia	N.m.s^2	0.0077
G_d	Driver gain	A/V	0.190
K_t	Motor torque constant	N.m/A	0.029
K_e	Motor back-EMF constant	V/rad/s	0.029
K_s	Over-speed circuit gain	V/rad/s	95
ω_s	Over-speed circuit threshold	rad/s	690
τ_c	Coulomb friction	N.m	0.002
N	Number of motor poles	–	36
B	Motor torque ripple coefficient	–	0.22
R_{in}	Input resistance	Ω	2.00
	Torque command range	V	±5
K_f	Voltage feedback gain	V/V	0.50
P_q	Quiescent power	W	3.00
ω_a	Torque noise high pass filter frequency	rad/sec	0.20
θ_a	Torque noise angle deviation	rad	0.05
R_b	Bridge resistance	Ω	2.00

The importance of this transfer function comes from the necessity of preliminary design phase fulfillment of the ACS. While the transfer function

of the linear satellite body dynamics is given by equation (3.2). Finally, the controller is designed and tuned based on the ACS requirements. Detailed description for each dynamics is given below.

3.5.1 Satellite Body Dynamics

A rigid and decoupled system is assumed by expressing the satellite inertia matrix about the principal axes which leads to a linear body dynamic transfer function given by equation (3.1),

$$G(s) = \frac{1}{I_{yy}s^2} \tag{3.2}$$

where I_{yy} is the body moment of inertia of the satellite pitch axis

3.5.2 Reaction Wheel Dynamics

The nonlinear model of the reaction wheel will be presented in this section. The angular momentum stored in the flywheel, H, is the product of the flywheel inertia, and the wheel speed ω, simply as:

$$H = J\omega \tag{3.3}$$

Net torque (τ_{net}), which is developed by the reaction wheel can be defined as the difference between the motor torque (τ_m), and all the torques due to friction, torque noise and motor disturbances (τ_d).

$$\tau_{net} = \tau_m - \tau_d \tag{3.4}$$

The reaction torque applied to the spacecraft is equal and opposite to the net torque which accelerate or decelerate the flywheel. So that the reaction torque can be derived at any time based on the Newton's second law from the negative rate change angular momentum, which leads to the derivative of the flywheel inertia and the flywheel polar as [54]:

$$\text{Reaction torque} = \tau_{net} = -\frac{\partial H}{\partial t} = -\frac{\partial (J\omega)}{\partial t} = -J\frac{dw}{dt} \tag{3.5}$$

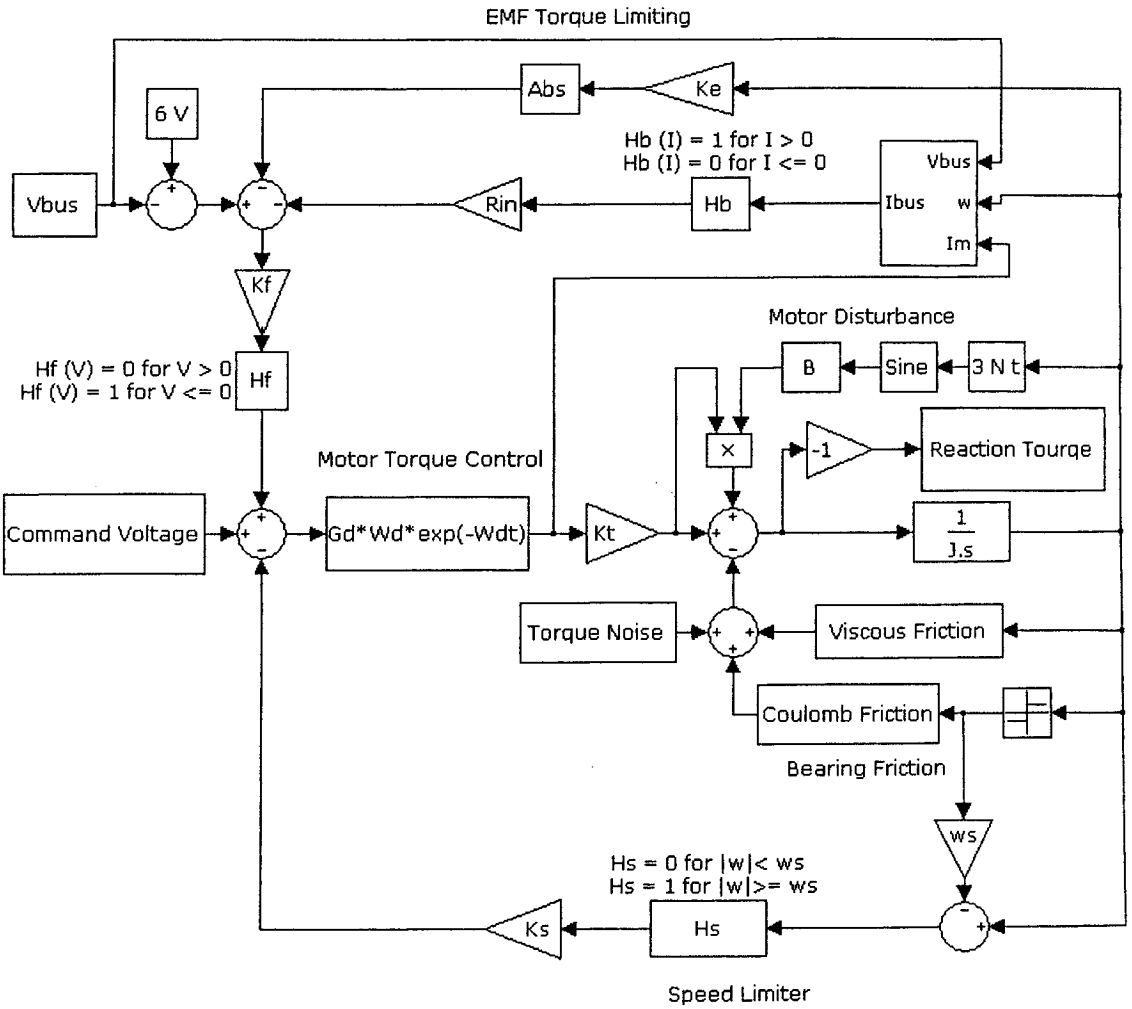


Fig. 3.4 Detailed Reaction Wheel (RW) block diagram [54]

The schematic diagram of the RW, as shown in Fig. 3.4, allows us to obtain a detailed relationship in the ACS of the satellite to be used for a high fidelity mathematical model. Typical RW parameters for the ITHACO's standard (type A) are provided in details in Table 3.2 [54].

The developed model is subsequently used for fault diagnosis in the ACS system including the nonlinear attitude dynamics model of the satellite.

The RW model modified in order to include fault injection capabilities that will be presented and investigated in the next chapter. Further details of each block and its function are provided below [54].

Motor Torque Control

The motor control torque block consists of a voltage controlled current source with gain G_d and a motor with torque constant K_t (as shown in Fig. 3.4). The function of this block is to generate a motor current that is proportional to the torque command voltage and to convert this current into torque through the motor torque constant K_t according to equation (3.6),

$$K_t = \frac{\tau_{motor}}{I_{motor}} \tag{3.6}$$

Speed Limiter

The function of the speed limiter block (as shown in Fig. 3.4) is to prevent the reaction wheel from any operation under unsaved speed by sensing the wheel speed using analog tachometer circuit speed and supply it into torque

command voltage through an appropriate negative feedback whenever the wheel speed exceeds a threshold value of the speed ω_s.

EMF Torque Limiting

At low bus voltage conditions, if the wheel runs at high speed, motor torque may be limited due to the increasing back-EMF, K_e, of the motor. From a disturbance standpoint, it should also be noted that the available motor torque will at that point be coupled directly to the bus voltage, and any fluctuations in bus voltage will be sensed as torque disturbance. It's clear from equation that the bus current is dependant on motor current, wheel speed, and bus voltage,

$$I_{bus} = \frac{I_m{}^2 R_B + 0.04|I_m|V_{bus} + P_q + \omega I_m k_e}{V_{bus} - 1} \tag{3.7}$$

When there is no power drawn from the bus, the voltage drop in the filter input resistance which caused by the bus current can be eliminated by the step function, $H_{b,}$ which included inside the EMF block diagram as shown in Fig. 3.4.

Bearing Friction

There are two components of the friction in reaction wheel:

- Viscous friction(τ_v) which is generated in the bearings due to the bearing lubricant and it has strong dependency on temperature also it varies

with the wheel speed and it can be calculated using the following formula:

$$\tau_v = \left(0.049 - \frac{0.0002}{°C}(T + 30°C)\right) \frac{\text{mN.m}}{\text{rad/sec}} \qquad (3.8)$$

- Coulomb friction (τ_c), is constant with polarity dependence on the direction of rotation of the wheel. It is caused by rolling friction within the bearings, and it's independent of wheel speed and temperature.

Torque Noise

Torque noise is a very low frequency torque variation from the bearings due to lubricant dynamics. It is a function of lubricant behavior and it has the most significant effect on satellite pointing. It can be specified as a deviation from the ideal location of the rotor at any constant speed. This noise can be modeled mathematically as a sine wave at the high pass filter frequency by the following Equation:

$$\tau_a = J\theta_a\omega_a{}^2 \sin\omega_a t \qquad (3.9)$$

The equation's constants are specified in Table 3.2.

Motor Disturbances

In reaction wheels, the motor torque can be a source of very high frequency disturbances due to the motor excitation and the magnetic construction. Two kinds of motor disturbances are introduced when a brushless DC motor is

employed in the reaction wheel; which includes torque ripple at the commutation frequency and a cogging at frequency corresponding to the number of poles and rate of rotation. Torque ripple can be defined as the amount of variation in the motor torque due to the commutation method and the shape of the back-EMF, the amount of the spacecraft disturbances due to this kind of torque is highly dependant on torque ripple frequency. The following equation has been implanted inside the block of the motor disturbances

$$\tau_{ripple} = B\sin(3N\omega t) \tag{3.10}$$

While cogging torque disturbance is completely eliminated due to the design of the ironless armature motor of ITHACO's (type A) reaction wheel. Further details about the RW model can be found in [54].

3.5.3 ACS Controller

A more commonly used control law is a position-plus-derivative controller, described by:

$$T_c = -K_p\theta - K_d\dot{\theta} \tag{3.11}$$

where T_c is the control torque, θ is the error signal (difference between actual and desired attitude), $\dot{\theta}$ is the time derivative of the error signal, and K_p, K_d are the proportional and derivative gains respectively. The derivative gain of the controller is necessary as the space environment has no physical damping which provides damping and reduces the angular excursion.

The gains of the PD controller are selected based on satisfying the ACS performance requirements such as a steady-state pointing error of 0.2° with settling time in less than a minute.

3.5.4 Developed 3-Axes ACS Model

In this section, a three-axis stabilized satellite control technique based on the reaction wheels and with *No Momentum Bias* will be considered. Three separate PD controllers will be employed to command the 3-RWs for control of each satellite axis. In order to make the developed three axis ACS model as realistic as possible, nonlinear and coupled ACS equations of motion are considered in addition to the nonlinear reaction wheel model which is presented in [50].

Firstly, we will start with Euler's equations (3.12), (3.13), and (3.14) which describe the general attitude motion of a vehicle body frame aligned with the principal axes, in which case the products of inertia are zero (the corresponded inertia matrix is given in equation (3.18)), about its center of mass [50],

$$\tau_{dx} + \tau_x = \dot{\omega}_x I_{xx} + \omega_y \omega_z (I_{zz} - I_{yy}) \tag{3.12}$$

$$\tau_{dy} + \tau_y = \dot{\omega}_y I_{yy} + \omega_x \omega_z (I_{xx} - I_{zz}) \tag{3.13}$$

$$\tau_{dz} + \tau_z = \dot{\omega}_z I_{zz} + \omega_x \omega_y (I_{yy} - I_{xx}) \tag{3.14}$$

From these equations it may be seen that the cross coupling will potentially be present, where $(\tau_{dx},\tau_{dy},\tau_{dz})$ represent the disturbance torques which act on the satellite about *roll*, *pitch* and *yaw* axis respectively, (τ_x,τ_y,τ_z) represent the developed torques due to the motion on each axis, (I_{xx},I_{yy},I_{zz}) represent the body moment of inertia of the satellite along x, y, and z axes respectively, and $(\omega_x,\omega_y,\omega_z)$ represent the angular velocity directed along x, y, and z axes respectively. We may rewrite equations (3.12), (3.13), and (3.14) as following:

$$\dot{\omega}_x = \frac{1}{I_{xx}}\left(\tau_{dx} + \tau_x - \omega_y\omega_z(I_{zz} - I_{yy})\right) \tag{3.15}$$

$$\dot{\omega}_y = \frac{1}{I_{yy}}\left(\tau_{dy} + \tau_y - \omega_x\omega_z(I_{xx} - I_{zz})\right) \tag{3.16}$$

$$\dot{\omega}_z = \frac{1}{I_{zz}}\left(\tau_{dz} + \tau_z - \omega_x\omega_y(I_{yy} - I_{xx})\right) \tag{3.17}$$

Based on equations (3.15), (3.16), and (3.17) we developed a 3-axes ACS model using Matlab-SIMULINK environment as shown in Fig. 3.5. Note that, in our developed ACS model, the values of the moment of inertia of the satellite body along x, y, and z axes are given in the decoupled satellite inertia matrix (I_v) as stated in equation (3.18)

$$I_v = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} = \begin{bmatrix} 22 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 36 \end{bmatrix} \tag{3.18}$$

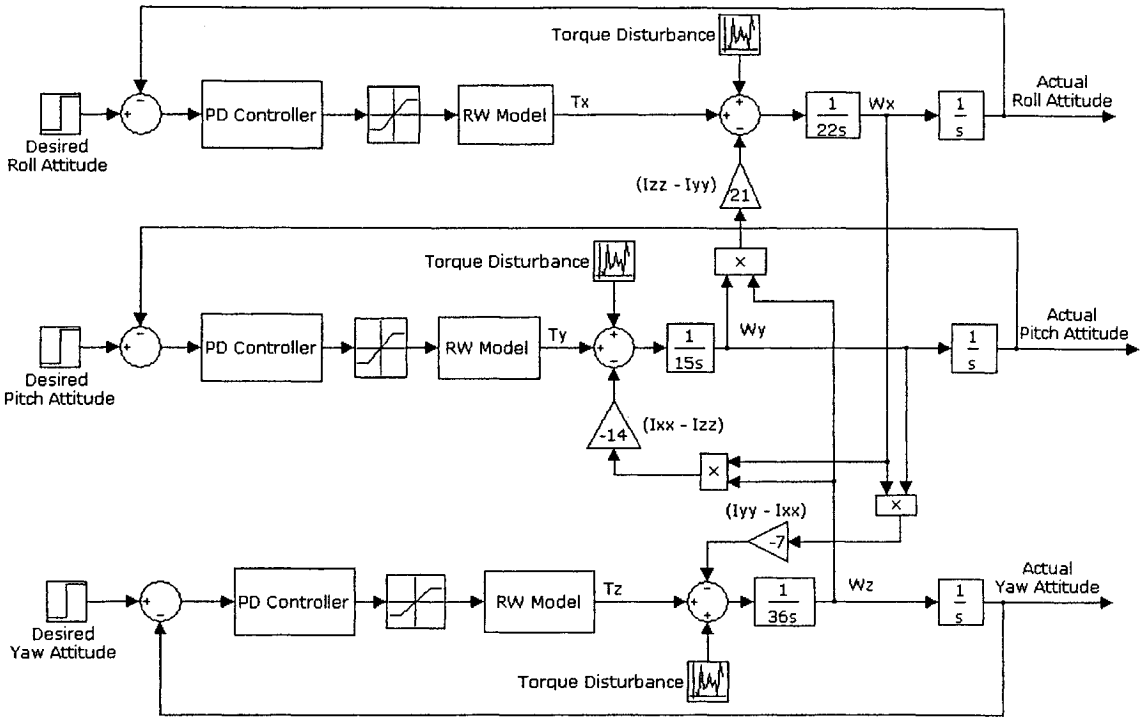


Fig. 3.5 Developed 3-axes ACS model using Matlab-SIMULINK blocks

3.6 Conclusions

In this chapter, the Attitude Control Subsystem (ACS) of a satellite has been reviewed along with a detailed description of the three-axis ACS model that has been developed in this thesis under Matlab-SIMULINK environment. Different ACS actuators and techniques have been presented in details. Afterwards, in Chapter 4, fault detection and isolation algorithm will be applied and the corresponding simulation results will be presented under different actuator faults scenarios.

Chapter 4

Detection and Isolation of Actuator Faults for the ACS of a Satellite

As an extension to our published works in [44] that have been done on a single axis ACS model; in this chapter we will apply and investigate our developed neural FDI algorithm on a three-axis ACS model. As mentioned earlier in Chapter 2, the neural FDI scheme can be constructed using two types of neural networks in order to detect the simulated actuator faults in the ACS and provide detailed information about the time and location of each corresponded fault. The components of the developed neural FDI scheme are depicted in Fig. 4.1, which designed specifically for the purpose of detecting and isolation various reaction wheel faults in the attitude control subsystem of a satellite. The structure of our developed neural FDI scheme can be explained as following:

- Neural Observer Structure: this observer is constructed based on the dynamic neural network because this kind of neural network will deal directly with the behavior of the dynamic system (reaction wheel) to identify all known modes of operations under healthy and normal conditions. The purpose of the neural observer is to provide an error signal that can distinguish between the healthy and faulty operation conditions.
- Neural Classifier Structure: the task of this neural structure can be fulfilled based on the static neural network. Dynamic properties are not necessary to perform the task of residual evaluation. The output of the neural residual observer is analyzed by the classifier in order to provide enough details about the occurrence moment of each simulated fault and its location.

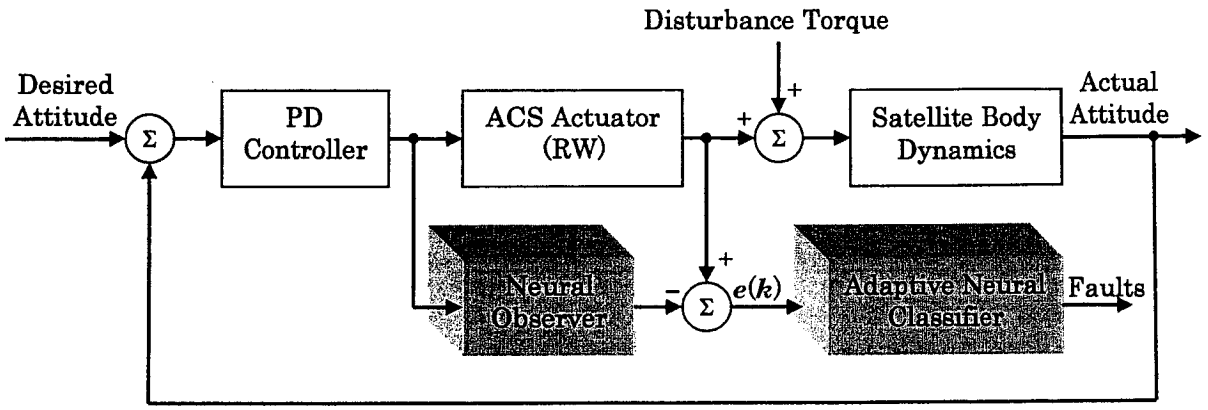


Fig. 4.1 Structure of neural FDI scheme for a satellite's single axis ACS

The implementation of the neural FDI scheme consists of three stages:

- System Identification

- Fault Detection
- Fault Isolation

4.1 Model-based System Identification

In order to justify the capability of our proposed neural approach, we developed and implemented a model-based linear observer (LO), which we treat as a benchmark fault detection strategy.

The development of the fault detection benchmark for the ACS actuator has been done based on the linear model of the reaction wheel that is presented in [53].

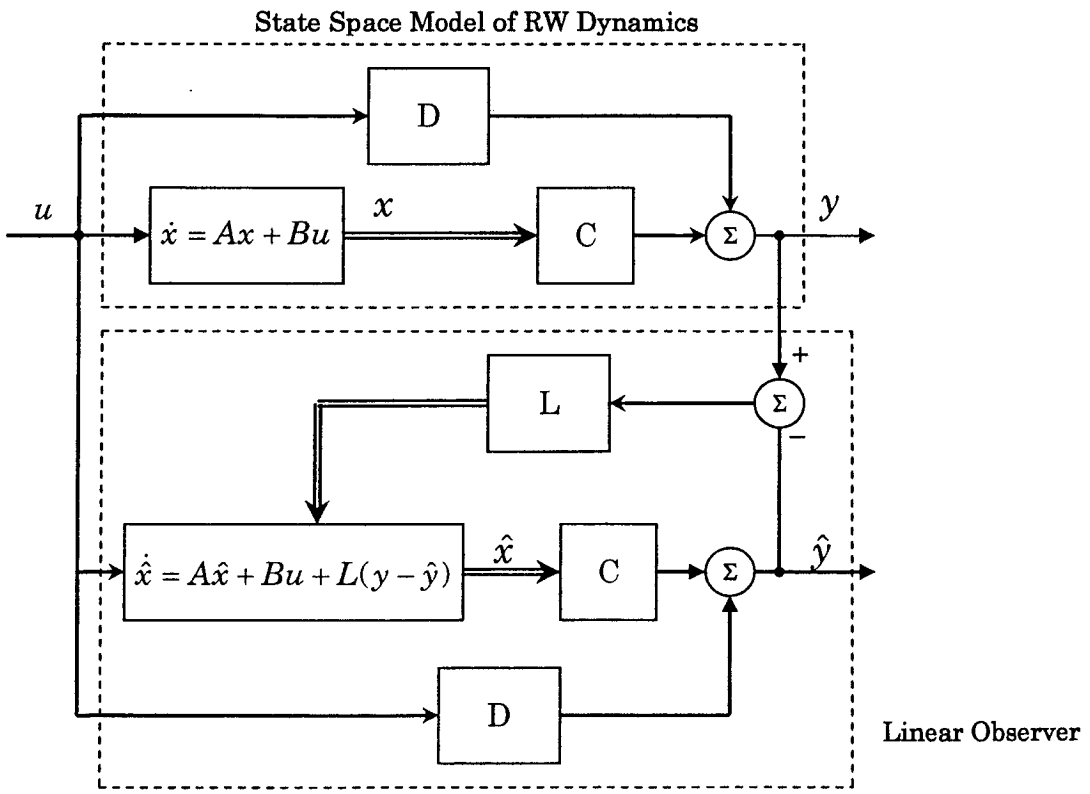


Fig. 4.2 Schematic diagram of the RW and linear observer (designing phase)

Basically, as shown in Fig. 4.2, the observer acts as a reaction wheel; it has the same dynamics of the reaction wheel that described by differential equation. Also the input of the RW (u) which is the torque command voltage represents the same input of the linear observer. In order to make the estimated states ($\hat{x}$) approach the values of the actual states (x) of the linear observer, an additional term (L) is employed to compare the actual measured output of the RW (y), which is the reaction torque, to the estimated output ($\hat{y}$).

The state space representation of the dynamics of the reaction wheel and the linear observer are given by equations 4.1 and 4.2 respectively.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{4.1}$$

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + L(y - \hat{y}) \\ \hat{y} &= C\hat{x} + Du \end{aligned} \tag{4.2}$$

where, A represents the system matrix, B the input vector, C the output vector and D represents the input-output ratio vector, also L represents the observer gain vector. Note that our system is Single-Input, Single-Output system (SISO).

From the transfer function of the RW linear model as given in Equation 3.1 in Chapter 3, we can find the values of each matrix as following:

$$A = \left[-\frac{G_d K_t K_e + \tau_v}{J} \right]$$

$$B = \left[-\left(\frac{G_d K_t K_e + \tau_v}{J} \right) (G_d K_t) \right]$$

$$C = [1]$$

$$D = [G_d K_t]$$

The key part in designing phase of linear observer is choosing the observer gain vector (L) to make the dynamics of the estimator to be much faster than the open loop dynamics of the RW itself. Using Ackermann's formula, we placed the poles of the observer four times farther to the left than the dominant poles of the RW system. This has been done using Matlab function *acker*.

4.2 Neural-based System Identification

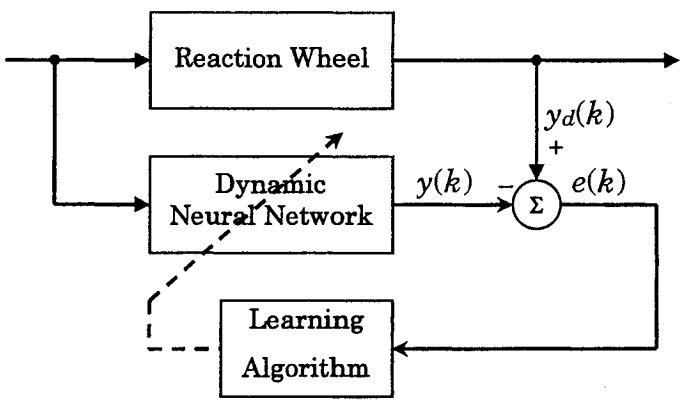


Fig. 4.3 Identification scheme for the nonlinear RW model using DNN

System identification task represents the most important part in the FDI algorithm. The identification configuration is shown in Fig. 4.3. It is clear from this configuration that the DNN needs just one input in order to

construct a suitable identification model and produce an output $y(k)$. During this task, the network of dynamic neurons is applied to modeling the normal behavior of the ACS actuator for each satellite axis (*roll*, *pitch* and *yaw* separately), two subtasks have to be done also, which include the training and the testing of the proposed DNN approach.

4.2.1 Training Phase

The modeled ACS actuator, RW, along each satellite axis has one input (torque command voltage) and one output (reaction torque) and it's simulated in order to generate input-output training data pairs to be used for training purposes. The training process for the dynamic networks was carried out using an extended dynamic backpropagation algorithm for about 10,000 time samples (msec) for each axis. Preprocessing steps are performed for the networks inputs and targets so that all the input-output data vectors are normalized in the range [-1, 1].

The learning algorithm is initialized with small random values for the network parameters except for the IIR filter's denominator coefficients, they are initialized to zeros. The neurons in the dynamic network structure all have embedded second order IIR filters with one hidden layer of *hyperbolic tangent* activation functions and *linear* activation functions for the output layer. The learning rate parameter was set to 0.05. Furthermore, starting from a relatively small structure we may develop an optimal architecture for

the proposed dynamic network by incrementally increasing the number of hidden neurons until desired performance specifications are obtained.

The characteristics of the learning phase are summarized in Table 4.1. For Roll and Pitch axes, the best results were obtained with the network structure $\mathbf{N_{1\text{-}6\text{-}1}}$ which means; one input, 6 neurons in the hidden layer, and one neuron in the output layer, while the best results with the network structure $\mathbf{N_{1\text{-}9\text{-}1}}$ were obtained for Yaw axis. The performance of each network during the training phase for each satellite axis is depicted in Fig. 4.4, indicating that the networks are trained quite well.

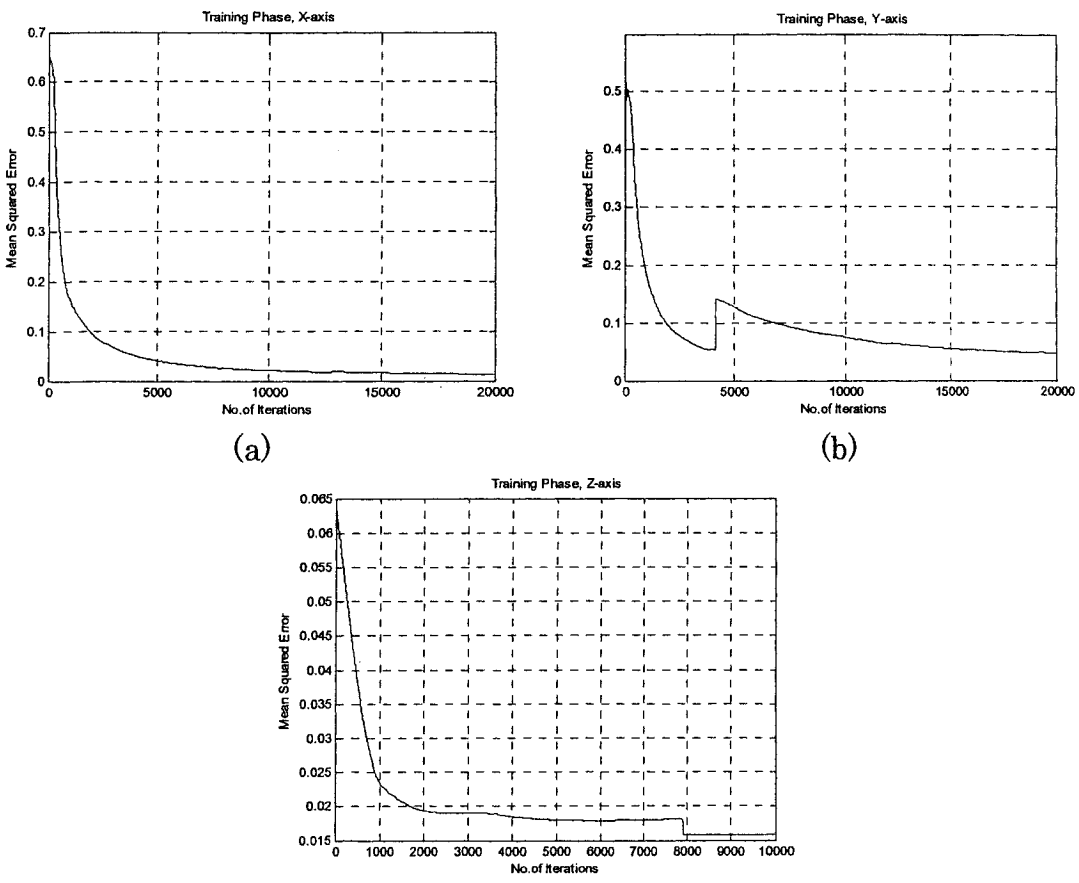


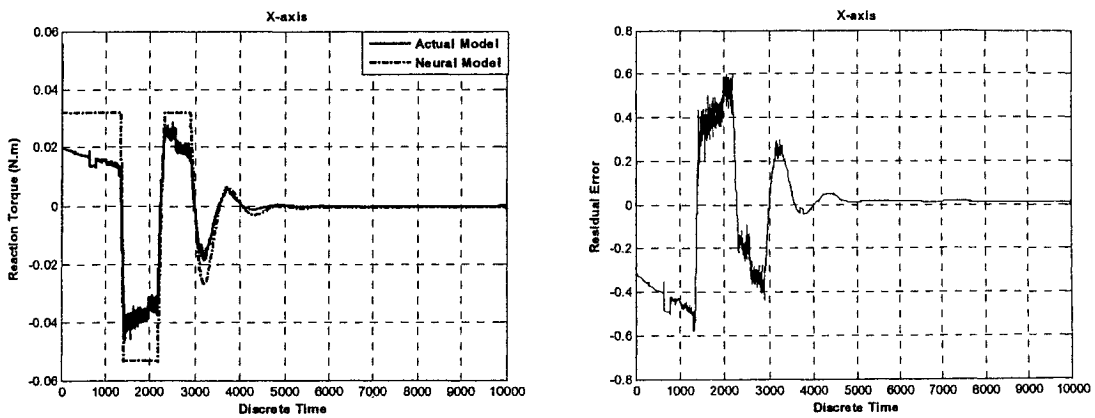
Fig. 4.4 Learning curves for the Dynamic Neural Network; (a) Roll axis (b) Pitch axis, and (c) Yaw axis

TABLE 4.1
CHARACTERISTICS OF THE LEARNING PHASE

Satellite Axis	Network Size	Number of Iterations	Performance Index	Filter Order
X-axis	1-6-1	20,000	0.0139	2nd Order
Y-axis	1-6-1	20,000	0.0467	2nd Order
Z-axis	1-9-1	12,000	0.0167	2nd Order

4.2.2 Testing Phase

The representation capabilities of the trained networks are evaluated through generalizing them with other data sets that were not seen previously by the each network.

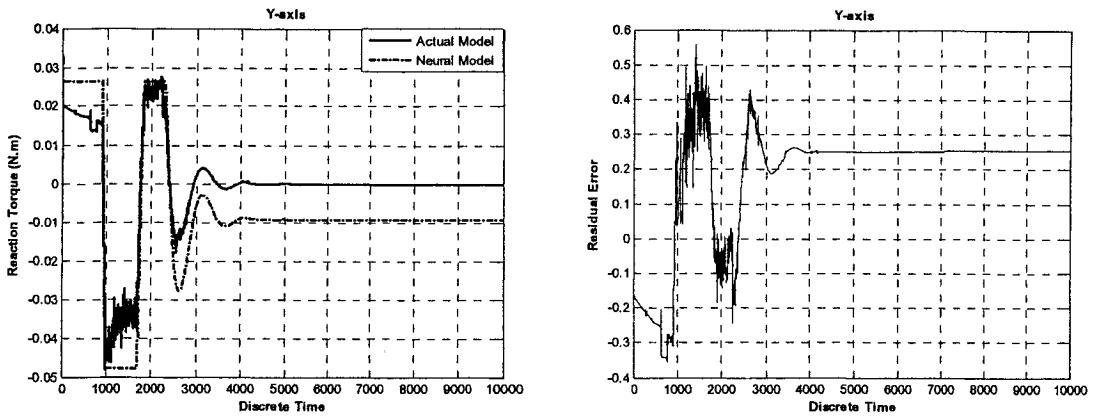


(a) (b)

Fig. 4.5 Testing phase of the DNN for X-axis: (a) Output of the actual and neural model, (b) Generated residual error

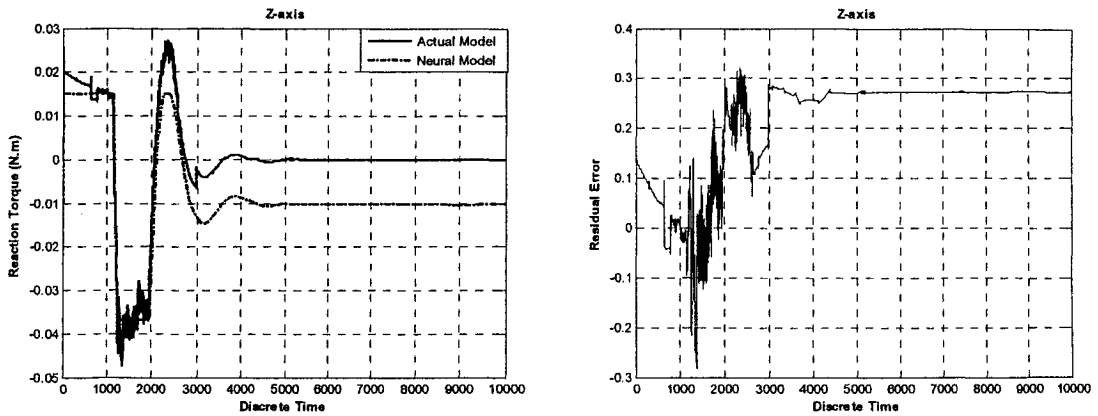
Figs. 4.5, 4.6, and 4.7 show the testing phase of the dynamic network estimator for each corresponding axis. As seen in Fig. 4.5, the output of the neural network follows the actual output of the model perfectly, and it's

confirmed by the corresponding error signal, which indicates that the neural residual generator has strong capability in detecting fluctuations that could happen in the input signal of the ACS actuator, i.e. torque command voltage.



(a) (b)

Fig. 4.6 Testing phase of the DNN for Y-axis: (a) output of the actual and neural model, (b) generated residual error



(a) (b)

Fig. 4.7 Testing phase of the DNN for Z-axis: (a) output of the actual and neural model, (b) generated residual error

Also as seen in Figs. 4.6 and 4.7, the output of the residual generator follows the actual output of the actuator quite well, which indicates that the neural models are capable of detecting changes in the reaction wheel input signal.

Note that the difference between the response of the actual and the neural model in each axis is due to the error between the actual and the estimated reaction torque signal when the noise is present.

A quantitative summary for the Figs. 4.5, 4.6, and 4.7 of testing phase that has been done on a specific attitude reference setting are given in Table 4.2.

TABLE 4.2
QUANTITATIVE SUMMARY OF TESTING PHASE FIGURES

Desired Attitude	Testing Samples	Steady State Time (msec)	Steady State Residual Error
$\Phi_{ref} = 18°$ (Roll angle)	10,000	4,500	0.0075
$\theta_{ref} = 15°$ (Pitch angle)	10,000	4,000	0.249
$\Psi_{ref} = 10°$ (Yaw angle)	10,000	4,000	0.275

4.3 Actuator Faults Detection

The constructed residual generators are applied for the purpose of fault detection in the satellite's reaction wheels. Different types of fault scenarios under worst case noisy working conditions are considered and have been injected to the closed loop attitude controlled system in each satellite's axis. The process for making fault detection decisions can be accomplished by using a simple threshold technique for each axis as shown in Table 4.3; so that any deviation from this range will be considered as a fault. This threshold was selected after performing a number of simulations under different operating settings to guarantee that our proposed approach will work successfully with minimal false alarms. Table 4.4 provides detailed information about different attitude settings and their residual error signals.

TABLE 4.3
THRESHOLD RANGES FOR EACH SATELLITE AXIS

Satellite Axis	Threshold
Roll axis	0.005 – 0.010
Pitch axis	0.246 – 0.252
Yaw axis	0.274 – 0.276

TABLE 4.4
SIMULATION RESULTS OF DIFFERENT ATTITUDE SETTINGS

Simulation No.	Desired Attitude	Residual Error
1	$\Phi_{ref} = 9°$	0.070
	$\theta_{ref} = 12°$	0.240
	$\Psi_{ref} = 5°$	0.250
2	$\Phi_{ref} = 90°$	0.070
	$\theta_{ref} = 12°$	0.240
	$\Psi_{ref} = 5°$	0.270
3	$\Phi_{ref} = 9°$	0.080
	$\theta_{ref} = 90°$	0.270
	$\Psi_{ref} = 5°$	0.270
4	$\Phi_{ref} = 9°$	0.080
	$\theta_{ref} = 12°$	0.250
	$\Psi_{ref} = 90°$	0.280
5	$\Phi_{ref} = 60°$	0.040
	$\theta_{ref} = 30°$	0.140
	$\Psi_{ref} = 45°$	0.270
6	$\Phi_{ref} = 90°$	– 0.030
	$\theta_{ref} = 75°$	0.170
	$\Psi_{ref} = 85°$	0.120
7	$\Phi_{ref} = 18°$	0.0075
	$\theta_{ref} = 15°$	0.249
	$\Psi_{ref} = 10°$	0.275
8	$\Phi_{ref} = 30°$	0.020
	$\theta_{ref} = 18°$	0.250
	$\Psi_{ref} = 25°$	0.230

4.3.1 Bus Voltage (V_{Bus}) Fault Scenario

For low bus voltage conditions, it is important to note that the motor torque may be limited at high speeds due to the increasing back-EMF, K_e, of the motor. From a disturbance standpoint, it should also be noted that the available motor torque will at that point be coupled directly to the bus voltage, and any fluctuations in bus voltage will be sensed as torque disturbance [54].

First, a low bus voltage (50% drop of nominal value) faulty scenario was injected at the time 9,000 msec for Roll axis, while the injection time was 7,000 msec for Pitch axes, and 8,000 msec for Yaw axis as shown in Figs. 4.8, 4.9, and 4.10 respectively. Note that the fault diagnosis is principally performed during the steady state response of the satellite as any large transitory variations may be mistakenly be treated and flagged as faults. As shown in Fig. 4.8, e.g. the steady state response of the residual generator for the RW is reached after approximately the 4,000 msec in Roll axis.

The results in Figs. 4.7(a), 8(a), and 9(a) clearly show that the proposed dynamic neural network residual generator is successfully capable of recognizing and determining the presence of a fault that is very close to the time that the actual fault has occurred in the corresponding axis while other axes work normally and they generated normal residual error which emphasis that each network is trained very well and it dose generate a normal residual error when there is no occurred faults inside its monitored system.

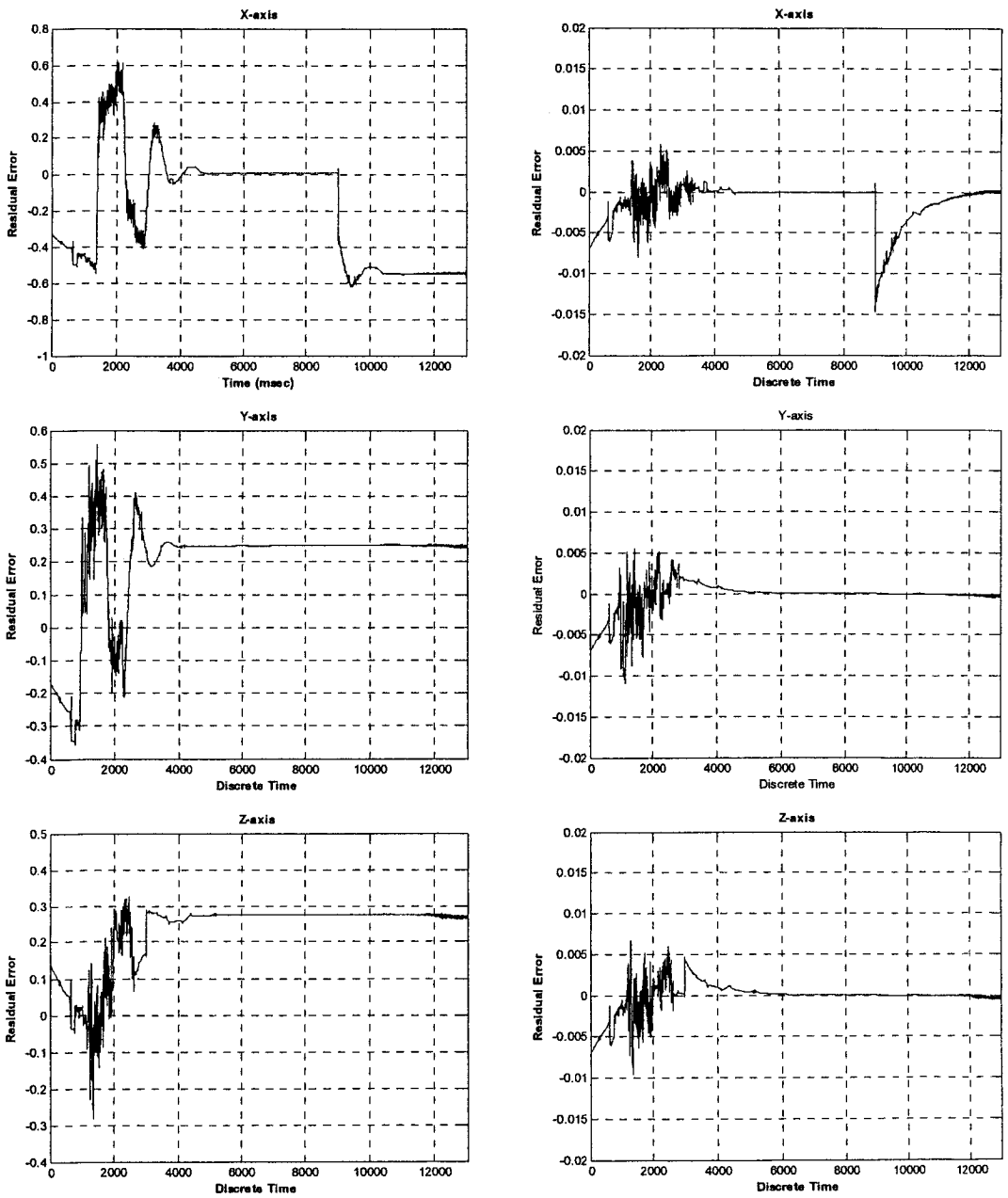
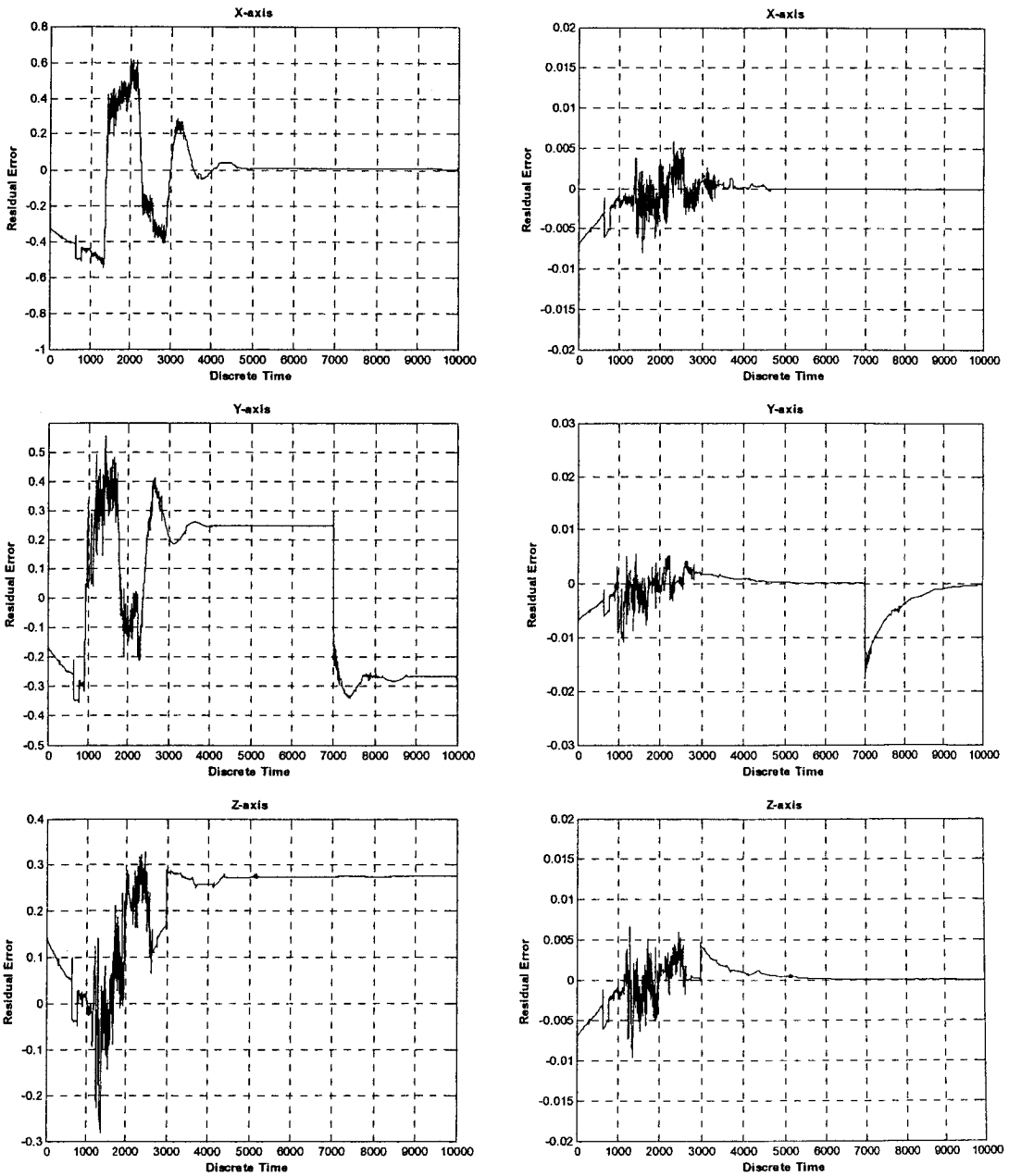


Fig. 4.8 Residual error signals in case of V_{Bus} fault in X-axis:
(a) Neural observer output (b) Linear observer output

Next, linear model-based observer was implemented in each axis in order to compare the results of the proposed neural observers. The output of the linear observers is shown in Figs. 4.8(b), 4.9(b), and 4.10(b), which shows that

the designed observer is not capable of unambiguously detect the presence of the fault. It is clearly seen that the residual generated by the model-based approach has converged back to its normal condition while the fault has persisted in the RW. Table 4.5 summarizes this fault scenario quantitatively.



(a) (b)

Fig. 4.9 Residual error signals in case of V_{Bus} fault in Y-axis: (a) Neural observer output (b) Linear observer output

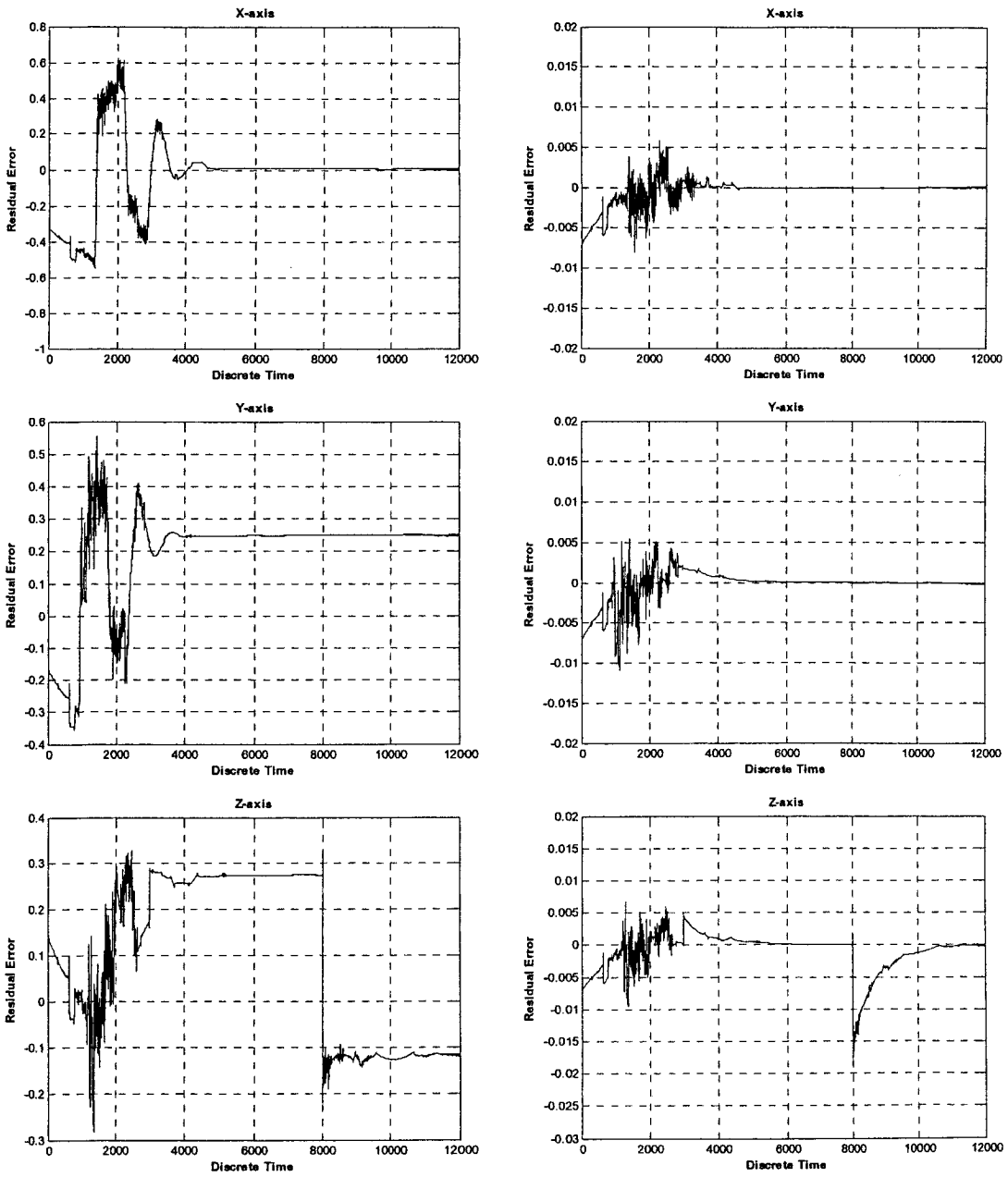


Fig. 4.10 Residual error signals in case of V_{Bus} fault in Z-axis:
(a) Neural observer output (b) Linear observer output

It is clear in Table 4.5 that the detection speed of the neural observer is very high and there is no delay time in fault detection, also the same thing in the

linear observer but its disadvantage that the steady state errors are zero before and after faults under the existence of the fault.

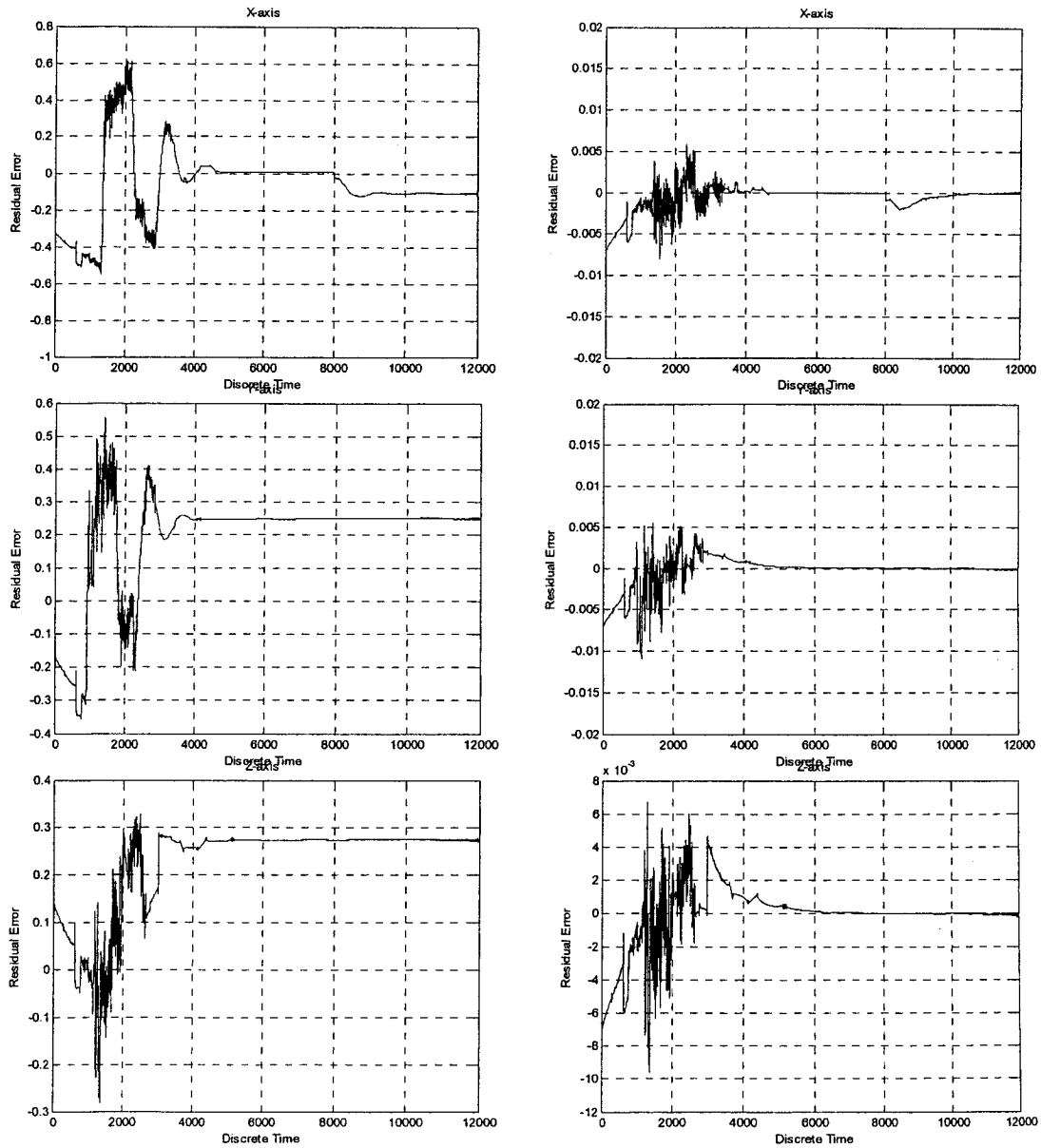
TABLE 4.5
QUANTITATIVE SUMMARY OF V_{Bus} FAULT SCENARIO FIGURES

50% drop of nominal V_{Bus} value in each axis									
Faulty Axis		Injection Time (msec)		Detection Time (msec)		Steady State Residual Error			
		DNN	LO	DNN	LO	DNN		LO	
						Normal	Faulty	Normal	Faulty
X	X	9,000	9,000	9,000	9,000	**0.0075**	**– 0.550**	0.0	0.0
	Y					0.249	0.250	0.0	0.0
	Z					0.275	0.275	0.0	0.0
Y	X					0.0075	0.007	0.0	0.0
	Y	7,000	7,000	7,000	7,000	**0.249**	**– 0.265**	0.0	0.0
	Z					0.275	0.275	0.0	0.0
Z	X					0.0075	0.007	0.0	0.0
	Y					0.249	0.250	0.0	0.0
	Z	8,000	8,000	8,000	8,000	**0.275**	**– 0.120**	0.0	0.0

4.3.2 Motor Current (I_m) Fault Scenario

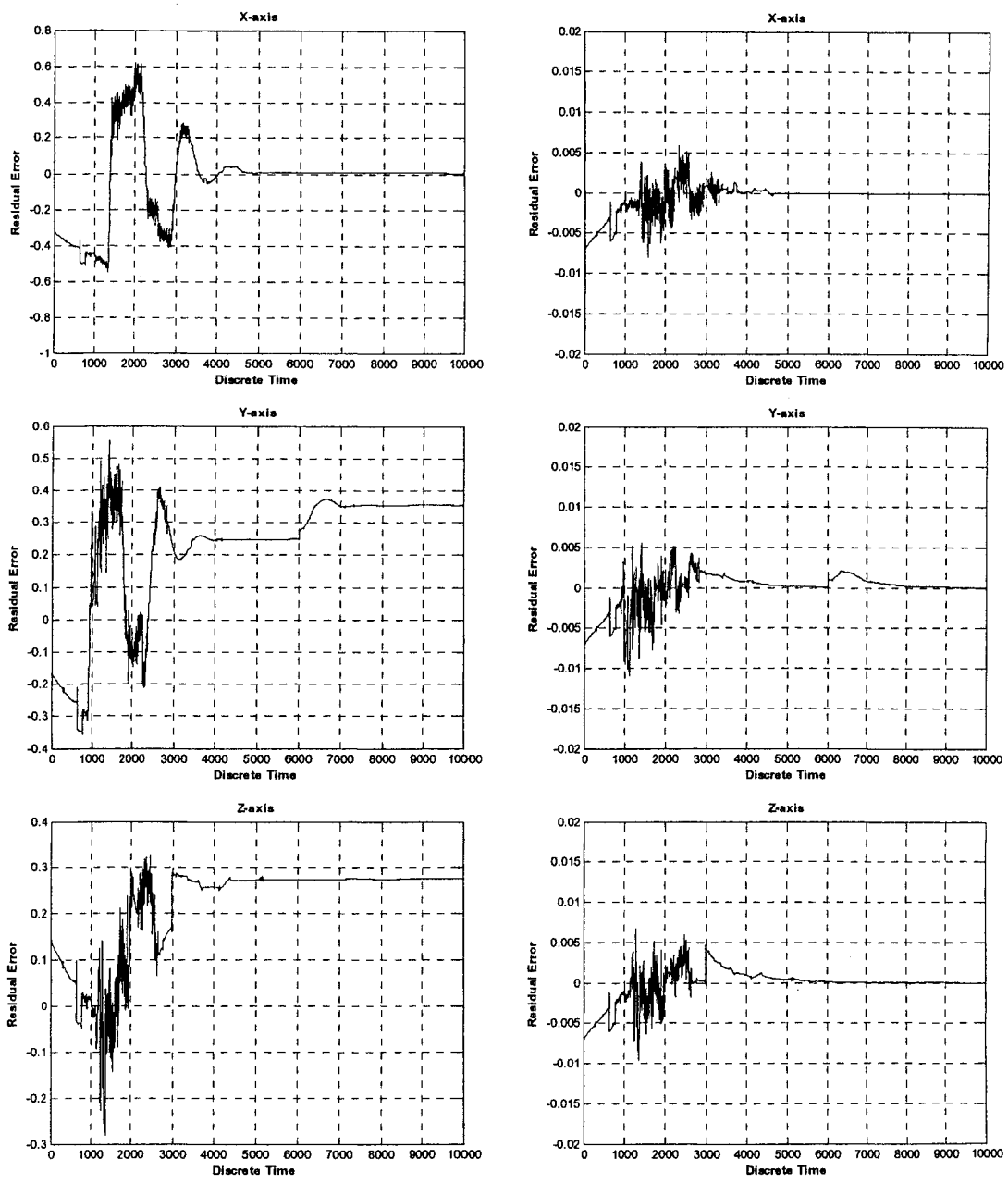
As mentioned in Chapter 3, the generated motor current from motor control torque block is proportional to the torque command voltage and the current is converted into torque through the motor torque constant K_t. Therefore, any injected fault in the motor driver gain will be reflected directly as fluctuations in the motor current and as result in the motor torque. A faulty scenario is presented to the motor driver so that a low current (50% of the normal peak

value for X & Y axes and 70% for Z-axis) faulty situation is obtained and as a result the supplied torque to satellite axes from the RW is decreased by the same percentage. The capabilities of our proposed dynamic network residual generator and a model-based linear observer in detecting this kind of fault are depicted in Figs. 4.11, 4.12, and 4.13 respectively.



(a) (b)

Fig. 4.11 Residual error signals in case of I_m fault in X-axis:
(a) Neural observer output (b) Linear observer output



(a) (b)

Fig. 4.12 Residual error signals in case of I_m fault in Y-axis:
(a) Neural observer output (b) Linear observer output

As can be observed from these figures; that the neural observer is able to clearly detect the injected fault directly without delays, while the linear residual generator has failed completely in detecting the severe motor

current faults. A normal residual error has been generated from other healthy axes neural observers (X and Z). This confirms that just the faulty axis indicates a fault and no false alarms are present in the nominal axes.

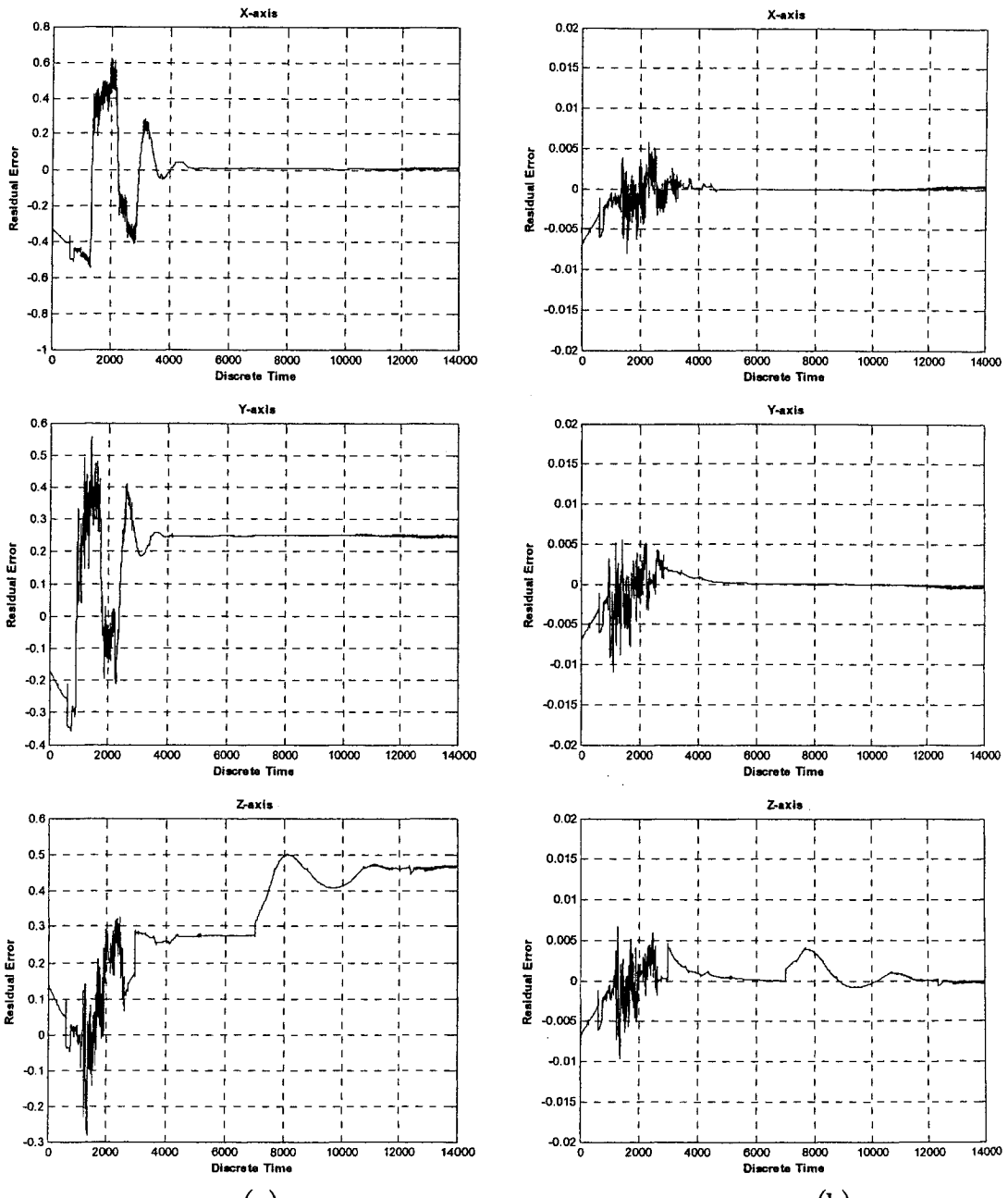


Fig. 4.13 Residual error signals in case of I_m fault in Z-axis:

(a) Neural observer output (b) Linear observer output

A quantitative summary for motor current fault is provided in Table 4.6. This table confirms the capabilities of the neural estimator numerically in detecting sever motor current faults in all satellite axes, while the linear observer has failed completely in performing the detection task.

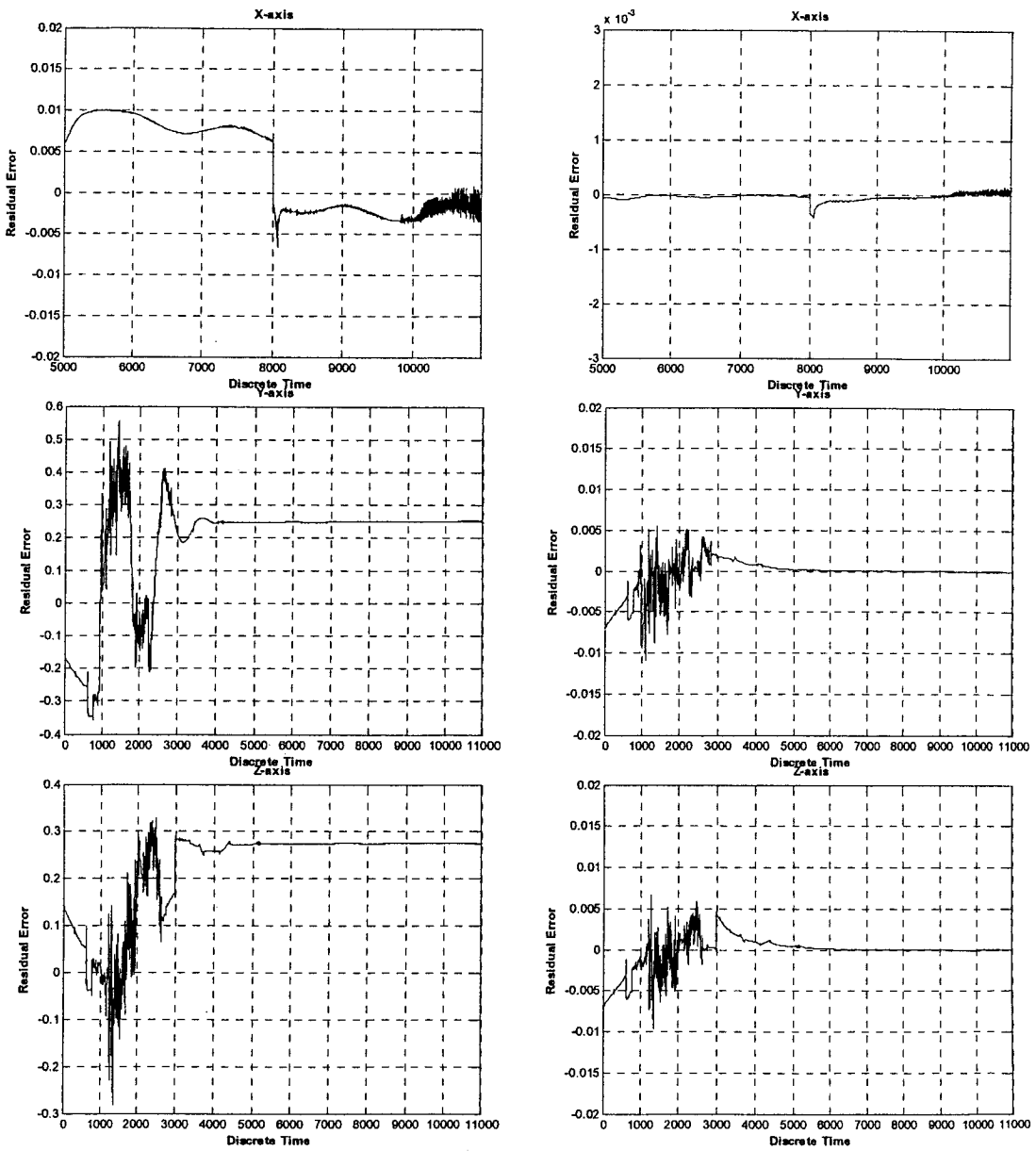
TABLE 4.6
QUANTITATIVE SUMMARY OF I_M FAULT SCENARIO FIGURES

50% drop of nominal peak I_m value in X & Y axes and 70% in Z-axis									
Faulty Axis		Injection Time (msec)		Detection Time (msec)		Steady State Residual Error			
						DNN		LO	
		DNN	LO	DNN	LO	Normal	Faulty	Normal	Faulty
X	X	8,000	8,000	8,000	8,000	**0.0075**	**– 0.120**	0.0	0.0
	Y					0.249	0.250	0.0	0.0
	Z					0.275	0.275	0.0	0.0
Y	X					0.0075	0.005	0.0	0.0
	Y	6,000	6,000	6,000	6,000	**0.249**	**0.354**	0.0	0.0
	Z					0.275	0.275	0.0	0.0
Z	X					0.0075	0.007	0.0	0.0
	Y					0.249	0.250	0.0	0.0
	Z	7,000	7,000	7,000	7,000	**0.275**	**0.465**	0.0	0.0

4.3.3 Viscous Temperature (τ_v) Fault Scenario

Due to the unknown working environment of the RW in the satellite in outer space, the friction model is designed to work under a limited range of temperatures. Since the bearing viscosity is temperature dependent in the friction model of the reaction wheel, thus means any fluctuation in the

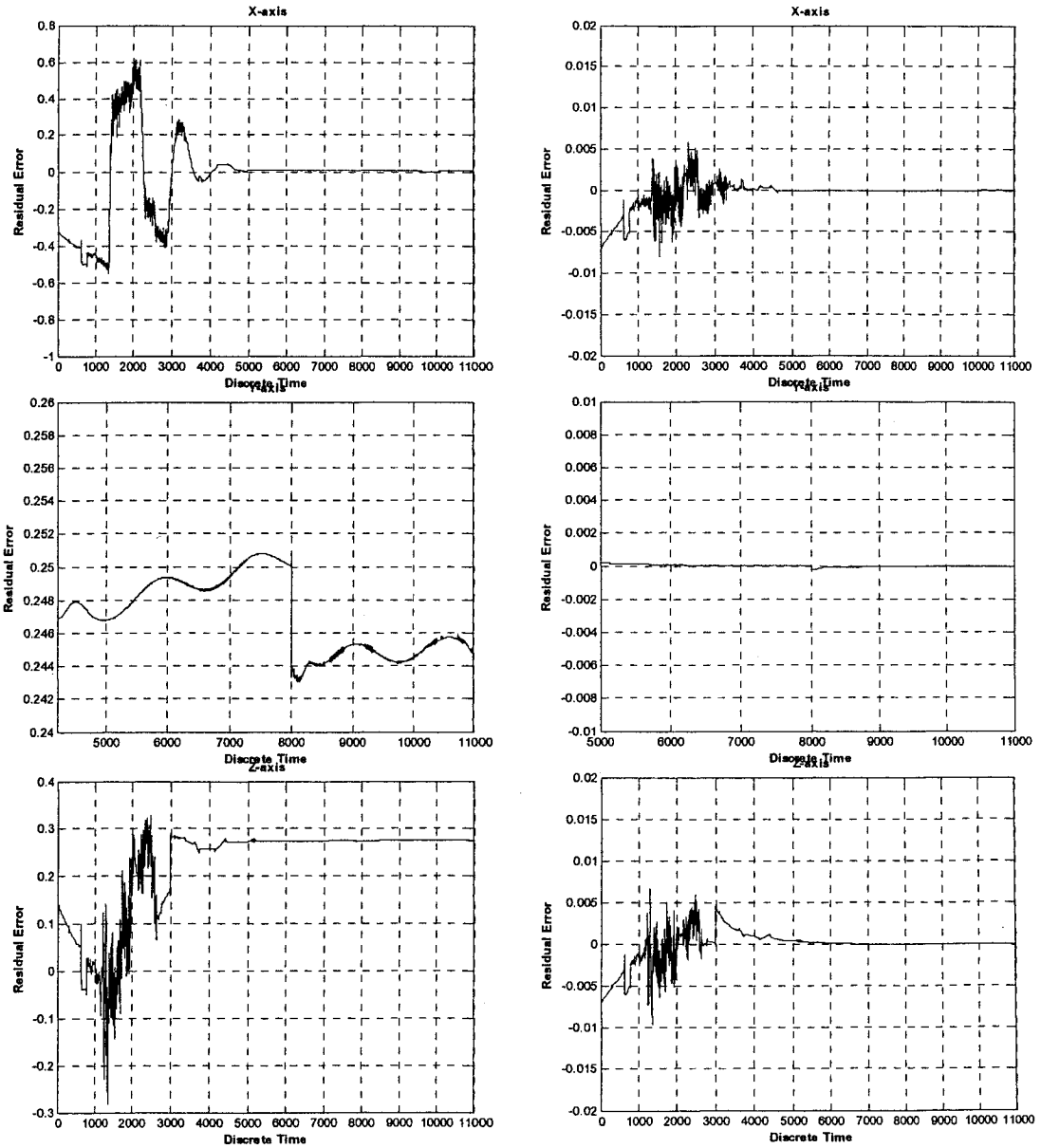
temperature will be reflected as fluctuations in the drag torque. A simulated temperature fault was considered which resulted in an increasing of 74.5% in the nominal produced viscous torque for X-axis, and a decreasing of 49.5% and 44.3% for Y-axis and Z-axis respectively.



(a) (b)

Fig. 4.14 Residual error signals in case of τ_v fault in X-axis:
(a) Neural observer output (b) Linear observer output

The dynamic neural network residual generator, as demonstrated in Figs. 4.14(a). 4.15(a), and 4.16(a), detects the injected temperature fault rapidly. This indicates that the predicted output of each dynamic neural network clearly deviates from the output of the actual system. Figs. 4.14(b), 4.15(b), and 4.16(b), show that the linear residual generator has yielded very poor performance in detecting the temperature faults.



(a) (b)

Fig. 4.15 Residual error signals in case of τ_v fault in Y-axis: (a) Neural observer output (b) Linear observer output

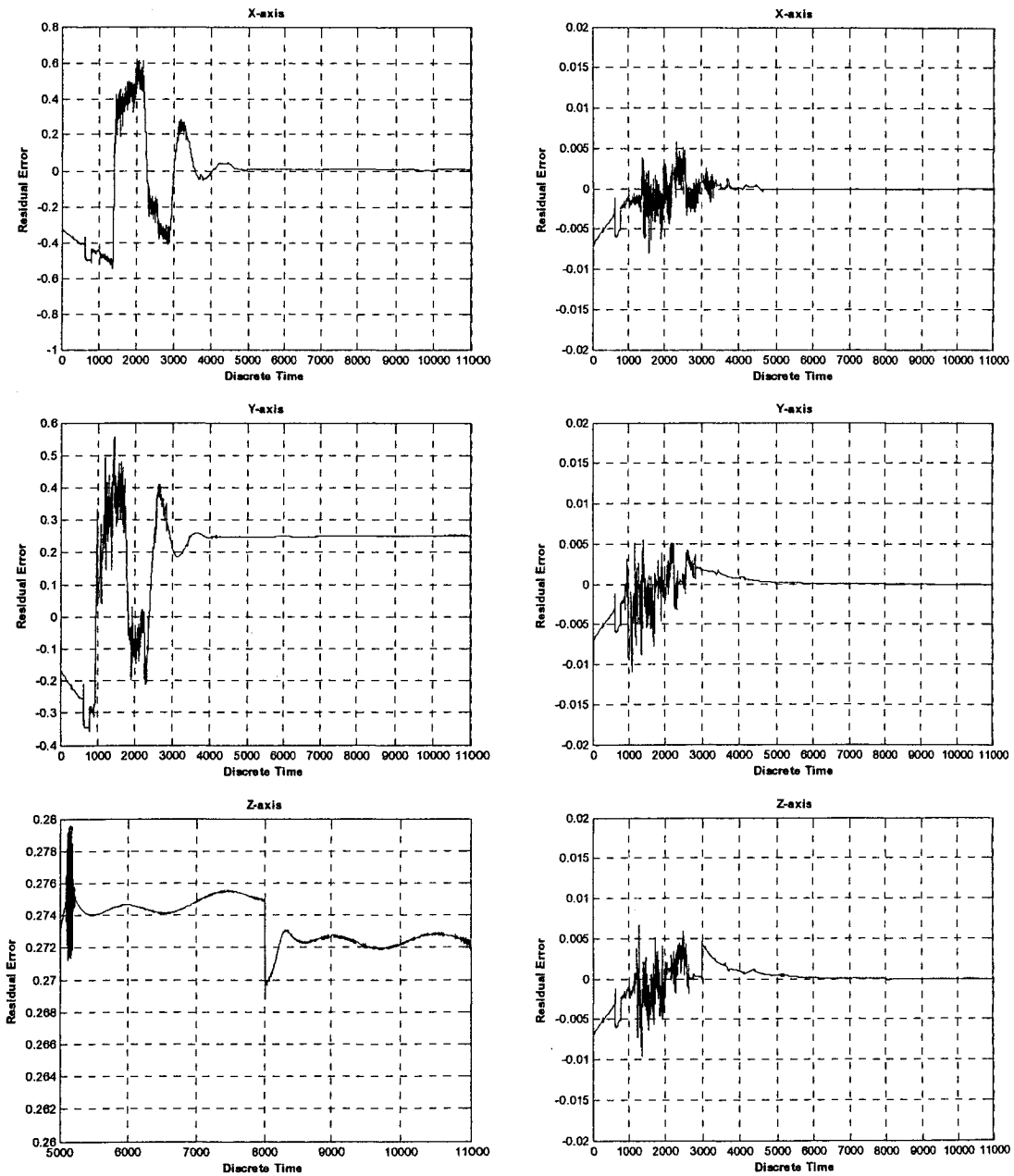


Fig. 4.16 Residual error signals in case of τ_v fault in Z-axis:
(a) Neural observer output (b) Linear observer output

Due to the fact that the temperature fault residual error are close to normal residual error so that there is a possibility of having some false alarms when the residual error deviates from the threshold setting.

Viscous temperature fault scenario is summarized quantitatively in Table 4.7. During this fault scenario the neural observers have shown that their capabilities in detecting even small various injected faults in all satellite axes that could not detected by model-based observers.

TABLE 4.7
QUANTITATIVE SUMMARY OF τ_v FAULT SCENARIO FIGURES

74.5% increasing in X-axis and a decreasing of 49.5% and 44.3% for Y-axis and Z-axis respectively.									
Faulty Axis		Injection Time (msec)		Detection Time (msec)		Steady State Residual Error			
		DNN	LO	DNN	LO	DNN		LO	
						Normal	Faulty	Normal	Faulty
X	X	8,000	8,000	8,000	8,000	**0.0075**	**−0.0075**	0.0	0.0
	Y					0.249	0.250	0.0	0.0
	Z					0.275	0.275	0.0	0.0
Y	X					0.0075	0.007	0.0	0.0
	Y	8,000	8,000	8,000	8,000	**0.249**	**0.245**	0.0	0.0
	Z					0.275	0.275	0.0	0.0
Z	X					0.0075	0.007	0.0	0.0
	Y					0.249	0.250	0.0	0.0
	Z	8,000	8,000	8,000	8,000	**0.275**	**0.272**	0.0	0.0

4.3.4 Double Sequential Faults Scenario

Due to the severity of double faults, in this section we considered a double faulty situation that injected in both the voltage bus and motor driver gain circuits sequentially. For X-axis, a low bus voltage (50% drop of nominal value) fault was injected at the time sample 7,000 msec, which is then

followed by a low motor current (50% drop of nominal peak value) fault injected at the time sample 10,000 msec as shown in Fig. 4.17. For Y & Z axes, a low I_m fault (70% and 60% drop of nominal peak value respectively) was injected firstly then followed by an over V_{Bus} fault (100% and 75% increase of nominal value respectively) as shown in Figs. 4.18 and 4.19.

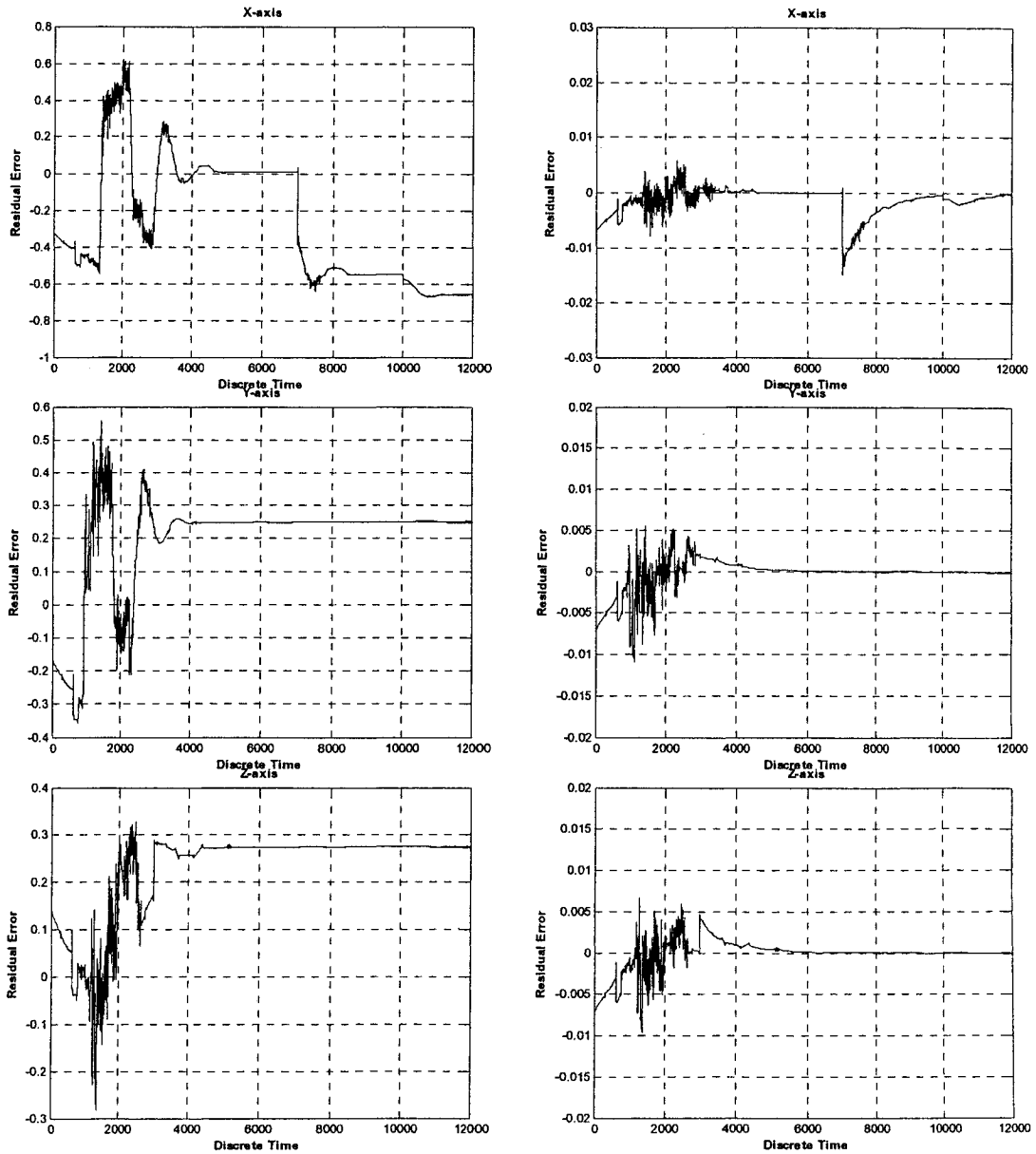


Fig. 4.17 Residual error signals in case of double sequential faults in X-axis: (a) Neural observer output (b) Linear observer output

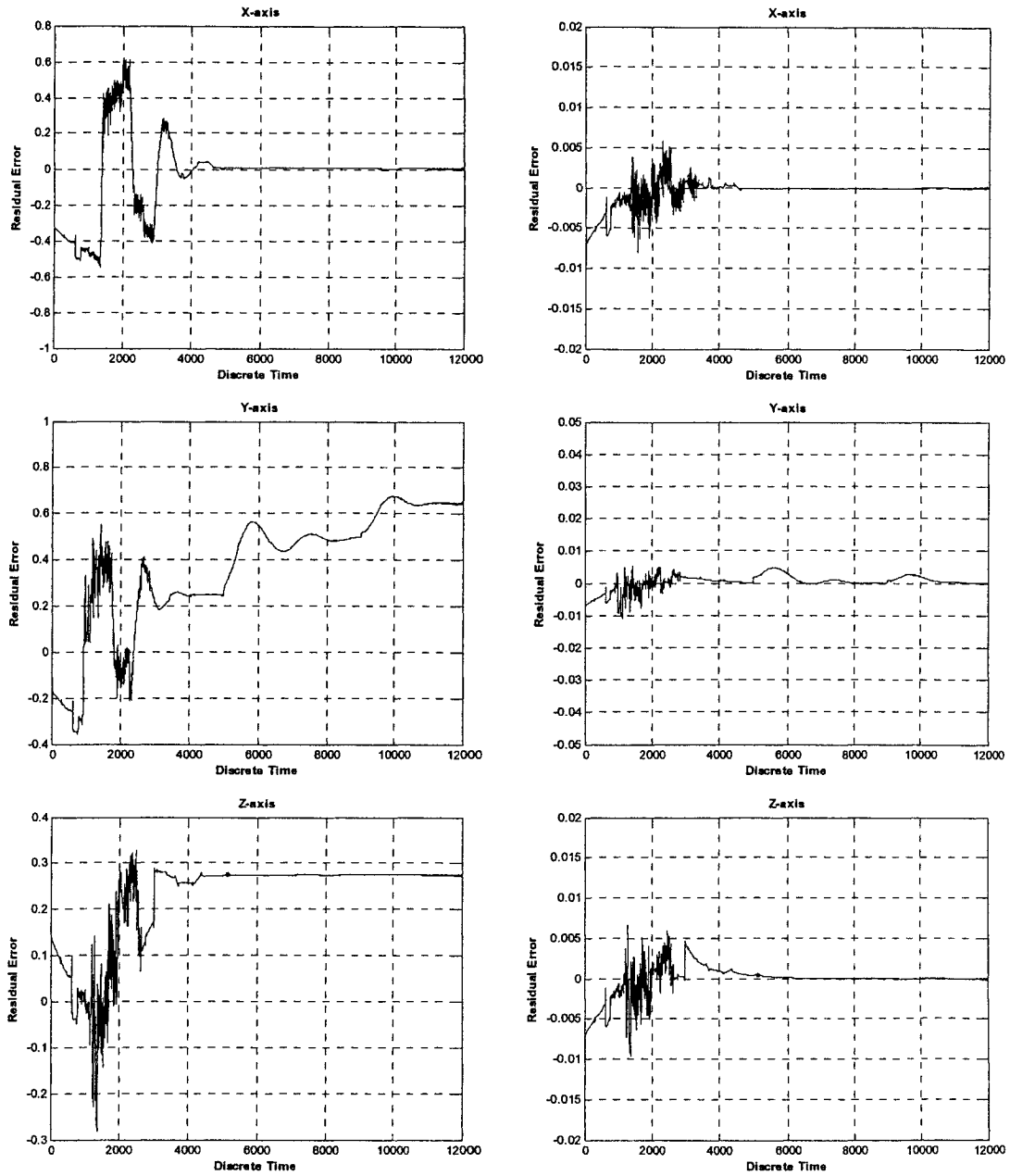


Fig. 4.18 Residual error signals in case of double sequential faults in Y-axis: (a) Neural observer output (b) Linear observer output

The dynamic neural network residual generator, as demonstrated in Figs. 4.17(a), 4.18(a) and 4.19(a) detects the injected double sequential faults rapidly. This indicates that the predicted output of the dynamic neural

network clearly deviates from the output of the actual system. Figs. 4.17(b), 4.18(b) and 4.19(b) show that the linear residual generator observer has failed completely in detecting the severe double faults.

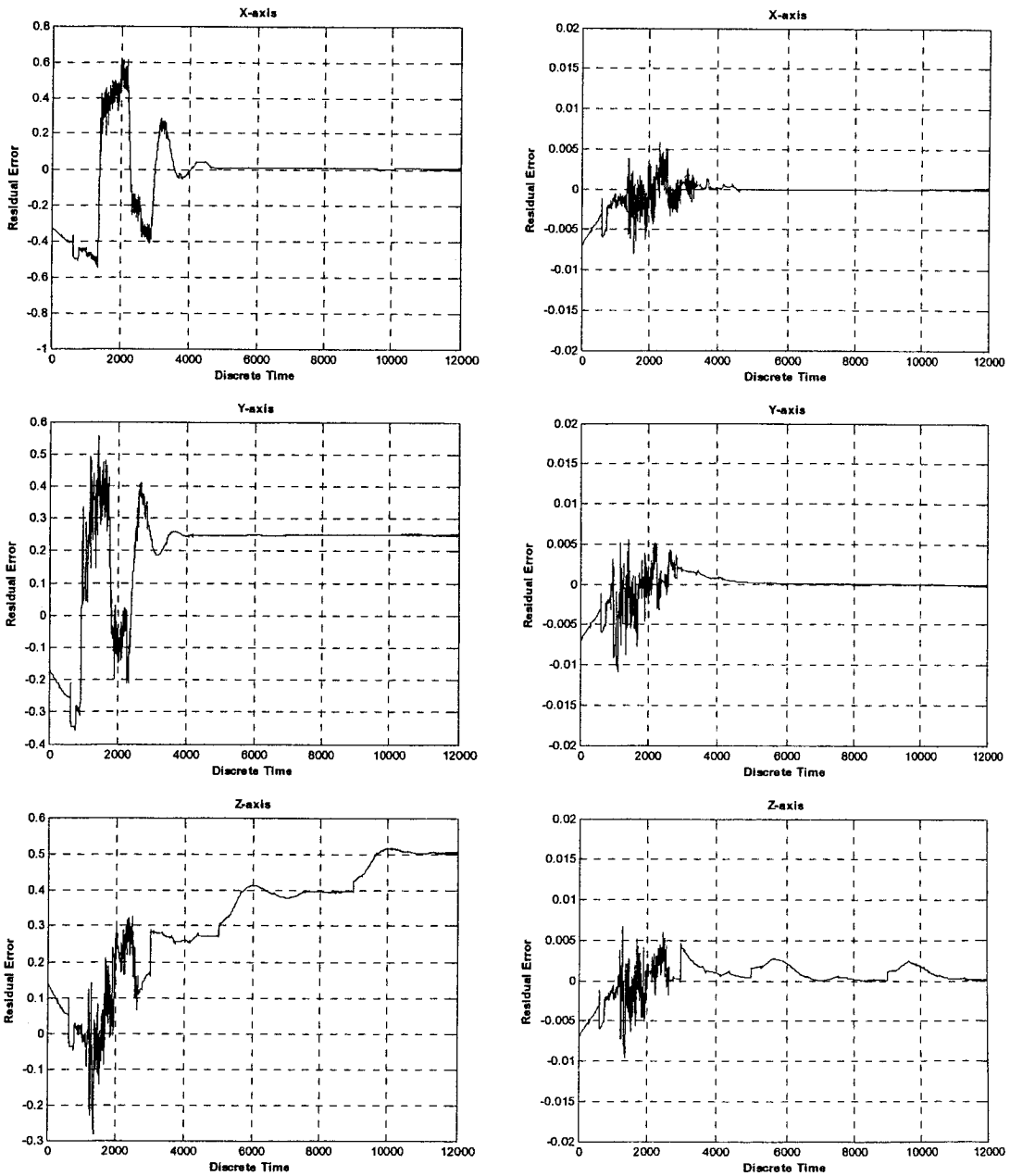


Fig. 4.19 Residual error signals in case of double sequential faults in Z-axis: (a) Neural observer output (b) Linear observer output

Quantitative measuring for the injected double sequential fault is summarized in Table 4.8. The severity of the double sequential fault is confirmed by measured values from the fault scenario figures. Also, the quick response of the neural diagnostic technique is shown quantitatively without a delay time.

TABLE 4.8
QUANTITATIVE SUMMARY OF DOUBLE SEQUENTIAL FAULT SCENARIO FIGURES

X, 50% drop of nominal V_{Bus} value + 50% drop of nominal I_m peak value
Y, 70% drop of nominal I_m peak value + 100% increase of nominal V_{Bus} value
Z, 60% drop of nominal I_m peak value + 75% increase of nominal V_{Bus} value

Faulty Axis		Injection Time (msec)		Detection Time (msec)		Steady State Residual Error					
						DNN			LO		
		1st Fault	2nd Fault	1st Fault	2nd Fault	Normal	1st Fault	2nd Fault	Normal	1st Fault	2nd Fault
X	X	7,000	10,000	7,000	10,000	**0.0075**	**– 0.55**	**– 0.66**	0	0	0
	Y					0.249	0.250	0.251	0	0	0
	Z					0.275	0.275	0.275	0	0	0
Y	X					0.0075	0.007	0.006	0	0	0
	Y	5,000	9,000	5,000	9,000	**0.249**	**0.500**	**0.650**	0	0	0
	Z					0.275	0.275	0.275	0	0	0
Z	X					0.0075	0.007	0.006	0	0	0
	Y					0.249	0.250	0.251	0	0	0
	Z	5,000	9,000	5,000	9,000	**0.275**	**0.395**	**0.510**	0	0	0

4.3.5 Double Concurrent Faults Scenario

The second type of considered double faults scenario is the concurrent faults which is the most sever fault type that has been injected in the actuator

subsystem compared with other fault types. This fact can be observed from Figs. 4.20, 4.21 and 4.22, where this fault was injected in the voltage bus and motor current circuits of each satellite axis simultaneously. Note that we considered the same faults percentages that applied in previous double fault.

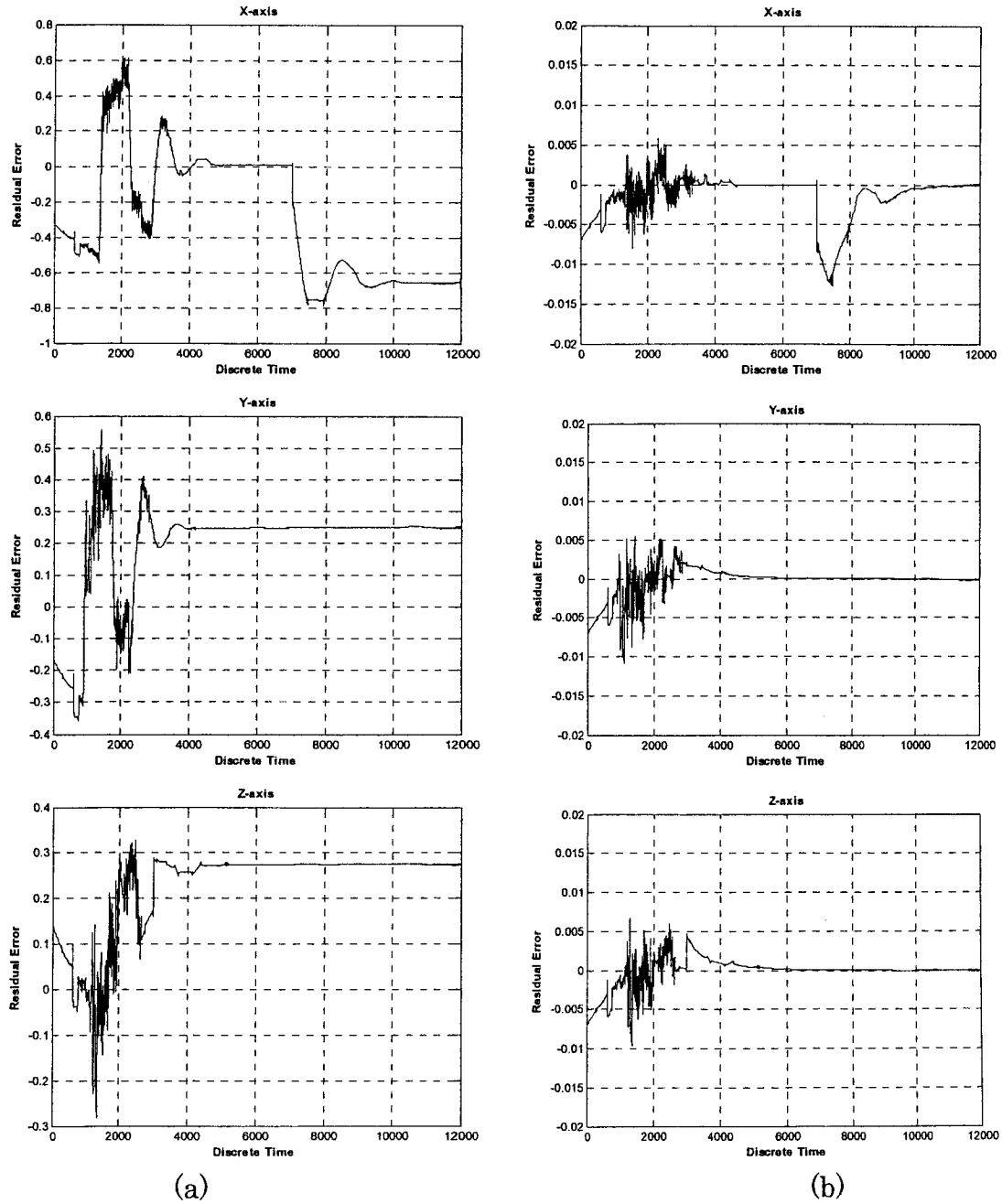


Fig. 4.20 Residual error signals in case of double concurrent faults in X-axis: (a) Neural observer output (b) Linear observer output

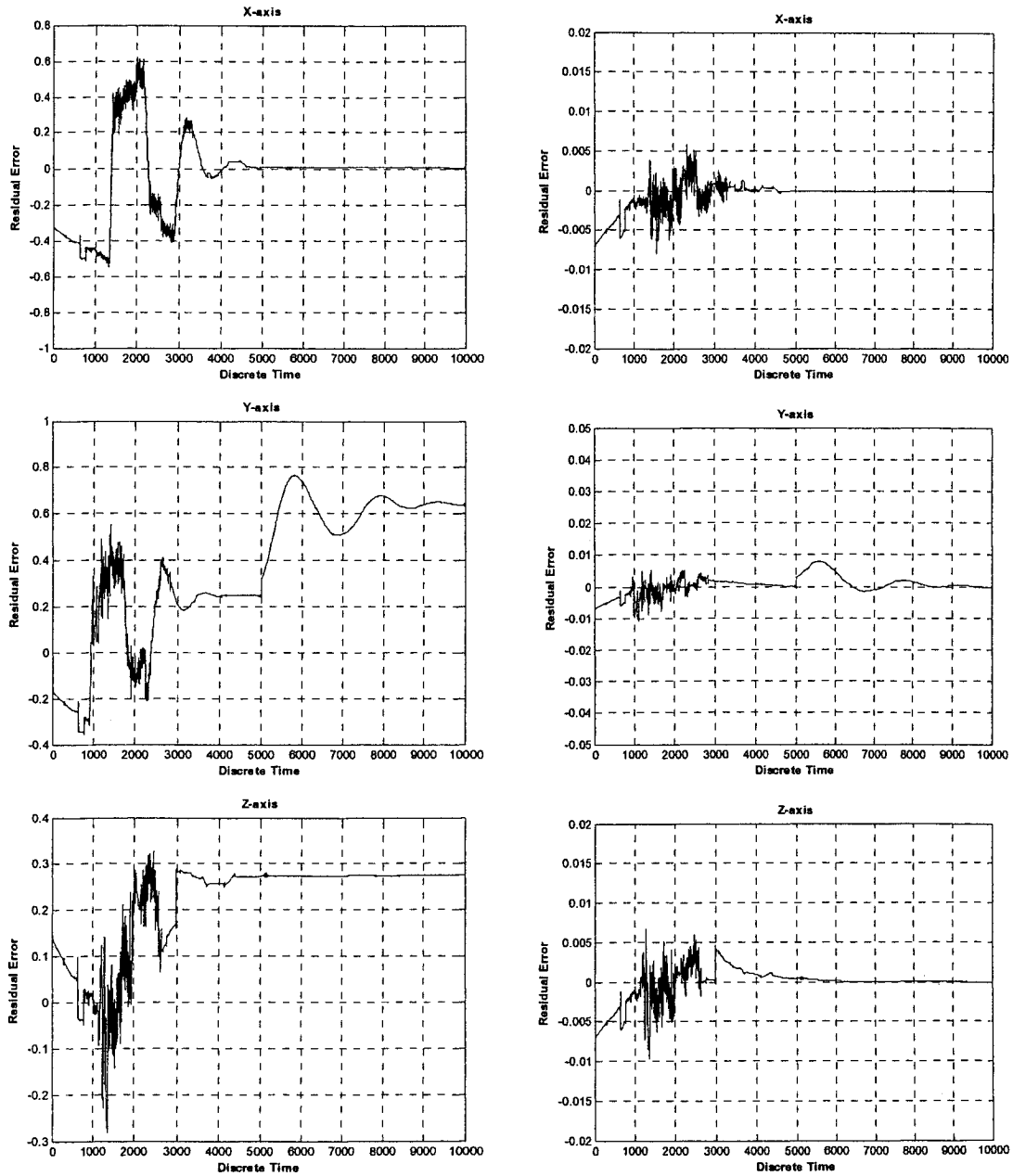


Fig. 4.21 Residual error signals in case of double concurrent faults in Y-axis: (a) Neural observer output (b) Linear observer output

The characteristic of the generated residual signals from double concurrent faults have a large transient time in addition to a large overshoot and a steady state error compared with other fault types. Again, the capabilities of

the intelligent neural observers are shown through observing Figs. 4.20(a), 4.21(a) and 4.22(a). While the linear observers have failed completely in detecting this sever fault as shown in Figs. 4.20(b), 4.21(b) and 4.22(b).

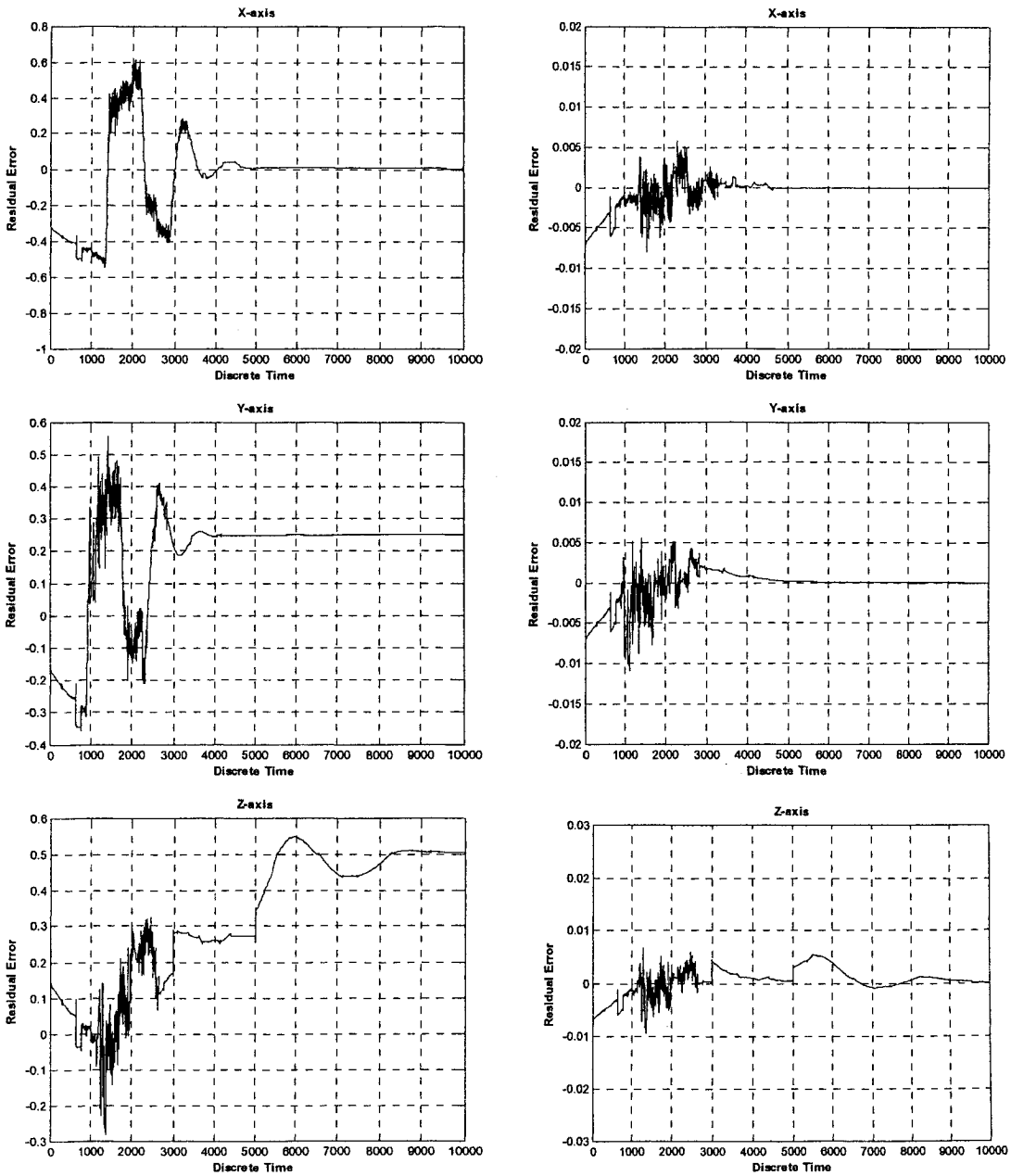


Fig. 4.22 Residual error signals in case of double concurrent faults in Z-axis: (a) Neural observer output (b) Linear observer output

A quantitative summary for double concurrent current faults is provided in Table 4.9. The capabilities of the neural estimator are shown quantitatively in detecting sever DC faults in all satellite axes, while the linear observer has a normal residual error value with persisting of the sever faults and it completely fails in detecting the double concurrent faults.

TABLE 4.9
QUANTITATIVE SUMMARY OF DOUBLE CONCURRENT FAULT SCENARIO FIGURES

X, 50% drop of nominal V_{Bus} value + 50% drop of nominal I_m peak value Y, 70% drop of nominal I_m peak value + 100% increase of nominal V_{Bus} value Z, 60% drop of nominal I_m peak value + 75% increase of nominal V_{Bus} value							
Faulty Axis		Injection Time (msec)	Detection Time (msec)	Steady State Residual Error			
		DNN	DNN	DNN		LO	
				Normal	Faulty	Normal	Faulty
X	X	7,000	7,000	**0.0075**	**– 0.660**	0.0	0.0
	Y			0.249	0.250	0.0	0.0
	Z			0.275	0.275	0.0	0.0
Y	X			0.0075	0.007	0.0	0.0
	Y	5,000	5,000	**0.249**	**0.650**	0.0	0.0
	Z			0.275	0.275	0.0	0.0
Z	X			0.0075	0.007	0.0	0.0
	Y			0.249	0.250	0.0	0.0
	Z	5,000	5,000	**0.275**	**0.510**	0.0	0.0

4.4 Actuator Faults Isolation

As shown in previous sections, we injected various faults scenarios in the actuator subsystem and the first task of FDI system has been fulfilled. In this section we will apply an adaptive neural classifier directly after the neural

observer in order to classify each generated residual error signal to get the information from theses signals, like which fault has occurred and when. In order to perform the isolation task of the neural FDI system, the adaptive neural classifier has to be trained on the residual error signals that are generated from the neural observer, and then the classifier is ready for validation where predefined classes have to be assigned.

Normally, we have two modes of operation; healthy and faulty mode. The healthy mode is assigned for one class, while the faulty mode is assigned for three classes as shown in following Table 4.10.

TABLE 4.10
MODES OF OPERATION AND ITS ASSIGNED CLASSES

Mode of Operation	Assigned Class	
Normal	1	1 0 0 0
Motor current fault	2	0 1 0 0
Bus voltage fault	3	0 0 1 0
Viscous temperature fault	4	0 0 0 1

Three LVQ networks are constructed and trained for the purpose of fault isolation in each axis. The generated residual errors from neural observer are transformed into input vector to the trained networks. The learning process was carried out with learning rate parameter of 0.10 using Matlab *Neural Network Toolbox*; the applied classifiers have a simple structure and a non-complicated training algorithm. For each satellite axis classifier (X, Y and Z),

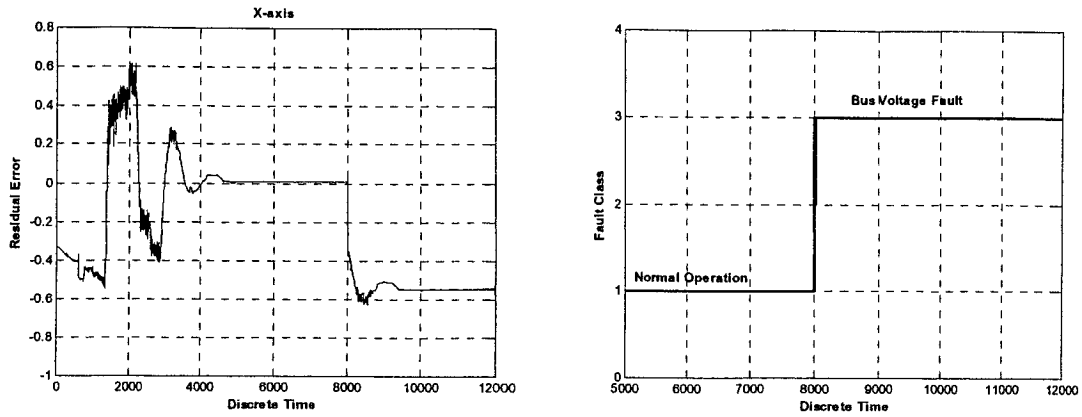
the best results were obtained with the network structure $\mathbf{N_{1\text{-}32\text{-}4}}$ which means one input, 32 neurons in the first competitive layer and 4 neurons in the second linear layer. Thus, we have 32 subclasses in layer 1 that are then assigned to one of 4 output classes by the 4 neurons in layer 2.

In this thesis, the constructed classifiers are employed for isolating only single faults that have been injected in the RW along each satellite axis, as voltage bus faults, motor current faults and viscous temperature faults.

Note that the classifiers are principally performed during the steady state response of the satellite as any large transitory variations may be mistakenly be flagged as faults and this is indeed the case for all the simulation results that are shown in the next sections.

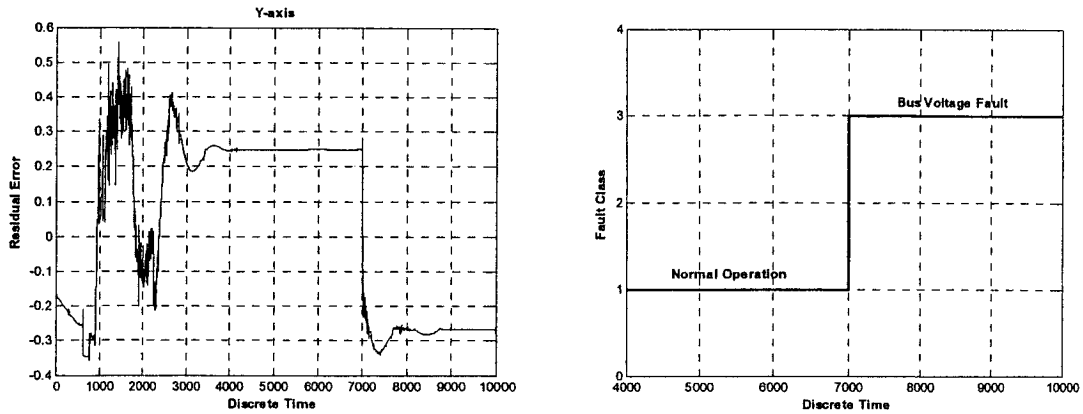
4.4.1 Isolation of V_{Bus} Faults

The performance of the classifiers along each satellite's axis is depicted in Figs. 4.23, 4.24, and 4.25. As seen in these figures, the bus voltage is detectable and isolable completely and with no time delay in each axis. Also each classifier output does provide detailed information about the time and the location of the injected faults. Figs. 4.23(b), 4.24(b), and 4.25(b) show clearly that the behavior of the classifiers is normal before occurring of V_{Bus} fault and then it responded quickly at the instant of fault occurring to indicate that at this moment there is a fault (e.g. 8,000 msec in X-axis) and it belongs to class 3 which is a voltage bus fault class. Quantitative summary for this fault isolation is provided in Table 4.11.



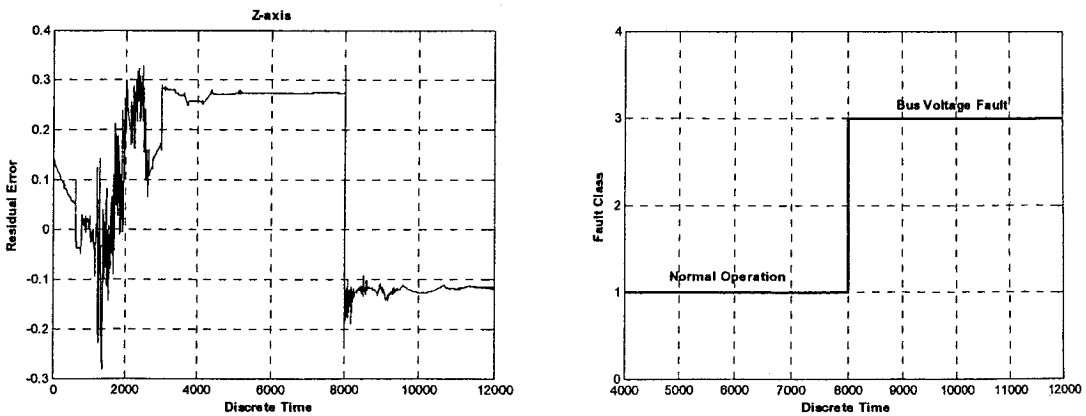
(a) (b)

Fig. 4.23 Detection and isolation of V_{Bus} fault in X-axis: (a) Neural observer output, (b) Classifier output



(a) (b)

Fig. 4.24 Detection and isolation of V_{Bus} fault in Y-axis: (a) Neural observer output, (b) Classifier output

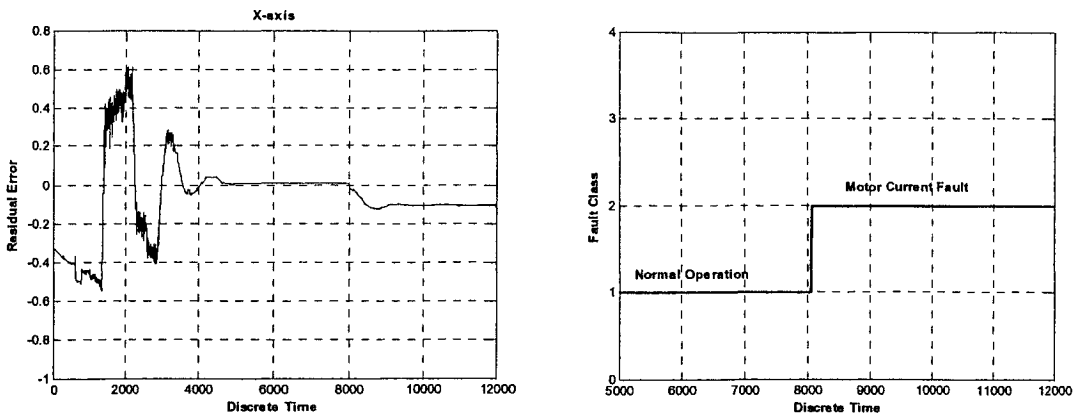


(a) (b)

Fig. 4.25 Detection and isolation of V_{Bus} fault in Z-axis: (a) Neural observer output, (b) Classifier output

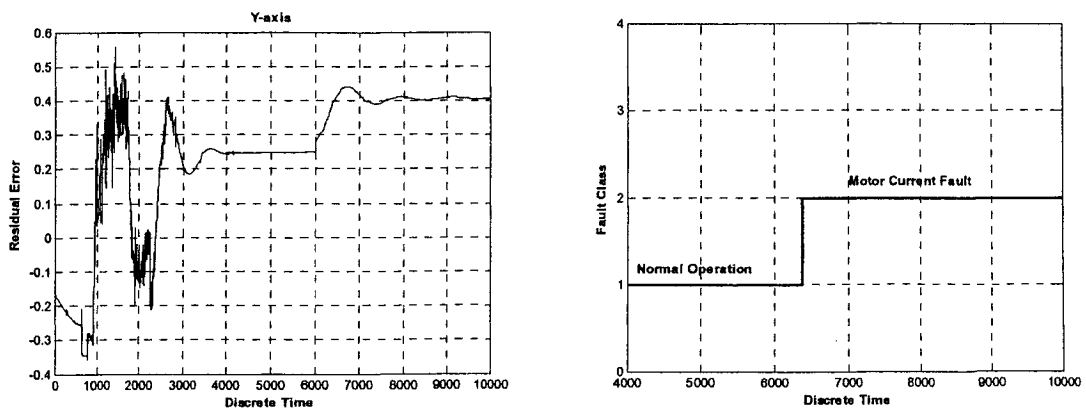
4.4.2 Isolation of I_m Faults

Second type of single faults is motor current fault. As can be observed from Figs. 4.26, 4.27, and 4.28, the motor current fault is detectable and isolable completely and with some time delay in each axis.



(a) (b)

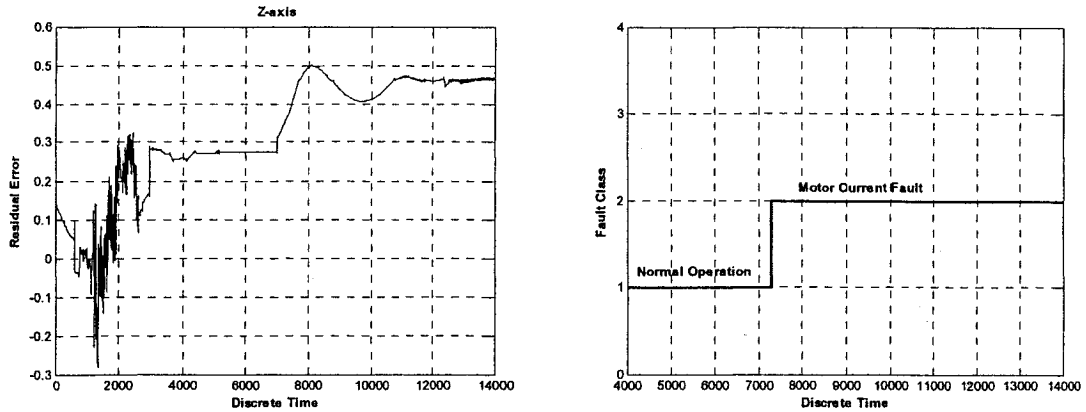
Fig. 4.26 Detection and isolation of I_m fault in X-axis: (a) Neural observer output, (b) Classifier output



(a) (b)

Fig. 4.27 Detection and isolation of I_m fault in Y-axis: (a) Neural observer output, (b) Classifier output

Because of the transit response of the generated residual error as shown in Figs. 4.26(a), 4.27(a), and 4.28(a), during the rise time the residual error is close somehow to the normal situation and it is classified badly to a normal behavior which produce some delay in isolating the motor current fault as shown in Figs. 4.26(b), 4.27(b), and 4.28(b). This fact is summarized quantitatively in Table 4.11.

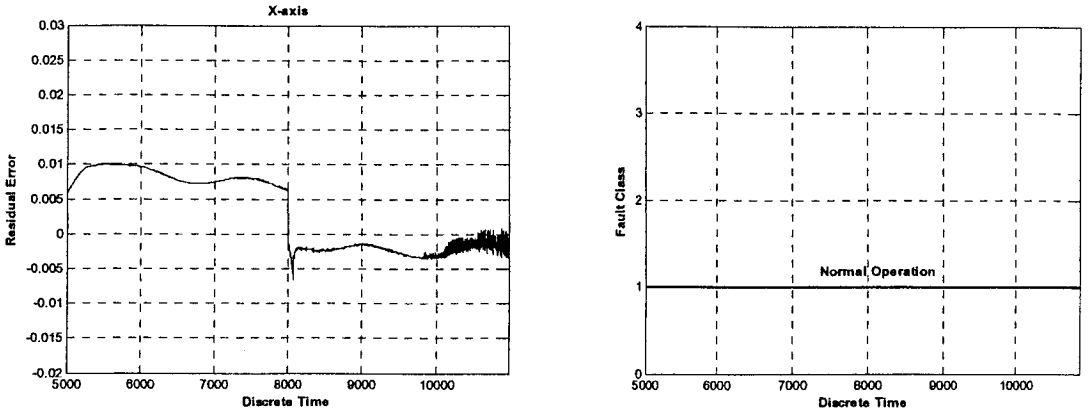


(a) (b)

Fig. 4.28 Detection and isolation of I_m fault in Z-axis: (a) Neural observer output, (b) Classifier output

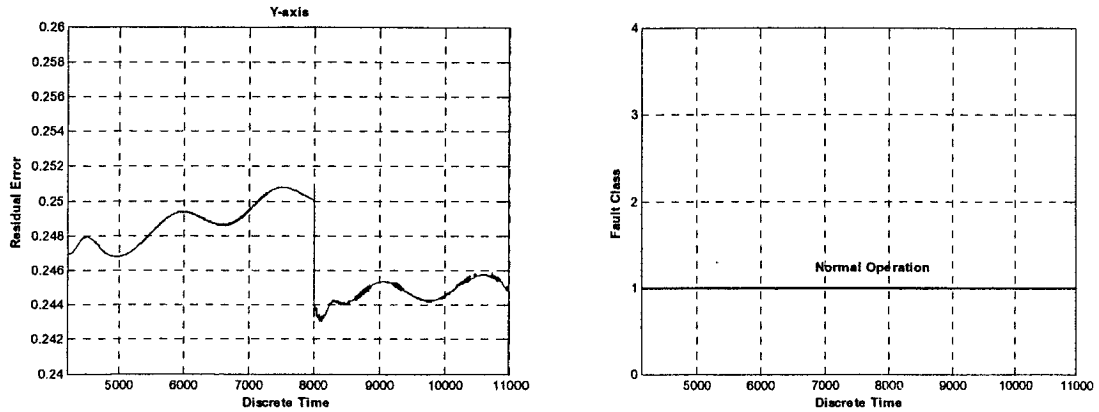
4.4.3 Isolation of τ_v Faults

Last isolation is concerned with a viscous temperature fault and as shown previously that this fault is detectable with some false alarms. Unfortunately, the adaptive neural classifier has failed in recognizing this type of fault and it is not isolable because of the closeness of its residual error vector to the normal operation vector at all times it is misclassified as normal behavior as shown in Figs. 4.29, 4.30, and 4.31.



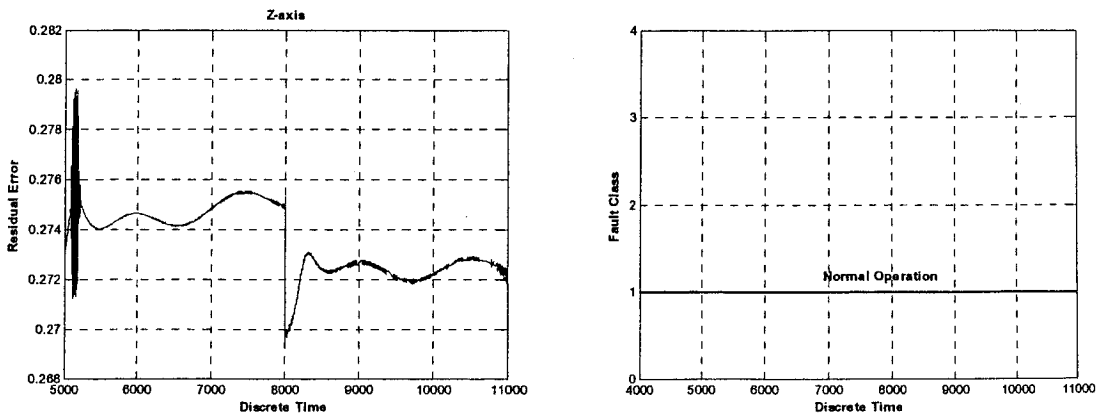
(a) (b)

Fig. 4.29 Detection and isolation of τ_v fault in X-axis:(a) Neural observer output, (b) Classifier output



(a) (b)

Fig. 4.30 Detection and isolation of τ_v fault in Y-axis: (a) Neural observer output, (b) Classifier output



(a) (b)

Fig. 4.31 Detection and isolation of τ_v fault in Z-axis: (a) Neural observer output, (b) Classifier output

Note that the performance of the adaptive classifier is depending on the magnitude of the generated residual error. This means that the performance will be more reliable when the residual error signals of the isolated faults are close to the residuals that have been used during the training of the neural classifier. But if the classifier is applied to isolate new faults that its residuals are far from the trained one then the percentage of wrongly classified system behaviours are increased and the reliability of the classifier is decreased. This fact was verified after we performed a number of simulations that have not reported in thesis.

TABLE 4.11
QUANTITATIVE SUMMARY OF ISOLATION FAULTS FIGURES

Fault Scenario	Faulty Axis	Injection Time (msec)	Detection Time (msec)	Delay Time (msec)
V_{Bus}	X, 50% drop	8,000	8,000	0.0
	Y, 50% drop	7,000	7,000	0.0
	Z, 50% drop	8,000	8,000	0.0
I_m	X, 50% drop	8,000	8,065	65
	Y, 70% drop	6,000	6,367	367
	Z, 70% drop	7,000	7,289	289
τ_v	X, 74.5% increasing	8,000	–	–
	Y, 44.3% drop	8,000	–	–
	Z, 49.5% drop	8,000	–	–

All isolation figures are summarized in Table 4.11, which provides details about the injection time of each fault and its detection time also. In addition

to the amount of time delay that took by classifier to isolate motor current faults while voltage bus fault was isolated quickly without time delay. This shows the capability of the intelligent approach in isolating various fault types using neural network with a mixed structure between supervised and unsupervised learning.

4.5 Conclusions

In this Chapter, various single and double fault scenarios are considered, and the neural Fault Detection and Isolation (FDI) scheme is applied to the actuator subsystem of a satellite. Two neural structures are employed in performing the FDI system tasks. The capabilities of the intelligent approach is demonstrated and compared with a model-based linear observer as bench mark. Finally, the discussions of each simulation results are provided under each section due the large amount of simulation results.

Chapter 5

Conclusions and Future Work

5.1 Thesis Summary

In this thesis, the problem of fault detection and isolation is solved using an artificial intelligent approach. Due to their capabilities to cope with nonlinearity, complexity, uncertainty, and noisy and corrupted data, neural networks are employed in this thesis to solve the problem of Fault Detection and Isolation (FDI) for the Attitude Control Subsystem (ACS) of a satellite. The neural FDI model was developed and applied to detect and isolate the injected faults in the Reaction Wheel (RW) which is often used as an actuator in the attitude control of a satellite. Since there are three actuators that provide reaction torques for three coupled axes in the satellite, we constructed three neural FDI Models. Each FDI model consists of two neural architectures and it is responsible for a possibly faulty actuator.

The first architecture (neural residual generator) was developed based on dynamic neural networks in order to identify the normal and abnormal modes of operation; and then to detect different types of simulated faults in the RW model. A dynamic neural network residual generator is constructed based on the Dynamic Multilayer Perceptron Network (DMLP) in order to generate residual signals that can distinguish between the faulty and non-faulty situation. A generalized embedded structure for the dynamic neuron model is considered in the DMLP network.

The second architecture (adaptive neural classifier) was developed based on the static neural networks. The fulfillment of isolation stage in FDI algorithm has been done using a simple and non-complicated structure of Learning Vector Quantization (LVQ) network as an adaptive neural network classifier which provided detailed information about the occurrence time and the location of various simulated faults.

A generic three-axis stabilized satellite control model based on the reaction wheels and with no momentum bias has been developed using Matlab-SIMULINK. Three separate PD controllers were designed and employed to command the three reaction wheels for control of each satellite axis. In order to make the developed three axis ACS model as realistic as possible, nonlinear and coupled ACS equations of motion are considered in addition to the nonlinear reaction wheel model.

In this work, five faults scenarios have been considered and simulated during the steady state period. In the first fault scenario, a low bus voltage fault was injected in the back-EMF circuit of the reaction wheel. This kind of faults was considered due to the sensitivity of the generated reaction torque from the RW when the bus voltage is decreased. Second fault scenario was injected in the circuit of motor torque control. Low motor current fault was considered and as a result, a shortage in the supplied RW torque will affect the satellite control. Third fault scenario was injected artificially in the bearing friction model, a viscous temperature fault was considered as abnormal behavior in the RW. Fourth and fifth fault scenarios were injected in both circuits of motor current and bus voltage and considered as double sequential and double concurrent faults respectively.

From the simulation results shown it can be concluded that the dynamic neural residual generator has produced a very reliable performance in detecting various single and double faults that have been injected into the actuator subsystem. Even small faults that could lead over time to a serious damage in the monitored system were being detected by the intelligent observer.

Comparisons with a linear model-based observer acting as a residual generator are also included to demonstrate the capabilities and advantages of our proposed dynamic neural network scheme. We have shown that the performance of the linear residual generator was indeed poor and

unacceptable in detecting all the injected faults, justifying and necessitating the additional computational burden that is associated with a dynamic neural network-based approach.

Also the simulation results have shown the ability of the diagnostic system to differentiate between different failures. Two types of the single faults were isolable. In other words we can say that bus voltage and motor current faults were detectable and isolable while the viscous temperature was detectable but not isolable, unfortunately.

5.2 Recommendations for Future Work

Our plans for future research have many directions:

- Since the proposed EDBP algorithm in this thesis usually finds one of the local minimums, which lead to the necessity for optimal algorithms of global optimization, e.g. evolutionary algorithms, simulated annealing algorithms, and adaptive random search stochastic method, to overcome the problem of getting stuck in local minimum through training. Furthermore, improving the quality of identification will result in improving the quality of fault diagnosis process.
- Another future Developing an advanced isolation technique that could isolate both of single and double faults, such as nonlinear (feedforward neural network) classifiers and Neuro-Fuzzy (NF) systems

- In the considered ACS model, we may also include more cross coupling, in order to capture the real behavior of such highly nonlinear dynamical system.

Bibliography

[1] R. Patton, F. Uppal & C. Lopez-Toribio, " Soft computing approaches to fault diagnosis for dynamic systems: a survey," *4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, Budapest, Vol. 1, pp. 298 – 311, June 2000.

[2] P. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge–based redundancy: a survey and some new results," *Automatica*, Vol. 26, pp. 459 – 474, 1990.

[3] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. Kavuri, "A review of process fault detection and diagnosis – Part I: Quantitative model–based methods," *Computers & Chemical Engineering*, Vol. 27, No. 3, pp. 293 – 311, March 2003.

[4] S. Simani, C. Fantuzzi, and R. Patton, *Model–Based Fault Diagnosis in Dynamic Systems Using Identification Techniques*, Springer, 2003.

[5] R. Patton, P. Frank and R. Clark (Eds), *Issues of Fault Diagnosis for Dynamic Systems*, Springer, 2000.

[6] J. Gertler, *Fault Detection and Diagnosis in Engineering Systems*, Marcel Dekker, 1998.

[7] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. Kavuri, " A review of process fault detection and diagnosis – Part II: Qualitative

models and search strategies," *Computers & Chemical Engineering*, Vol. 27, No. 3, pp. 313 – 326, March 2003.

[8] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. Kavuri, " A review of process fault detection and diagnosis – Part III: Process history based methods," *Computers & Chemical Engineering*, Vol. 27, No. 3, pp. 327 – 346, March 2003.

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

[10] P. Frank and B. Koppen–Seliger, "New development using AI in fault diagnosis," *Automatica*, Vol. 1, pp. 3 – 14, 1997.

[11] P. Frank and B. Koppen–Seliger, "Fuzzy logic and neural network applications to fault diagnosis," *Automatica*, Vol. 1, pp. 67 – 88, 1997.

[12] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, Vol. 1, pp. 4–27, March 1990.

[13] S. Leonhardt and M. Ayoubi, "Methods of fault diagnosis," *Control Engineering Practice*, Vol. 5, No. 5, pp. 683 – 692, 1997.

[14] X. Red, A. Rad and P. Chan, "Identification and control of nonlinear systems using dynamic neural networks," *Proceedings of the 4th World Congress on Intelligent Control and Automation*, pp. 2002–2006, 2000.

[15] A. Yazdizadeh and K. Khorasani, "Nonlinear system Identification using embedded dynamic neural network structures," *IEEE*

International Joint Conference on World Congress on Computational Intelligence, Vol. 1, pp. 378 – 383, 1998.

[16] T. Marcu, L. Mirea, P. Frank, "Neural observer schemes for robust detection and isolation of process faults" *Control '98. UKACC International Conference*, Vol. 2 pp. 958 – 963, 1998

[17] T. Sorsa, and H. Koivo, "Application of artificial Neural networks in process fault diagnosis," *Automatica*, pp. 843 – 849, 1993.

[18] A. Shaw , F. Doyle III, and J. Schwaber, "A Dynamic Neural Network Approach to Nonlinear Process Modeling," *Computers & Chemical Engineering*, Vol. 21, No.4, pp. 371 – 385, 1997.

[19] P. Xiaoqin, F. Chowdhury, "Unsupervised neural network for fault detection and classification in dynamic systems Control Applications," *Proceedings of the 1999 IEEE International Conference on* Vol. 1, pp. 640 – 645, Aug.1999.

[20] A. Alessandri, "Fault diagnosis for nonlinear systems using a bank of neural estimators," *Computers in Industry*, Vol. 52, No. 3, pp. 271 – 289, Dec. 2003.

[21] M. Karpenko, N. Sepehri, and D. Scuse, "Diagnosis of process valve actuator faults using a multilayer neural network," *Control Engineering Practice*, Vol. 11, No. 11, pp. 1289 – 1299, Nov. 2003.

[22] T. Sorsa, H. Koivo, and H. Koivisto, "Neural networks in process fault diagnostics," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No.4, pp. 815 – 825, 1991.

[23] J. Korbicz, "Neural networks and their application in fault detection and diagnosis," *Proc. IFAC. Fault Detection Supervision and Safety for Technical Processes, SAFEPROCESS'97*, Kingston, Upon Hull UK, Vol. 1, pp. 367 – 623, 1997.

[24] J. Korbicz, K. Patan, "Dynamic neural network with filter of different orders," *4rd Int. Symp. on Methods and Models in Automation and Robotics, MMAR'97*, Miedzyzdroje, Poland, Vol. 2, pp. 745 – 750, 1997.

[25] A. Yazdizadeh, K. Khorasani, and R. Patel, "Identification of a two–link flexible manipulator using adaptive time delay neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 30, No.1, pp. 165 – 172, 2000

[26] N. Sinha, M. Gupta and D. Rao, "Dynamic neural networks: an overview," *Proceedings of IEEE International Conference on Industrial Technology 2000*, Vol. 2, pp. 491–496, Jan. 2000.

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

[28] D. Rao and M. Gupta, "Performance comparison of dynamic neural networks as applied to robot inverse kinematic computations," *American Control Conference*, Vol. 2, pp. 2153 – 2157, July 1994.

[29] T. Marcu, P. Frank, L. Mirea, "Development of dynamic neural networks with application to observer–based fault detection and isolation," *International Journal of Applied Mathematics and Computer Science*, Vol. 9, No. 3, pp. 547 – 570, 1999

[30] M. Hagan, H. Demuth, M. Beale, *Neural Network Design*, PWS publisher, 1996.

[31] A. Yazdizadeh and K. Khorasani, "Adaptive time delay neural network structures for nonlinear system identification," *Neurocomputing*, Vol. 47, pp. 207 – 240, 2002.

[32] D. Rao and M. Gupta, "Dynamic neural units and function approximation," *IEEE International Conference on Neural Networks*, Vol. 2, pp. 743 – 748, April 1993.

[33] M. Gupta and D. Rao, "A neural processor for coordinating multiple systems with dynamic uncertainties," *Proc. Second International Symposium on Uncertainty Modeling and Analysis*, pp. 633 – 640, April 1993.

[34] J. Korbicz, A. Obuchowic and K. Patan, "Network of dynamic neurons in fault detection system," *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1862 – 1867, 1998.

[35] D. Rao and M. Gupta, "a generic neural model based on excitatory - inhibitory neural population," *Proceeding of International Joint Conference on Neural Networks, IJCNN'93*, Vol. 2, pp. 1393 – 1396, Oct. 1993.

[36] J. Korbicz, K. Patan, and A. Obuchowicz, "Dynamic neural networks for process modelling in fault detection and isolation systems," *International Journal of Applied Mathematics and Computer Science*, Vol. 9, No. 3, pp. 519 – 546, 1999.

[37] A. Yazdizadeh and K. Khorasani, "Identification of a class of nonlinear systems using dynamic neural network structures," *IEEE International Joint Conference on Neural Networks*, Vol. 1, pp. 194 – 198, 1997.

[38] M. Ayoubi, "Fault diagnosis with dynamic neural structure and application to a turbo–charger," *Proc. Int. Symp. Fault Detection Supervision and Safety for Technical Processes, SAFEPROCESS'94*, Espoo, Finland, Vol. 2, pp. 618 – 623, 1994.

[39] K. Patan, and J. Korbicz, "Application of dynamic neural networks in modeling and identification," *3rd Int. Symp. Methods and Models in Automation and Robotics, MMAR'96*, Miedzyzdroje, Poland, vol. 3, pp. 1141–1146, 1996.

[40] K. Patan, and T. Parisini, "Stochastic learning methods for dynamic neural networks: simulated and real-data comparisons," *Proc. 2002*

American Control Conference, ACC'02, Anchorage, Alaska, USA, pp. 2577 – 2582, May 2002.

[41] T. Marcu and L. Mira, "Robust detection and isolation of the process faults using neural networks," *IEEE Control Systems Magazine*, Vol. 17 ,No. 5, pp. 72 – 79, Oct. 1997.

[42] K. Patan, A. Obuchowicz, and J. Korbicz, "Network of dynamic neurons approach to residual generators," *5th Int. Symp. Methods and Models in Automation and Robotics, MMAR'98*, Miedzyzdroje, Poland, Vol. 2, pp. 645 – 650, 1998.

[43] K. Patan, and T. Parisini, "Identification of neural dynamic models for fault detection and isolation: the case of a real sugar evaporation process," *Journal of Process Control*, Vol. 15, No. 1, pp. 67 – 79, 2004.

[44] I. Al-Zyoud, and K. Khorasani, "Detection of Actuator Faults Using a Dynamic Neural Network for the Attitude Control Subsystem of a Satellite," *Proceedings 2005 IEEE International Joint Conference on Neural Networks, IJCNN'05*, Montréal, Canada, pp. 1746 – 1751, 2005.

[45] L. Ma and K. Khorasani, "A new strategy for adaptively constructing multilayer feedforward neural networks," *Neurocomputing*, Vol. 51, pp. 361– 385, 2003.

[46] L. Ma and K. Khorasani, "New training strategies for constructive neural networks with application to regression problems," *Neural Networks*, Vol. 17, No. 4, pp. 589 – 609, 2004.

[47] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Ed., Prentice–Hall, 1999.

[48] H. Demuth, and M. Beale, *Neural Network Toolbox User's Guide*, Version 4, Mathworks, 2004.

[49] C. Brown, *Elements of Spacecraft Design*, AIAA education series, 2002.

[50] R. Froelich and H. Papapoff, "Reaction wheel attitude control for space vehicles," *Automatic Control, IRE Transactions*, Vol. 4, No. 3, pp. 139 – 149, Dec. 1959.

[51] P. Fortescue and J. Stark (Eds.), *Spacecraft Systems Engineering*, Second Edition, John wiley and sons, 1995.

[52] W. Larson and J. Wertz (Eds.), *Space Mission Analysis and Design*, Second Edition, Kluwer Academic Publishers, 1999.

[53] M. Griffin and J. French, *Space Vehicle Design*, AIAA education series, 1991.

[54] B. Bialke, "High fidelity mathematical modeling of reaction wheel performance," *21st Annual American Astronautical Society Guidance and Control Conference*, pp. 483 – 496, Feb. 1998.