Worcester Polytechnic Institute
Digital WPI

Masters Theses (All Theses, All Years) | Electronic Theses and Dissertations

2002-02-22

Automated On-line Diagnosis and Control Configuration in Robotic Systems Using Model Based Analytical Redundancy

Vitaly M. Kmelnitsky
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/etd-theses

Repository Citation
Kmelnitsky, Vitaly M., "*Automated On-line Diagnosis and Control Configuration in Robotic Systems Using Model Based Analytical Redundancy*" (2002). *Masters Theses (All Theses, All Years)*. 167.
https://digitalcommons.wpi.edu/etd-theses/167

This thesis is brought to you for free and open access by Digital WPI. It has been accepted for inclusion in Masters Theses (All Theses, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.

AUTOMATED ON-LINE DIAGNOSIS AND CONTROL CONFIGURATION IN ROBOTIC SYSTEMS USING MODEL BASED ANALYTICAL REDUNDANCY

by

Vitaly M. Kmelnitsky

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements of the

Degree of Master of Science

in

Mechanical Engineering

by

_____

January 2002

APPROVED:

_____

Dr. Michael A. Demetriou, Mechanical Engineering Dept., Advisor

_____

Dr. David J. Olinger, Mechanical Engineering Dept., Committee Member

_____

Dr. Zhikun Hou, Mechanical Engineering Dept., Committee Member

_____

Dr. Nikos A. Gatsonis, Mechanical Engineering Dept., Committee Representative

ABSTRACT

Because of the increasingly demanding tasks that robotic systems are asked to perform, there is a need to make them more reliable, intelligent, versatile and self-sufficient. Furthermore, throughout the robotic system's operation, changes in its internal and external environments arise, which can distort trajectory tracking, slow down its performance, decrease its capabilities, and even bring it to a total halt. Changes in robotic systems are inevitable. They have diverse characteristics, magnitudes and origins, from the all-familiar viscous friction to Coulomb/Sticktion friction, and from structural vibrations to air/underwater environmental change. This thesis presents an on-line environmental **C**hange, **D**etection, **I**solation and **A**ccommodation (**CDIA**) scheme that provides a robotic system the capabilities to achieve demanding requirements and manage the ever-emerging changes. The CDIA scheme is structured around a priori known dynamic models of the robotic system and the changes (faults). In this approach, the system monitors its internal and external environments, detects any changes, identifies and learns them, and makes necessary corrections into its behavior in order to minimize or counteract their effects. A comprehensive study is presented that deals with every stage, aspect, and variation of the CDIA process. One of the novelties of the proposed approach is that the profile of the change may be either time or state-dependent. The contribution of the CDIA scheme is twofold as it provides robustness with respect to unmodeled dynamics and with respect to torque-dependent, state-dependent, structural and external environment changes. The effectiveness of the proposed approach is verified by the development of the CDIA scheme for a SCARA robot. Results of this extensive numerical study are included to verify the applicability of the proposed scheme.

ACKNOWLEDGEMENTS

I would like to express great appreciation and thanks to Worcester Polytechnic Institute and to all the people who I came in contact with during the five years of my undergraduate and graduate studies. This thesis would not be possible without Professor Michael Demetriou, who envisioned this research and advised me through it every step of the way.

My greatest thanks and dedication is to my parents, who are my greatest inspiration and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A - a set of a priori known changes
a - center of the position-dependent gaussian network neurons
B - matrix of change history profiles
b - center of the velocity-dependent gaussian network neurons
C - parameters of the fault *m*
D - dynamic detection threshold
d - element of the dynamic detection threshold vector
e - state estimation error
f - change (fault) dynamics
F - general representation of the change
G - gravitational torque
h - parameters in the torque-dependent neuron
H - matrix consisting of parameters from the torque-dependent neuron
j - matchability factor
J - identification threshold
k - total number of neurons or dynamic functions
l - parameters of the position dependent neuron
L - matrix consisting of parameters from the position dependent neurons
$l_{(\ )}$ - length of the robot link
M - inertia matrix
$m_{(\ )}$ - weight of the robot link
n - total number of states (degrees of freedom)
N - total number of known changes
p - change profile parameter
P - region in the parameter *p* history where change is present
q - dynamic function in position dependent neuron
Q - matrix of dynamic functions from position dependent neuron
r - element of the isolation threshold vector
R - isolation threshold vector
s - parameters of the velocity dependent neuron
S - matrix consisting of parameters from the position dependent neuron
t_{dt} - detection time

t_{is} - isolation time

t_{pr} - change absence limit

v - number of types of changes in combination

U - Lyapunov function

V - coriolis and centripetal forces

w - width of the velocity gaussian network neurons

w - parameters in change dynamics

ω - dynamic functions of i^{th} change

x - displacement of the fourth link of the SCARA robot

z - matrix of dynamic functions for the velocity dependent neuron

Z - dynamic function in velocity dependent neuron

β - change history profile of the i^{th} the state

θ - joint displacement

τ - input torque

τ_o - nominal input torque

η - unmodeled dynamics

η_o - upper bound of unmodeled dynamics

μ_m - equivalency deviation between the true fault dynamics and the m^{th} isolation filter dynamics

$\tilde{\mu}_m$ equivalency margin for the m^{th} isolation filter

Φ - set of all the changes in the system

ρ - Idle-monitoring threshold

σ - width of the position gaussian network neuron

γ - detection/approximation stability matrix

Υ - matrix of velocity-dependent approximation gains

Γ - matrix of torque-dependent approximation gains

Ψ - matrix of position-dependent approximation gains

$(\ )_\tau$ torque-dependent

$(\ )_\theta$ - state-dependent

$(\hat{\ })$ - estimate

$diag(\ )$ - matrix whose diagonal elements are the entries of the vector

1 INTRODUCTION

Robotic systems play an essential role in our society, and their presence and our dependence on them are increasingly growing. Manufacturing industry has been able to make tremendous leaps only due to the advances in robot technology. Robotic systems are the best and most of the time the only replacement to human beings in applications where human presence is either not possible or harmful. Such applications include space and underwater exploration, radioactive environments, automated bomb detonation, fire-hazardous environments and many more (Figure 1).

Figure 1 - Robotic system.

1.1 Problem Statement

The external operational environment of robotic systems either willingly or unwillingly evolves constantly (Figure 2). Some autonomous robotic systems have to operate both in air and water. Atmospheric pressure, temperature, and humidity are constantly varying, and of course wind can exert extreme forces on the system. The internal environment in robotic system is very unstable as well, and it can exert even larger dynamic changes (Figure 2). Friction, degradation/wear, noise, vibration, and etc. are regular guests in any robotic system.

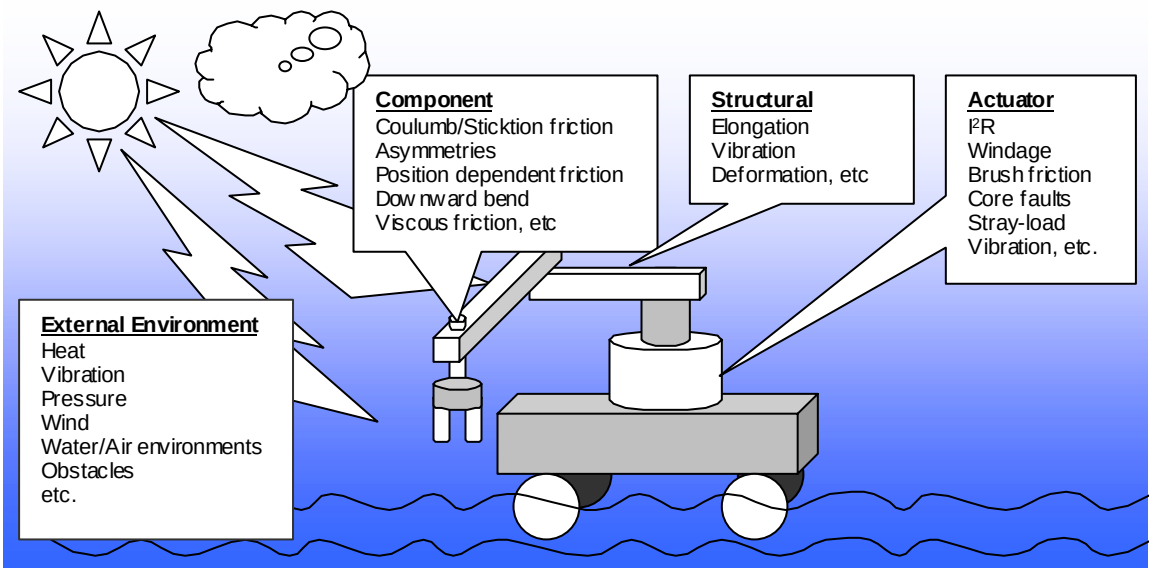


Figure 2 - Internal & external changes.

Changes (faults) can make the system unsafe and less reliable. Productivity of the robotic system can degrade because changes can impose performance limitations on the system and may also require frequent system shut downs for its maintenance. In the case of technologically challenging applications, like space or underwater technology, where a system's full automation is expected, the presence of changes can limit what engineers can accomplish in their designs. The bottom line effect of the changes is on environmental and human safety, cost, and ability of creation of autonomous systems.

1.2 Goals

Fortunately, all of the above-described situations can be managed by giving the system *self-diagnostic* capabilities, which allow it to detect any changes, analyze them and handle them appropriately. The system's ability to learn how its environment has changed makes it more self-sufficient and intelligent, and improves its behavioral

decisions. Self-diagnosis of the system can be accomplished by the introduction of either *analytical* or *hardware* redundancy. In the hardware redundancy approach, additional physical instrumentation is introduced, sensors for instance. In the analytical redundancy approach, additional software is introduced which usually employs model-based techniques [17][26]. Analytical redundancy is less expensive, much easier to upgrade and has more potential. It requires a lot of computational resources because of its on-line application. Recent improvements in digital processing technology provide tools for its present-day development and implementation, which was not visible even a decade ago.

Because the exact dynamic models of the changes in the robot are never known a priori, they should be accounted for in the control design. In robotic systems, the primary source of the changes is at a manipulator's joints. Due to the highly nonlinear nature of joint change dynamics in the robotic systems, any linear models used by a *change-monitoring* scheme cannot accurately represent their dynamics, and as joint velocities and accelerations reach high values, such models fail to capture the salient features of robot motion. In earlier works in change monitoring a number of nonlinear models have been proposed in [15][16][21][25][26], but most of these models have limitations and do not reflect the whole spectrum of the possible changes and change configurations.

A change is classified as any deviation in the robotic system's environment from the originally anticipated one [10]. All of the earlier researched change models ignore two very important factors in the change dynamics. First, the presence of change is not only time dependent, but it also depends on other parameters in the change dynamics (states for instance). Second, the torque-dependent changes should not be ignored and should be treated separately from the state-dependent changes. They affect a robotic system's behavior just as extremely as state-dependent changes do, and by treating them separately

additional improvements are possible (Section 3.8). In this thesis a change model is proposed that addresses both of these factors.

This thesis goal was to design an analytical redundancy model-based technique that makes a robotic system more intelligent, self-sufficient, improves its performance, life span, and on top of all can be very cost efficient.

1.3 Contributions and Innovations

The current research effort makes a number of major steps. It brings the automated change (fault) diagnosis and accommodation area to the whole new level by introducing the ***C****hange* ***D****etection*, ***I****solation*, and ***A****ccommodation* (CDIA) approach.

Unlike well-investigated FDA (Fault Detection and Accommodation [17]), and FDI (Fault Detection and Isolation [21]) schemes that follow the evolution of the change (fault) just up to the accommodation stage, CDIA is an all-encompassing approach that manages the changes (faults) throughout the whole life cycle of the robotic system.

Moreover, unlike the FDI, FDA and other conventional treatments of the internal fault, CDIA deals with the general concept of the changes in both internal and external environments.

CDIA can be divided into four distinct stages: (i) *detection*, (ii) *isolation*, (iii) *accommodation*, and (iv) *idle-monitoring*. The idle-monitoring is a new concept that makes CDIA a complete scheme by monitoring the evolution of the change well after the accommodation stage. It allows dynamic repetition of the CDIA stages.

The isolation stage has been enhanced by constructing a bank of isolation filters from all the combinations of a priori known types of changes (faults). In previous treatments,

each filter corresponded to a specific change (faults) type, but in the proposed approach each filter corresponds to the combined dynamics of multiple changes that occur in the system simultaneously.

The CDIA is developed based on the innovative approach that models the change (fault) history profile as parametric. The new model reflects the true dynamics of the change in a way that previous models were not able to. Its structure resonates improvements in each of the stages of CDIA, but especially in isolation stage by allowing minimization of the number of the isolation filters and narrowing down the isolation effort.

1.4 What is CDIA?

As an example, Figure 3 depicts arbitrary change dynamics development in the single state. This scenario can repeat itself multiple times, depending on the history of the change (Section 2.2.2).

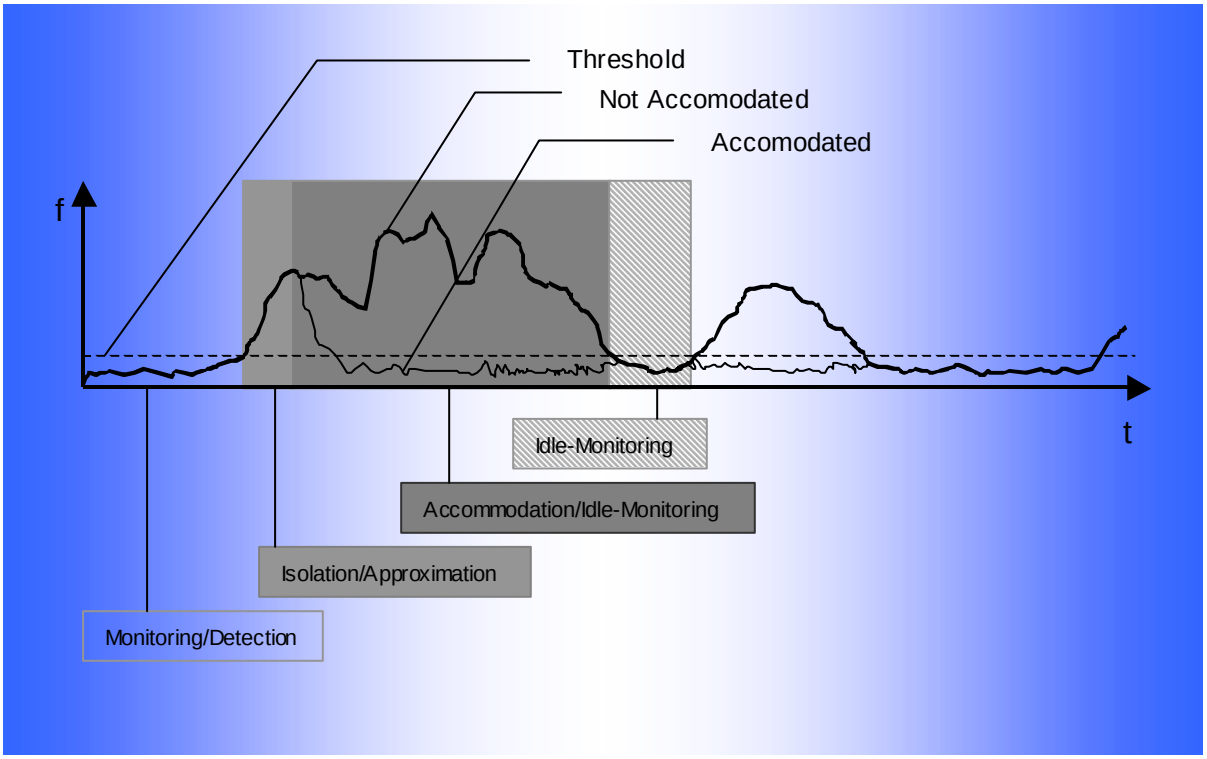


Figure 3 - CDIA scenario.

During the detection stage, changes (faults) are monitored and detected using a detection/approximation observer, which is robust with respect to unmodeled dynamics. The detection/approximation observer is also used to approximate changes whose dynamics are not found to be equivalent to any a priori known change scenarios. The dynamics of the change can be approximated using on-line approximation techniques, which include: *multi-layer neural networks*, *polynomials*, *rational functions*, *spline functions*, *radial-basis-function (RBF) networks*, *adaptive fuzzy systems*, etc [3][17]. From the past experience, RBF networks performed very well in robotic applications. For this reason, they are employed in this thesis for approximation purposes [17].

The isolation stage of the CDIA employs a bank of isolation filters. There are multiple numbers of a priori known types of change dynamics. In a general situation,

these changes may occur one type at a time, in multiple and concurrent combinations, or in multiple and asynchronous combinations. Therefore, the bank of isolation filters consists of all combinations of the concurrent a priori known types of changes plus one previously unknown type, which can be approximated by the detection/approximation observer. Once the change has been detected, the bank of isolation filters is activated and every filter in the bank is compared with the occurred change. After the change had been either identified or approximated, the control law is modified accordingly in order to counteract its effects.

Because the change presence is not constant in the system, after the change had been accommodated the system continues to monitor its presence. If it is determined that the change is not present any longer or reached insignificant magnitude, the whole CDIA scheme is modified in order to reflect the new conditions.

1.5 Outline of the Thesis

Chapter 2 describes the dynamic model of the robotic system and of the changes. The general framework of the proposed scheme is studied in Chapter 3. This chapter thoroughly investigates every stage of CDIA. Simulation studies are presented in Chapter 4, which is followed by the final Chapter 5. This chapter summarizes this thesis and sets up directions for future work.

2 DYNAMIC MODELS

The mathematical models are the essential elements of the CDIA design. Initially, this chapter presents the well-studied dynamic structure of the robotic system. The second part of this chapter, concentrates on the dynamical structure, configuration and nomenclature of the changes (faults) in the robotic system. Innovations like parametric change history profiling and decoupled torque-dependent and state-dependent change model are introduced and thoroughly analyzed.

2.1 Robotic System

The dynamic motion of the manipulator arm in a robotic system is produced by the torques generated by the actuators. This relationship between the input torques and the time rates of change of the robot arm components configurations, represent the dynamicmodel of the robotic system [1]. This thesis analyses an *n*-degree of freedom

robot configuration. Figure 4 provides a pictorial representation of a SCARA robotic system, which is analyzed in detail in Chapter 4.

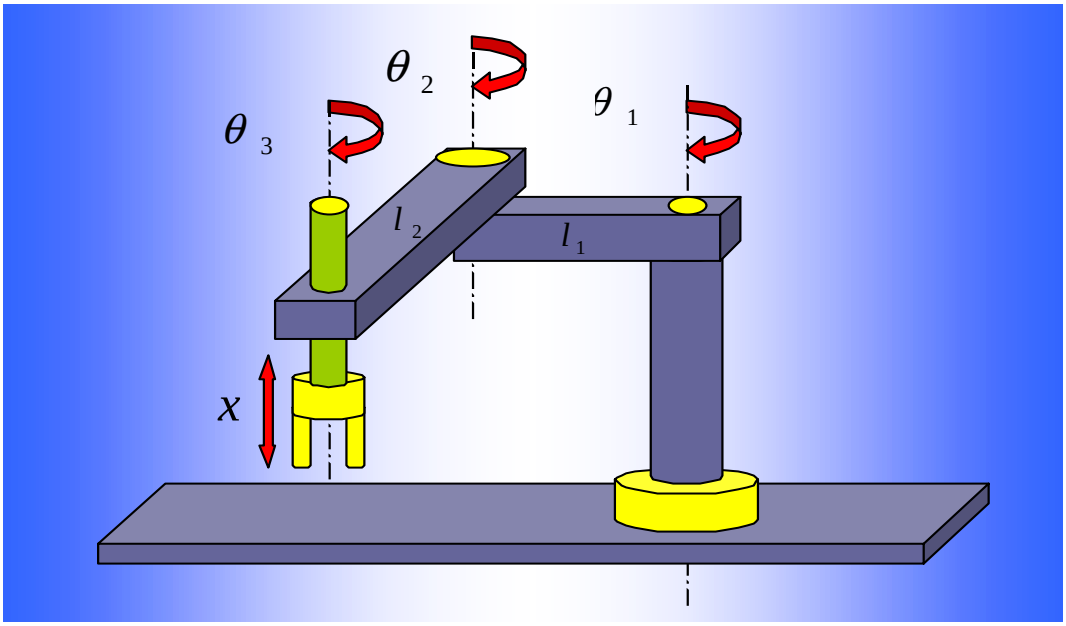


Figure 4 - SCARA robot diagram.

The dynamic model of the robotic system can be derived using either Lagrangian, or Newton-Euler methods [1]. Both methods lead to the identical system of differential equations, which have been extensively studied in the literature on robots [1][2][4]. A general healthy *n*-degree of freedom robotic system is described by the following system of differential equations:

$$M(\theta)\,\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) + \eta\,(\theta,\dot{\theta},\tau,t) = \tau \quad , \tag{2.1}$$

where $\theta,\dot{\theta},\ddot{\theta} \in R^n$ denote the vectors of joint positions, velocities, and accelerations, respectively, $\tau \in R^n$ is the vector of input torques, $G(\theta) \in R^n$ is the vector of gravitational torque, $V(\theta,\dot{\theta}) \in R^n$ is the vector representing Coriolis and centripetal forces, $M(\theta) \in R^{n\times n}$ is the inertia matrix whose inverse exists, and $\eta(\theta,\dot{\theta},\tau,t) \in R^n$ denotes the unmodeled dynamics. It is assumed that the unmodeled dynamics are bounded. It is impossible to achieve a mathematical model that is a perfect mirror image of the actual physical system. Unmodeled dynamics are always present; therefore they have to be considered in the CDIA design in order to reflect the true dynamics of the system.

The presented model was derived for robotic systems operating in the air environment. It can be modified to describe robotic systems operating in the underwater environment or any other environments by including additional dynamics for instance drag, turbulence, etc.

2.2 Changes

Any additional dynamics, which were not present initially in the system, are considered to be a change (fault). This section reviews changes (faults) in a robotic system, categorizes them and introduces an innovative change modeling approach. Throughout this thesis the terms *change* and *fault* are being used interchangeably. The term *fault* falls under the definition of the change, and it has been extensively used in the control and robotics communities referring specifically to the undesirable changes in the internal dynamics. The term change is a more general term, which includes both internal and external variations of the changes, and does not concentrate only on undesirable

changes or faults. The use of the term *change* may bring additional confusions with regards to its definition. Because the term fault is so widely accepted in the control and robotics communities, it will be therefore used in the rest of this thesis.

2.2.1 Fault Magnitudes and Categories

There are faults, which are referred to as *catastrophic*. Catastrophic faults affect the system in such a way that it cannot function any further, and any ordinary control techniques cannot counteract their effects. An example of component catastrophic fault is a break of a joint or a link section. An example of actuator catastrophic fault is a short circuit in electric motor, permanently damaging the wiring. This type of faults is the worse case fault scenario and its effects on the system are obviously devastating. The only way they can be corrected is by direct operator (human) involvement and replacement of the system components. This thesis concentrates only on the faults of smaller magnitudes, or *non-catastrophic*, which can be accommodated with ordinary control techniques. This type of faults includes different variations of friction, misbalances in the joint or actuator, water/air external environment switch and many more. These faults can significantly affects the system's performance as well, which can be expressed in the loss of productivity, reduced life expectancy of the system, and unsafe environment for people and outside environment.

Faults can be separated into two distinct categories: those that change the nonlinear dynamics of the nominal model, and those that do not. The second category depends only on time, and not on the states or the inputs, and therefore can be modeled as *additive*. There are very effective techniques that can accommodate such faults, which include

robust control and adaptive control. Faults belonging to the first category have nonlinear dynamics and are beyond of the capabilities of the conventional techniques. They are more difficult to handle because they depend both on the system's states and the inputs. The purpose of this is to device a very effective method that specifically deals with the state and input dependent faults, while being robust with respect to the unmodeled dynamics.

2.2.2 Parametric Fault History

It is reasonable to assume that faults are not continuously present in the system and emerge only after the system has been in operation for some time or once one of the system parameters exceeded a certain threshold value. Coulomb/Sticktion friction is present in the system only at low velocities [23][24][25], and as the joint velocity exceeds a certain velocity value, it approaches zero. Therefore, it would be incorrect to apply a conventional approach and use time to express the fault history profile of Coulomb/Sticktion friction, since only velocity affects its presence. Viscous friction has significant effects only after the system had been in operation for certain time. Even then, its effects are significant only after the velocity exceeds a certain value. In this case, both time and velocity govern its behavior.

Thus, the presence of faults is *both* state and time dependent, and their presence and magnitude is affected by a number of parameters. A general representation of the fault dynamics is taken to be

$$F\left(\theta,\dot{\theta},\tau,t\right)=B(P-p)\,f\left(\theta,\dot{\theta},\tau\right), \tag{2.2}$$

where $f(\theta,\dot{\theta},\tau)$: $R^n \times R^n \times R^+ \rightarrow R^n$ denotes the fault dynamics, and $B(P-p) \in R^{n \times n}$ represents the state and/or time dependent *fault profile* that has the following structure

$$B(P-p) = diag\left[\ \beta_1(P_1-p_1), \beta_2(P_2-p_2), \ldots \beta_n(P_n-p_n)\ \right],$$

$$\beta_j(P_j-p_j) = \begin{cases} 1 \ if \ \ p_j \in P_j \\ 0 \ otherwise \end{cases},$$

where $\beta_j(P_j-p_j)$ represents the state and time history of the fault in the j^{th} state, p_j is some parameter (for example time, or velocity), and P_j is a region in this parameter history where the fault is present. *The instance of the fault is declared when the value of the* p_j *traverses into the* P_j *region*. *diag( )* denotes a matrix whose diagonal elements are the entries of the vector included in the brackets.

The combined state and time dependent approach to model the fault history profile has advantages over the traditional approaches which model fault history profile as only time dependent [17][18][20][21][26][28]. The time history profiling is a special case in the parametric history profiling general framework. This approach provides a more accurate mathematical representation of a real fault phenomenon. With this approach, if the history profile of the fault can be learned by the monitoring system, faults can be avoided by staying away from the regions of the profile (P_j) where it is present. In addition, [21] and similar treatments consider parameters only to effect the dynamics of the fault, the history of the fault is separated and is only time dependent. By

treating fault dynamics and the history as two dependent entities, a more complete and thorough fault model is shaped.

2.2.3 Fault Nomenclature

Decades of research on the mechanical systems unveiled a major portion of the common faults. Many of them have been extensively studied and well modeled. This knowledge is used to improve the design of the new mechanical systems, and it can be employed in the design of the control systems and the CDIA in particular.

Let us assume that there are *N* types of a priori known faults, which may appear in the system, or a set of a priori known faults is

$$\mathrm{A} = \left\{ \; f_1(\theta, \dot{\theta}, \tau), \; \ldots \; , f_N(\theta, \dot{\theta}, \tau) \; \right\} .$$

Faults may occur one at a time, in multiple concurrent combinations, or in multiple asynchronous combinations. This leads to a conclusion that for *N* types of a priori known faults there are $2^N - 1$ possible concurrent combinations (the combination with no faults present is excluded), or a collection of instances of all a priori known faults can be described by

$$\mathrm{P(A)} = \left\{ \; f_1(\theta, \dot{\theta}, \tau), \; \ldots \; , f_{2^N-1}(\theta, \dot{\theta}, \tau) \; \right\} \; \setminus \; \{\varnothing\},$$

where $\mathrm{P}(\mathrm{A})$ is a power set of A. Therefore a complete set of faults in the system is

$$\Phi = \left\{ \; f_0 \left(\theta, \dot{\theta}, \tau\right), \mathrm{P}(\mathrm{A}) \; \right\} \;\; ,$$

where f_0 is an unknown fault type (Section 3.1).

2.2.4 Fault Dynamics

Each fault is assumed to be linearly parameterized, which can be expressed in the following form

$$f_m\left(\theta,\dot{\theta},\tau\right) = \begin{bmatrix} \sum_{i=1}^{s} c_{1_i}^m \; w_{1_i}^m\left(\theta_1,\dot{\theta}_1,\tau_n\right) \\ \sum_{i=1}^{s} c_{2_i}^m \; w_{2_i}^m\left(\theta_2,\dot{\theta}_2,\tau_n\right) \\ \vdots \\ \sum_{i=1}^{s} c_{n_i}^m \; w_{n_i}^m\left(\theta_n,\dot{\theta}_n,\tau_n\right) \end{bmatrix} = \sum_{i=1}^{s} \begin{bmatrix} c_{1_i}^m \; w_{1_i}^m \; \left(\theta_1,\dot{\theta}_1,\tau_1\right) \\ c_{2_i}^m \; w_{2_i}^m \; \left(\theta_2,\dot{\theta}_2,\tau_2\right) \\ \vdots \\ c_{n_i}^m \; w_{n_i}^m \; \left(\theta_n,\dot{\theta}_n,\tau_n\right) \end{bmatrix}$$

$$= \sum_{i=1}^{s} \begin{bmatrix} c_{1_i}^m & 0 & . & . & 0 \\ 0 & c_{2_i}^m & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & . & . & c_{n_i}^m \end{bmatrix} \begin{bmatrix} w_{1_i}^m\left(\theta_1,\dot{\theta}_1,\tau_1\right) \\ w_{2_i}^m\left(\theta_2,\dot{\theta}_2,\tau_2\right) \\ \vdots \\ w_{3_i}^m\left(\theta_n,\dot{\theta}_n,\tau_n\right) \end{bmatrix}$$

$$= \sum_{i=1}^{s} diag\left[C_{m_i}\right] W_{m_i}\left(\theta,\dot{\theta},\tau\right), \qquad \text{for } m = 1, 2, \ldots, 2^N - 1, \qquad (2.3)$$

where $C_{m_i} \in R^n$ is a vector of the weights or parameters and $W_{m_i} : R^n \times R^n \times R^+ \rightarrow R^n$ is a vector of dynamic functions. Consequently, the dynamics of a fault $m \in [\, N+1,\ 2^N - 1\,]$ are cumulative dynamics of a combination consisting of $v \in [\, 1,\ N\,]$ types of concurrent faults:

$$f_m(\theta, \dot{\theta}, \tau) = \sum f_{i \in v} = f_{a \in v} + f_{b \in v} + \ldots + f_{g \in v} =$$

$$= \sum_{i=1}^{s_{a \in v}} diag\left[C_{a_i \in v}\right] W_{a_i \in v} + \sum_{i=1}^{s_{b \in v}} diag\left[C_{b_i \in v}\right] W_{b_i \in v} + \ldots + \sum_{i=1}^{s_{v \in v}} diag\left[C_{g_i \in v}\right] W_{g_i \in v} =$$

$$= \sum_{i=1}^{s} diag\left[C_{m_i}\right] W_{m_i}(\theta, \dot{\theta}, \tau)\,, \qquad \text{for } m = 1,2, \ldots, 2^N - 1\,.$$

Hence, throughout this thesis any reference to a fault pointing to a complete fault dynamics, but not to any specific type of fault dynamics.

2.2.5 State and Torque-dependent Faults

It is important to be able to differentiate between torque and state-dependent faults. It corresponds to a more comprehensive fault models and in turn allows the CDIA to separately pinpoint faults related to the actuator or the component. It is the fact that the faults might occur either in the actuator or the joint or in both at the same time (Figure 5).

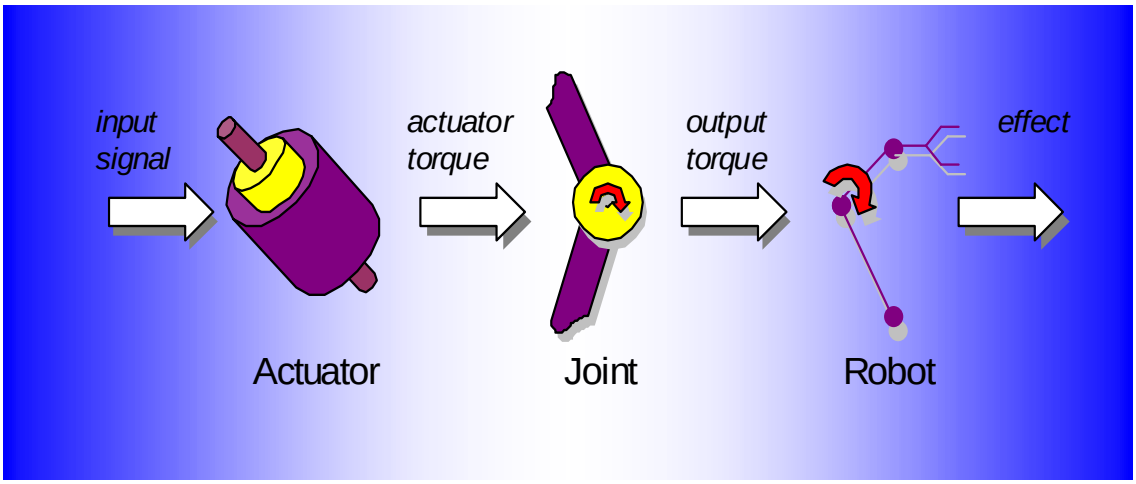


Figure 5 - Input signal – output effect diagram.

Therefore, the fault dynamics can be represented as

$$f_m(\theta,\dot{\theta},\tau) = f_{m_\theta}(\theta,\dot{\theta}) + f_{m_\tau}(\tau),$$

where $f_{m_\tau}(\tau)$ and $f_{m_\theta}(\theta,\dot{\theta})$ represent *torque-dependent* and *state-dependent* faults respectively.

This thesis treats state and torque-dependent faults as two separate entities. In a similar treatment in [20], torque is assumed to be a function of input states only and thus the possibility of faults in the actuator are not explicitly presented. The work in [18][22] does consider faults due to input torque but does not separate (decouple) actuator faults from component faults and treat them as one entity. Adaptation in this case is structured around states only. These approaches eliminate the possibility of separate actuator and component fault isolation.

2.2.6 Summary

Summarizing this section's analysis of the faults in the robotic system, we arrive at the following comprehensive model of the robotic system:

$$\underbrace{M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) + \eta(\theta,\dot{\theta})}_{\text{Robotic System's Dynamics}} = \underbrace{\tau}_{\substack{\text{Input} \\ \text{Torque}}} + \underbrace{B_m(P-p)\left[f_{m\theta}(\theta,\dot{\theta}) + f_{m_\tau}(\tau)\right]}_{\text{Fault Dynamics}} \quad (2.4)$$

3 CDIA ARCHITECTURE

In this chapter CDIA's, architecture is analyzed in details. Figure 3 offers a graphic representation of its architecture. CDIA consists of four distinct stages: (i) *detection*, (ii) *isolation*, (iii) *accommodation*, and (iv) *idle-monitoring*, each of which are described in sections 3.2, 3.4, 3.5, 3.6, and 3.7 respectively. Sections 3.1 and 3.3 discuss and analyze vital mechanisms in the CDIA design that make it robust and effective. Imbedded in the gray regions are the key results of the analysis in each section. These results can be used as guidelines for the implementation of the CDIA scheme.

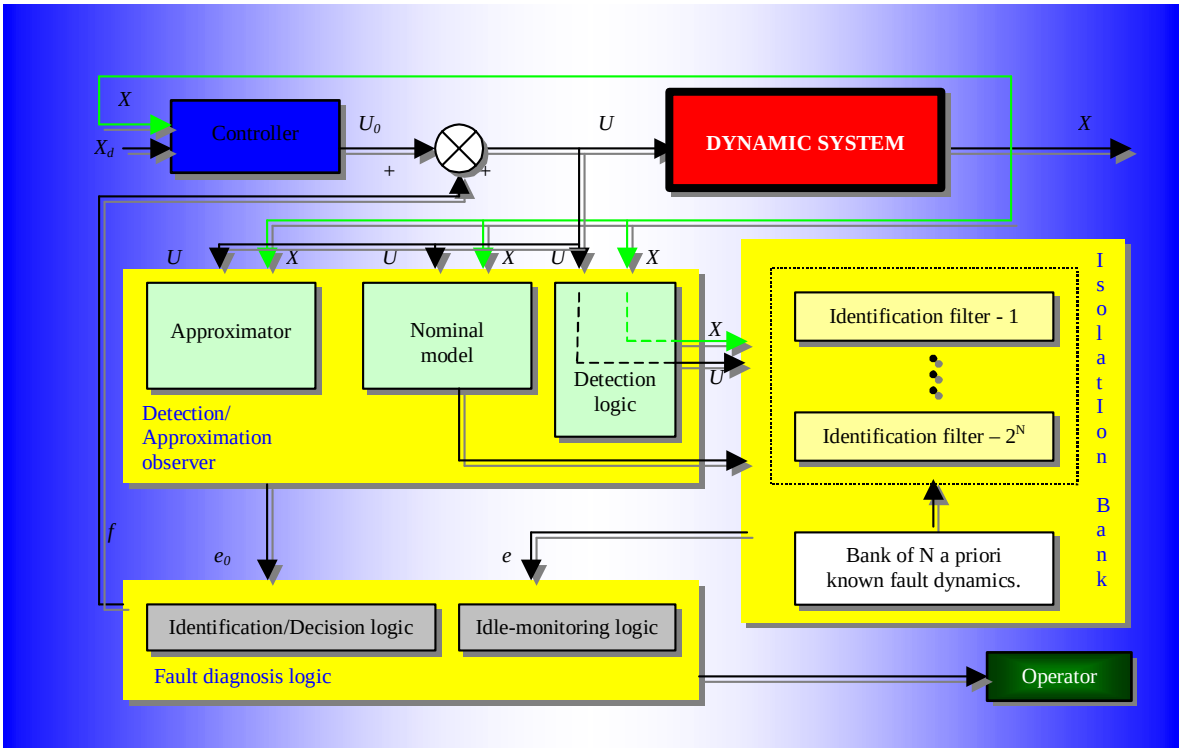


Figure 6 - CDIA architecture.

The CDIA scheme can be made as sophisticated and complicated, as the designer would need, depending on the system, hardware, and other requirements. Each of the CDIA stages is the building block of the scheme, where the order of their implementation has to be preserved. Each instance of the fault may require dedication of a separate CDIA process, consisting of the isolation, accommodation and idle-monitoring stages.

Therefore, the duration and the number of the separate processes are very dynamic. In the present design of the CDIA, a very important assumption is made which is based on both the analytical and the hardware capabilities of the system. It is assumed, that the presented scheme is fast enough to detect and isolate any fault combination set before the next set may occur. With this assumption, the analyzed schemes for a single

set of multiple faults can be applied to the multiple random fault situations without any modifications.

3.1 Detection/Approximation Observer

The detection/approximation observer is a multifunction mechanism that bonds the entire CDIA scheme together. While the system is healthy it is used to monitor it for faults and detect them if they do occur. During the subsequent stages, it is used to approximate and accommodate unknown fault dynamics, and to monitor the system for fault absence. Each of the detection/approximation observer application becomes evident in later sections. It is carefully designed to be robust with respect to unmodeled dynamics, and state and torque-dependent faults.

In section 2.2.4 the parametric structure of the fault dynamics was analyzed. Based on it, the *approximated* torque-dependent and state-dependent fault dynamics in an *n*-degree of freedom system can be represented by the following equations:

$$\hat{f}_{\tau}(\tau,t)=\begin{bmatrix} h_1(t)\ \tau_1 \\ h_2(t)\ \tau_2 \\ . \\ . \\ h_n(t)\ \tau_n \end{bmatrix}=\begin{bmatrix} h_1(t) & 0 & . & . & 0 \\ 0 & h_2(t) & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & . & . & h_n(t) \end{bmatrix}\begin{bmatrix} \tau_1 \\ \tau_2 \\ . \\ . \\ \tau_n \end{bmatrix}=diag[H(t)]\,\tau$$

$$\hat{f}_{\theta}(\theta,\dot{\theta},t)=\begin{bmatrix}\sum_{i=1}^{k} l_{1_i}(t)\, q_{1_i}(\theta_1)\\ \sum_{i=1}^{k} l_{2_i}(t)\, q_{2_i}(\theta_2)\\ \vdots\\ \sum_{i=1}^{k} l_{n_i}(t)\, q_{n_i}(\theta_n)\end{bmatrix}+\begin{bmatrix}\sum_{i=1}^{k} s_{1_i}(t)\, z_{1_i}(\dot{\theta}_1)\\ \sum_{i=1}^{k} s_{2_i}(t)\, z_{2_i}(\dot{\theta}_2)\\ \vdots\\ \sum_{i=1}^{k} s_{n_i}(t)\, z_{n_i}(\dot{\theta}_n)\end{bmatrix} \tag{3.1}$$

$$=\sum_{i=1}^{k}\begin{bmatrix} l_{1_i}(t)\, q_{1_i}(\theta_1)\\ l_{2_i}(t)\, q_{2_i}(\theta_2)\\ \vdots\\ l_{n_i}(t)\, q_{n_i}(\theta_n)\end{bmatrix}+\sum_{i=1}^{k}\begin{bmatrix} s_{1_i}(t)\, z_{1_i}(\dot{\theta}_1)\\ s_{2_i}(t)\, z_{2_i}(\dot{\theta}_2)\\ \vdots\\ s_{n_i}(t)\, z_{n_i}(\dot{\theta}_n)\end{bmatrix}$$

$$=\sum_{i=1}^{k}\begin{bmatrix} l_{1_i}(t) & 0 & \cdots & 0\\ 0 & l_{2_i}(t) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & l_{n_i}(t)\end{bmatrix}\begin{bmatrix} q_{1_i}(\theta_1)\\ q_{2_i}(\theta_2)\\ \vdots\\ q_{n_i}(\theta_n)\end{bmatrix}+\sum_{i=1}^{k}\begin{bmatrix} s_{1_i}(t) & 0 & \cdots & 0\\ 0 & s_{2_i}(t) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & s_{n_i}(t)\end{bmatrix}\begin{bmatrix} z_{1_i}(\dot{\theta}_1)\\ z_{2_i}(\dot{\theta}_2)\\ \vdots\\ z_{n_i}(\dot{\theta}_n)\end{bmatrix}$$

$$=\sum_{i=1}^{k} diag\left[L_i(t)\right] Q_i(\theta)+\sum_{i=1}^{k} diag\left[S_i(t)\right] Z_i(\theta)$$

$$=\sum_{i=1}^{k}\Big(diag\left[L_i(t)\right] Q_i(\theta)+ diag\left[S_i(t)\right]\ Z_i(\theta)\Big),$$

where $H(t)\in R^n$, $L_i(t)\in R^n$, and $S_i(t)\in R^n$ are the vectors of the weights or parameters. In equation (3.1) the velocity and the position dynamics are decoupled for analytical purposes. It does not affect the approximation effort, although it allows detecting the position-dependent faults and the velocity-dependent faults individually. Both velocity-dependent and position-dependent dynamics of the fault are approximated using RBF

neural network structures composing the $Q_i(\theta) \in R^n$ and $Z_i(\dot{\theta}) \in R^n$ vectors, and are structured as follows

$$q_{ij}(\theta_j) = \exp\left(-\frac{(\theta_j - a_{ij})^2}{\sigma_{ij}^{\ 2}} \right),$$

$$z_{ij}(\dot{\theta}_j) = \exp\left(-\frac{(\dot{\theta}_j - b_{ij})^2}{\omega_{ij}^{\ 2}} \right) \qquad for \quad i = 1, 2, \ldots, k \quad and \quad j = 1, 2, \ldots, n,$$

where a_{ij}, b_{ij} are centers of the gaussian networks for position and velocity neurons respectively in the j^{th} state and the i^{th} neuron. Likewise, σ_{ij}, ω_{ij} are widths of the gaussian networks for position and velocity neurons respectively in the j^{th} state and the i^{th} neuron [17][31]. Going along with the same architecture as the approximated fault dynamics above, the *true* state-dependent and torque-dependent fault dynamics (equation (2.3) decoupled) are assumed to have an equivalent form:

$$f_\tau(\tau) \cong \begin{bmatrix} h_1^* \tau_1 \\ h_2^* \tau_2 \\ . \\ . \\ h_n^* \tau_n \end{bmatrix} = \begin{bmatrix} h_1^* & 0 & . & . & 0 \\ 0 & h_2^* & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & . & . & h_n^* \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ . \\ . \\ \tau_n \end{bmatrix} = diag[H^*]\tau, \qquad (3.2)$$

$$f_\theta(\theta,\dot{\theta}) \cong \begin{bmatrix} \sum_{i=1}^{k} l_{1_i}^{*}\ q_{1_i}(\theta_1) \\ \sum_{i=1}^{k} l_{2_i}^{*}\ q_{2_i}(\theta_2) \\ \vdots \\ \sum_{i=1}^{k} l_{n_i}^{*}\ q_{n_i}(\theta_n) \end{bmatrix} + \begin{bmatrix} \sum_{i=1}^{k} s_{1_i}^{*}\ z_{1_i}(\dot{\theta}_1) \\ \sum_{i=1}^{k} s_{2_i}^{*}\ z_{2_i}(\dot{\theta}_2) \\ \vdots \\ \sum_{i=1}^{k} s_{n_i}^{*}\ z_{n_i}(\dot{\theta}_n) \end{bmatrix}$$

$$= \sum_{i=1}^{k} \begin{bmatrix} l_{1_i}^{*}\ q_{1_i}(\theta_1) \\ l_{2_i}^{*}\ q_{2_i}(\theta_2) \\ \vdots \\ l_{n_i}^{*}\ q_{n_i}(\theta_n) \end{bmatrix} + \sum_{i=1}^{k} \begin{bmatrix} s_{1_i}^{*}\ z_{1_i}(\dot{\theta}_1) \\ s_{2_i}^{*}\ z_{2_i}(\dot{\theta}_2) \\ \vdots \\ s_{n_i}^{*}\ z_{n_i}(\dot{\theta}_n) \end{bmatrix}$$

$$= \sum_{i=1}^{k} \begin{bmatrix} l_{1_i}^{*} & 0 & . & . & 0 \\ 0 & l_2^{*} & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & . & . & l_{n_i}^{*} \end{bmatrix} \begin{bmatrix} q_{1_i}(\theta_1) \\ q_{2_i}(\theta_2) \\ \vdots \\ q_{n_i}(\theta_n) \end{bmatrix} + \sum_{i=1}^{k} \begin{bmatrix} s_{1_i}^{*} & 0 & . & . & 0 \\ 0 & s_{2_i}^{*} & . & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & . & . & s_{n_i}^{*} \end{bmatrix} \begin{bmatrix} z_{1_i}(\dot{\theta}_1) \\ z_{2_i}(\dot{\theta}_2) \\ \vdots \\ z_{n_i}(\dot{\theta}_n) \end{bmatrix}$$

$$= \sum_{i=1}^{k} diag\left[L_i^{*}\right] Q_i(\theta) + \sum_{i=1}^{k} diag\left[S_i^{*}\right] Z_i(\dot{\theta})$$

$$= \sum_{i=1}^{k} \left(diag\left[L_i^{*}\right] Q_i(\theta) + diag\left[S_i^{*}\right]\ Z_i(\dot{\theta}) \right), \qquad (3.3)$$

and

$$H^{*} = \begin{bmatrix} h_1^{*} & h_2^{*} & . & . & h_n^{*} \end{bmatrix}^{T}, \qquad h_j^{*} \neq -1 \quad \forall\ j = 1, 2, \ldots, n\ ,$$

where $H^{*} \in R^{n}$, $L_i^{*} \in R^{n}$ and $S_i^{*} \in R^{n}$ represent the weight of the true fault dynamics and are assumed to be constants. In fact, the real values of the weights are never known, but it is assumed that H^{*}, L_i^{*}, and S_i^{*} represent their counterparts that constrain the fault to

exhibit identical behavior. Therefore, $H(t)$, $L_i(t)$ and $S_i(t)$ can be varied in time in order to approximate the values of H^*, L_i^*, and S_i^* respectively. This makes $\hat{f}_\tau$ and $\hat{f}_\theta$ the *on-line approximators* of f_τ and f_θ respectively. Through the next part of the analysis, the notation '$*(*)$' will be replaced with '$*$' for reasons of simplicity. Therefore equation (2.4) can be rewritten as

$$\begin{aligned} & M\ \ddot{\theta} + V + G + \eta = \\ & = \left(I + B\, diag\left[H^*\right] \right)\tau + B\sum_{i=1}^{k}\left(diag\left[L_i^*\right] Q_i + diag\left[S_i^*\right] Z_i \right), \end{aligned} \tag{3.4}$$

In order to find expressions for updating $\hat{f}_\tau$ and $\hat{f}_\theta$, initially $H_i(0)$, $L_i(0)$ and $S_i(0)$ are set so that $\hat{f}_\tau = 0$ and $\hat{f}_\theta = 0$ at $t = 0$. In the state space form, equation (3.4) can be rewritten as

$$\begin{aligned} \ddot{\theta} = & -M^{-1}\left(V + G + \eta \right) + M^{-1}\left(I + B\ diag\left[H^*\right] \right)\tau + \\ & + M^{-1}B\sum_{i=1}^{k}\left(\ diag\left[L_i^*\right] Q_i(\theta) + diag\left[S_i^*\right] Z_i\left(\dot{\theta}\right) \right). \end{aligned} \tag{3.5}$$

Following the above-presented analysis, the detection/approximation observer is proposed

$$\ddot{\hat{\theta}} = -M^{-1}\left(V + G \right) + M^{-1}\left(I + diag[H] \right) \tau + \\ + M^{-1}\sum_{i=1}^{k}\left(diag[L_i] Q_i + diag[S_i] Z_i \right) - \gamma\left(\dot{\hat{\theta}} - \dot{\theta} \right) \tag{3.6}$$

where

$$\gamma = diag[\gamma_1 \ \gamma_2 \ \ldots \gamma_n]$$

is a positive definite stability matrix [1][2].

3.2 Detection

Let $e_0 = \dot{\hat{\theta}} - \dot{\theta}$ denote the state estimation error, which will serve also as the *residual vector* [2][17]. During the detection stage, the CDIA monitors the system for the presence of the faults. While the system is healthy or no fault is present, the true system dynamics is represented as follows

$$\ddot{\theta} = -M^{-1}\left(V + G \right) + M^{-1} \tau \tag{3.7}$$

Unmodeled dynamics η is excluded from the equation. Its presence will be addressed in details in the next section. Consequently, while the system is healthy, the approximation model has the following form

$$\ddot{\hat{\theta}} = -M^{-1}\left(\; V + G \; \right) + M^{-1}\,\tau - \gamma\left(\dot{\hat{\theta}} - \dot{\theta}\right) \; . \qquad (3.8)$$

Therefore, by calculating the estimation error e_0 from equations (3.7) and (3.8), it is established that *for a fully observable system, while it is healthy, or there is no fault present the estimation error must be zero or* $\boxed{e_0(t) = 0}$. This consequently means that $H^* = 0$, $L_i^* = 0$, and $S_i^* = 0$. The estimates of the faults are also set to zero ($H = 0$, $L_i = 0$, $S_i = 0$) in order to detect any difference between the nominal dynamics in the detection/approximation observer and the real system.

As a result of the above analysis, *if* $\boxed{e_0(t) \neq 0}$, $\boxed{p_i \in P_i}$ *and the additional dynamics are present in the system and a fault is* ***declared***. By subtracting the estimated model (3.8) from the true model (3.5) of unhealthy system, we obtain

$$\dot{e}_0 = -\gamma e_0 - M^{-1}\left[B\; diag\left[H^*\right]\tau + B\sum_{i=1}^{k}\left(L_i^*\; diag\left[Q_i\right] + S_i^*\; diag\left[Z_i\right] \right) - \eta \right]. \qquad (3.9)$$

The detection/approximation observer will be in detection mode until the residual vector exceeds the dynamic detection threshold at time t_{dt} analyzed below. t_{dt} is the point in the time history when the fault was detected.

3.3 Dynamic Detection Threshold

The unmodeled dynamics η are always present in the system, and can be mistakenly identified by the CDIA as a fault. In order to avoid such *false alarms* and to improve performance, a *detection threshold* is introduced. Prior to the fault occurrence, from the equation (3.9) the error equation is given by

$$\dot{e}_0 = -\gamma e_0 + M^{-1}\eta .$$

$$\Rightarrow \quad e_0(t) = \exp(-\gamma e_0) e_0(0) + \int_0^t \exp(-\gamma(t-T))\, M^{-1}(T)\, \eta(T)\, dT \qquad (3.10)$$

By introducing the upper bound on each element of η, given by $(\eta_0)_j = \sup_t |\eta_j|$, and taking into account that $e_0(0) = 0$, we arrive at

$$\Rightarrow \quad e_0(t) \le \int_0^t \exp(-\gamma(t-T)) \left[\, M^{-1}(T)\, \eta_0 \,\right] dT$$

Define the detection threshold vector $D(\theta, \dot{\theta}, t) = [d_1, d_2, \ldots, d_n]$ to be

$$D(\theta, \dot{\theta}, t) \equiv \int_0^t \exp(-\gamma(t-T)) \left[\, M^{-1}(T)\, \eta_0 \,\right] dT \; .$$

Therefore, *a fault is declared if*

$$\left| e_{0_j} \right| > d_j(t) \qquad for \quad j = 1, 2, \ . \ . \ . \ , n$$,

(once any element of the residual vector exceeds the corresponding element of the detection threshold). Detection delay can be observed on the plot below. Because the detection threshold is dynamic, detection delay is small in comparison with the isolation time.

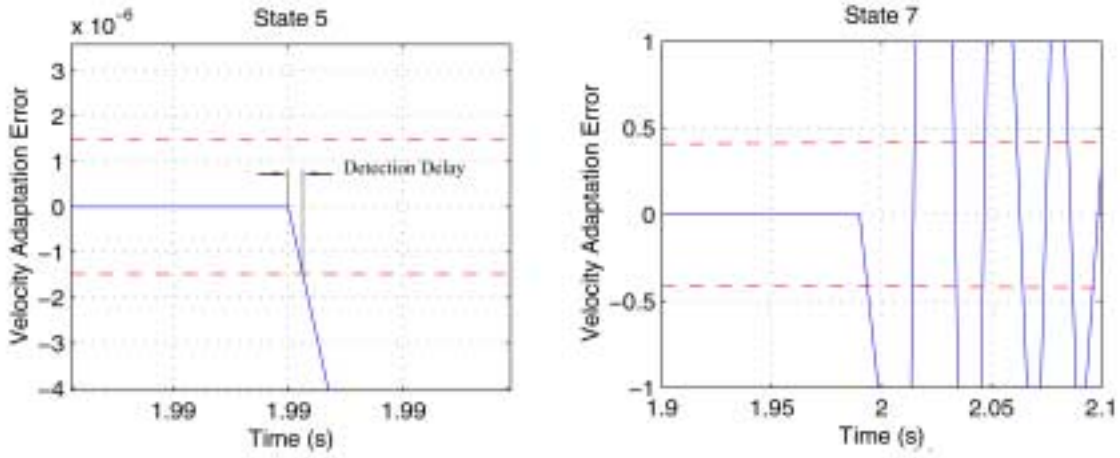


Figure 7 - Detection delay;
(_ _ _ threshold, _____ velocity approximation error).

The detection threshold is dynamic with respect to both time and the states. Such design minimizes the detection time and brings additional advantages. It gives the system the ability to distinctly determine the point in the fault history profile where the fault emerges. Until such point, the systems approximation efforts are put on hold, therefore preserving system's computational resources.

Only time-varying version of the dynamic detection threshold is also derived. By introducing the upper bound, given by $(\overline{\eta}_0)_j = \sup_t \left| M^{-1}\eta \right|_j$, and taking into account that $e_0(0) = 0$, from equation (3.10), we arrive at

$$e_9 \leq \left[\mathrm{I} - \exp(-\gamma t) \right] \gamma^{-1} \overline{\eta}_0$$

Define the time-varying detection threshold vector $D_t(\theta, \dot{\theta}, t) = \left[d_1^t, d_2^t, \ldots, d_n^t \right]$ to be

$$\boxed{D_t(t) \equiv \left[\mathrm{I} - \exp(-\gamma t) \right] \gamma^{-1} \overline{\eta}_0} .$$

It minimizes the detection time, but only during the initial stages of the operation. The plot below depicts its performance

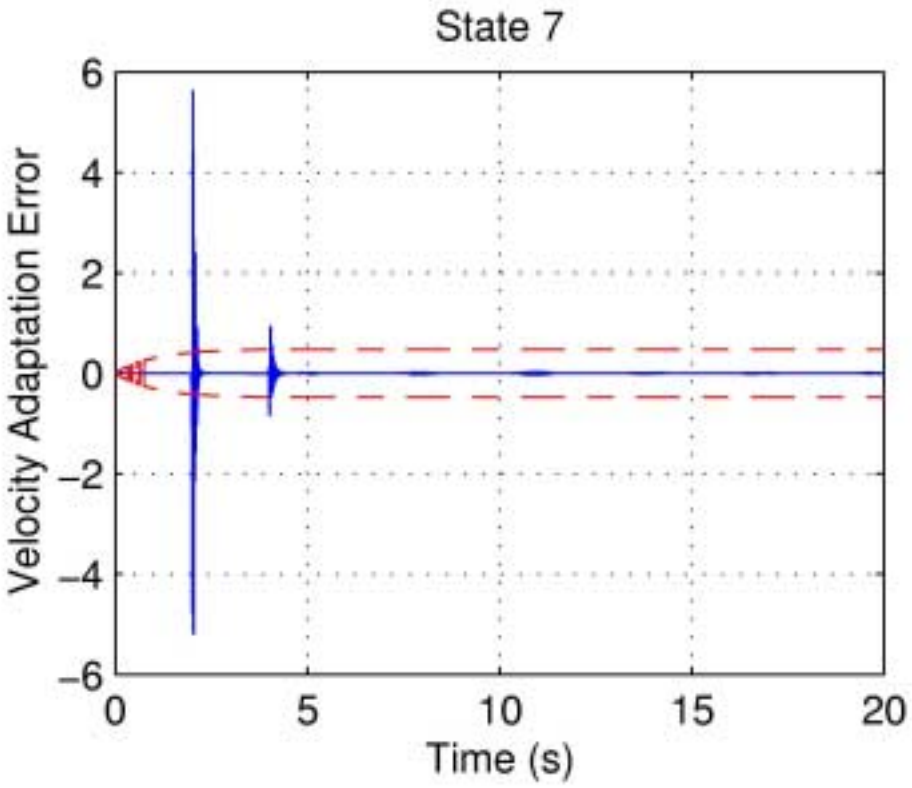


Figure 8 - Time-varying detection threshold performance;
(_ _ _ threshold, _____ velocity approximation error).

3.4 Approximation

This thesis considers only abrupt faults; therefore a fault occurrence implies that $B = \mathrm{I}$. Once the fault has been detected, from equations (3.5) and (3.6), the complete error equation is given by

$$\dot{e}_0 = M^{-1}\tau + M^{-1}\hat{f}_\tau + M^{-1}\hat{f}_\theta - M^{-1}\tau - M^{-1}f_\tau - M^{-1}f_\theta - \gamma e_0$$

$$= -\gamma e_0 + M^{-1} diag[H]\,\tau - M^{-1}\, diag[H^*]\,\tau +$$

$$+ M^{-1}\sum_{i=1}^{k}\left(diag[L_i]\, Q_i + diag[S_i]\, Z_i \right) - M^{-1}\sum_{i=1}^{k}\left(diag[L_i^*]\, Q_i + diag[S_i^*]\, Z_i \right) - \eta$$

$$= -\gamma e_0 + M^{-1}\left[\left(diag[H] - diag[H^*] \right)\tau \right.$$

$$\left. + \sum_{i=1}^{k}\left[\left(diag[L_i] - diag[L_i^*] \right) Q_i + \left(diag[S_i] - diag[S_i^*] \right) Z_i \right] - \eta \right].$$

Let $\tilde{H} = H - H^*$, $\tilde{L}_i = L_i - L^*_i$, and $\tilde{S}_i = S_i - S_i^*$. Consequently one has

$$\dot{e}_0 = -\gamma e_0 + M^{-1}\left[diag[\tilde{H}]\,\tau + \sum_{i=1}^{k}\left(diag[\tilde{L}_i]\, Q_i + diag[\tilde{S}_i]\, Z_i \right) - \eta \right].$$

In accordance to Lyapunov stability theory [2], the global stability of the system is guaranteed if it can be shown that some function *U* is globally positive definite (for $t \neq 0,\ U(t) > 0$), and if its derivative $\dot{U}(t)$ is globally negative definite or semi-definite (for $t \neq 0,\ \dot{U}(t) \leq 0$) [2]. We use the stability analysis to accomplish two goals simultaneously: first to show that the system approximation error does converge to zero, and second to derive adaptation laws that make it to converge to zero. It is being done in a backward way, by assuming that the approximation error can be stable, and using Lyapunov stability analysis to establish rules that force this convergence to zero. If the approximation error does gradually converge to zero, consequently the weight in the

detection/approximation filters will mimic the behavior of the weights in the true fault dynamics. A Lyapunov function of the following form is employed:

$$U = \frac{1}{2} e_0^T e_0 + \frac{1}{2} \tilde{H}^T \Gamma^{-1} \tilde{H} + \frac{1}{2} \sum_{i=1}^{k} \tilde{L}_i^T \Psi^{-1} \tilde{L}_i + \frac{1}{2} \sum_{i=1}^{k} \tilde{S}_i^T \Upsilon^{-1} \tilde{S}_i \geq 0,$$

where $\Gamma, \Psi, \Upsilon \in R^{n \times n}$ are adaptive gain matrices gains. Therefore

$$\begin{aligned}
\dot{U} &= e_0^T \dot{e}_0 + \dot{H}^T \Gamma^{-1} \tilde{H} + \sum_{i=1}^{k} \dot{L}_i^T \Psi^{-1} \tilde{L}_i + \sum_{i=1}^{k} \dot{S}_i^T \Upsilon^{-1} \tilde{S}_i \\
&= -e_0^T \gamma e_0 - e_0^T M^{-1} diag\left[\tilde{H}\right] \tau - e_0^T M^{-1} \sum_{i=1}^{k} diag\left[\tilde{L}_i\right] Q_i - e_0^T M^{-1} \sum_{i=1}^{k} diag\left[\tilde{S}_i\right] Z_i + \\
&\quad + \dot{H}^T \Gamma^{-1} \tilde{H} + \sum_{i=1}^{k} \dot{L}_i^T \Psi^{-1} \tilde{L}_i + \sum_{i=1}^{k} \dot{S}_i^T \Upsilon^{-1} \tilde{S}_i - e_0^T M^{-1} \eta \\
&= -e_0^T \gamma e_0 - e_0^T M^{-1} diag[\tau] \tilde{H} + \dot{H}^T diag[\Gamma]^{-1} \tilde{H} - e_0^T M^{-1} \sum_{i=1}^{k} diag\left[Q_j\right] \tilde{L}_i + \\
&\quad + \sum_{i=1}^{k} \dot{L}_i^T diag[\Psi]^{-1} \tilde{L}_i - e_0^T M^{-1} \sum_{i=1}^{k} diag\left[Z_i\right] \tilde{S}_i + \sum_{i=1}^{k} \dot{S}_i^T diag[\Upsilon]^{-1} \tilde{S}_i - e_0^T M^{-1} \eta \\
&= -e_0^T \gamma e_0 + \left(\dot{H}^T \Gamma^{-1} - e_0^T M^{-1} diag[\tau] \right) \tilde{H} - \sum_{i=1}^{k} e_0^T M^{-1} diag\left[Q_i\right] \tilde{L}_i + \\
&\quad + \sum_{i=1}^{k} \dot{L}_i^T \Psi^{-1} \tilde{L}_i - \sum_{i=1}^{k} e_0^T M^{-1} diag\left[Z_i\right] \tilde{S}_i + \sum_{i=1}^{k} \dot{S}_i^T \Upsilon^{-1} \tilde{S}_i - e_0^T M^{-1} \eta
\end{aligned}$$

$$= -e_0^T \gamma e_0 - e_0^T M^{-1} \eta +$$

$$+ \left(\dot{H}^T \Gamma^{-1} - e_0^T M^{-1} diag[\tau] \right) \tilde{H} +$$

$$+ \sum_{i=1}^{k} \left(\dot{L}_i^T \Psi^{-1} - e_0^T M^{-1} diag[Q_i] \right) \tilde{L}_i +$$

$$+ \sum_{i=1}^{k} \left(\dot{S}_i^T \Upsilon^{-1} - e_0^T M^{-1} diag[Z_i] \right) \tilde{S}_i \ .$$

By setting

$$\dot{H}^T \Gamma^{-1} = e_0^T M^{-1} diag[\tau] \quad or \quad \dot{H} = \Gamma diag[\tau] M^{-1} e_0 \ ,$$

$$\dot{L}_i^T \Psi^{-1} = e_0^T M^{-1} diag[Q_i] \quad or \quad \dot{L}_i = \Psi diag[Q_i] M^{-1} e_0 \ , \qquad for \ \ i = 1, 2, \ldots, k$$

$$\dot{S}_i^T \Upsilon^{-1} = e_0^T M^{-1} diag[Z_i] \quad or \quad \dot{S}_i = \Upsilon diag[Z_i] M^{-1} e_0 \ ,$$

we obtain $\dot{U} = -e^T \gamma e - e^T M^{-1} \eta$. When $\eta = 0$, one acquires

$$\dot{U} = -e^T \gamma e \leq 0 \ ,$$

which is negative semi-definite, and therefore the approximation error will converge to zero. When $\eta \neq 0$, one acquires

$$\dot{U} = -e_0^T \gamma e_0 - e_0^T M^{-1}\eta$$

$$\leq -\lambda_{\min}(\gamma)|e_0|^2 + |e_0|\|M^{-1}\||\eta|$$

$$\leq -\lambda_{\min}(\gamma)|e_0|^2 + |M^{-1}|\left\{\frac{|e_0|^2}{2\mu} + \frac{|\eta|^2}{2}\mu\right\}$$

$$= -\left(\lambda_{\min}(\gamma) - |M^{-1}|\frac{1}{2\mu}\right)|e_0|^2 + |M^{-1}|\frac{\mu}{2}|\eta|^2$$

where $\lambda_{\min}(*)$ denotes the smallest eigenvalue. Choose μ: $\lambda_{\min}(\gamma) > |M^{-1}|\frac{1}{2\mu}$, i.e. $\mu > \frac{|M^{-1}|}{2\lambda_{\min}(\gamma)}$, then $\dot{U} = -\alpha|e_0|^2 + \beta|\eta|^2$. Results of this analysis guarantee the uniform boundedness of the velocity estimation error and the weights in the neural network. Furthermore it leads to the conclusion that the overall system remains stable. Following the previous analysis, the approximation observer's architecture will be

$$
\begin{cases}
\ddot{\hat{\theta}} = -M^{-1}\left(V + G \right) + M^{-1}\left(I + diag[H] \right)\tau + M^{-1}\sum_{i=1}^{k}\left(diag\left[L_i\right]Q_i + diag\left[S_i\right]Z_i \right) - \gamma e_0 \\
\dot{H} = \Gamma\, diag[\tau] M^{-1} e_0 \\
\begin{bmatrix} \dot{L}_1 \\ \dot{L}_2 \\ . \\ . \\ \dot{L}_k \end{bmatrix} = \begin{bmatrix} \Psi\, diag\left[Q_1\right] M^{-1} e_0 \\ \Psi\, diag\left[Q_2\right] M^{-1} e_0 \\ . \\ . \\ \Psi\, diag\left[Q_k\right] M^{-1} e_0 \end{bmatrix} \qquad \begin{bmatrix} Q_1(\theta) \\ Q_2(\theta) \\ . \\ . \\ Q_k(\theta) \end{bmatrix} = \begin{bmatrix} \left[q_{1_1}(\theta) \;\; q_{2_1}(\theta) \;\; . \;\; . \;\; q_{n_1}(\theta)\right]^T \\ \left[q_{1_2}(\theta) \;\; q_{2_2}(\theta) \;\; . \;\; . \;\; q_{n_2}(\theta)\right]^T \\ . \\ . \\ \left[q_{1_k}(\theta) \;\; q_{2_k}(\theta) \;\; . \;\; . \;\; q_{n_k}(\theta)\right]^T \end{bmatrix} \\
\begin{bmatrix} \dot{S}_1 \\ \dot{S}_2 \\ . \\ . \\ \dot{S}_k \end{bmatrix} = \begin{bmatrix} \Upsilon\, diag\left[Z_1\right] M^{-1} e_0 \\ \Upsilon\, diag\left[Z_2\right] M^{-1} e_0 \\ . \\ . \\ \Upsilon\, diag\left[Z_k\right] M^{-1} e_0 \end{bmatrix} \qquad \begin{bmatrix} Z_1(\dot{\theta}) \\ Z_2(\dot{\theta}) \\ . \\ . \\ Z_k(\dot{\theta}) \end{bmatrix} = \begin{bmatrix} \left[z_{1_1}(\dot{\theta}) \;\; z_{2_1}(\dot{\theta}) \;\; . \;\; . \;\; z_{n_1}(\dot{\theta})\right]^T \\ \left[z_{1_2}(\dot{\theta}) \;\; z_{2_2}(\dot{\theta}) \;\; . \;\; . \;\; z_{n_2}(\dot{\theta})\right]^T \\ . \\ . \\ \left[z_{1_k}(\dot{\theta}) \;\; z_{2_k}(\dot{\theta}) \;\; . \;\; . \;\; z_{n_k}(\dot{\theta})\right]^T \end{bmatrix}
\end{cases}
$$

Approximation in the isolation filters is identically structured using the same approximation rules as in the detection/approximation observer.

3.5 Isolation

Once the fault has been detected, the entire bank of isolation filters including the detection/approximation observer is activated, and the detected fault is compared with each filter. If one of the isolation filters is found to be equivalent to the detected fault, the exact nature and the source of the fault become known. Throughout this process, the detection/approximation observer keeps approximating the true fault dynamics just in case none of the filters in the bank is equivalent. After the fault function is extracted either by matching it with one of the filters in the isolation bank or using neural networks

in the approximation observer, it can be used to reconfigure the control input and accomplish fault accommodation.

It is preferable to accommodate the system using the fault dynamics extracted from one of the isolation filters. Let us consider the situation when the fault is found to be equivalent to one of the isolation filters, excluding the detection/approximation observer. It takes a certain isolation time t_{is} after the detection time t_{dt} to determine which fault had occurred. At this point, the weights of the isolation filter are adjusted to mimic the actual fault function. After t_{is} accommodation is based on the precisely known fault function, and therefore it requires minimal adaptation activity, and the approximation error is kept at minimum. Most importantly, the operator and the system will have the knowledge of the magnitude and the nature of the fault.

In situations when the approximation observer is used to extract the fault function, the neural networks in approximation observer will be active indefinitely past t_{is} for as long as there is the need to accommodate the fault. The exact dynamics and the nature of the fault will never be known. In addition, it is not known whether the detected fault is just one type of fault or a combination of many faults.

An isolation time t_{is} is not a set quantity and it is different for each isolation effort. Initially it should be set to the predetermined minimal value $t_{is(min)}$. If none of the isolation filters are found to be equivalent on the interval [t_{dt}, t_{is}], then the fault is declared unknown and the detection/approximation observer is used to accommodated it. If more than one fault is found to be equivalent on the interval [t_{dt}, t_{is}], then t_{is} is increased until the true fault dynamics is distinguished from the similar ones on the interval.

The following isolation filter is proposed

$$\ddot{\hat{\theta}}_m = -M^{-1}\left(V + G\ \right) + M^{-1}\sum_{i=1}^{s} diag\left[C_{m_i}\right] W_{m_i} - \gamma\left(\dot{\hat{\theta}}_m - \dot{\theta}\right), \tag{3.11}$$

Therefore the isolation bank would have the following structure:

$$\ddot{\hat{\theta}} = -M^{-1}\left(\ V + G\right) + M^{-1}\left(\ I + diag[H]\ \right)\tau + M^{-1}\sum_{i=1}^{k}\left(\ diag\left[L_i\right]Q_i + diag\left[S_i\right]Z_i\right) - \gamma\, e_0$$

$$\begin{bmatrix} \ddot{\hat{\theta}}_1 \\ \ddot{\hat{\theta}}_2 \\ . \\ . \\ \ddot{\hat{\theta}}_{2^N-1} \end{bmatrix} = \begin{bmatrix} -M^{-1}\left(V + G\ \right) + M^{-1}\sum_{i=1}^{s} diag\left[C_{1_i}\right] W_{1_i} - \gamma\, e_1 \\ -M^{-1}\left(V + G\ \right) + M^{-1}\sum_{i=1}^{s} diag\left[C_{2_i}\right] W_{2_i} - \gamma\, e_2 \\ . \\ . \\ -M^{-1}\left(V + G\ \right) + M^{-1}\sum_{i=1}^{s} diag\left[C_{2^N-1_i}\right] W_{2^N-1_i} - \gamma\, e_{2^N-1} \end{bmatrix}$$

Let $e_m = \dot{\hat{\theta}}_m - \dot{\theta}$ denote the state estimation error in the m^{th} filter. After the fault occurrence, by subtracting the approximated dynamics in the m^{th} filter (3.11) from the true dynamics (3.5), the error equation is given by

$$\dot{e}_m = -\gamma e_m + M^{-1}\eta + M^{-1}\mu = -\gamma e_m + M^{-1}\left(\ \eta + \mu_m\ \right), \tag{3.12}$$

where

$$\mu_m = \sum_{i=0}^{s} diag\left[C_{m_i}^{*}\right] W_{m_i} - \sum_{i=0}^{s} diag\left[C_{m_i}\ \right] W_{m_i} = \sum_{i=0}^{s}\left(\ diag\left[C_{m_i}\right] - diag\left[C_{m_i}\ \right]\ \right) W_{m_i}\ ,$$

is the equivalency deviation between the true fault dynamics and the m^{th} isolation filter dynamics. $C^*_{m_i}$ and $W^*_{m_i}$ are the vectors of weights and dynamic functions respectively belonging to the true dynamics. After multiplying both sides of equation (3.12) by $\exp(\gamma t)$ and rearranging it, we obtain

$$\exp(\gamma t)\dot{e}_m + \gamma \exp(\gamma t) e_m = \exp(\gamma t) M^{-1}\left(\eta + \mu_m\right)$$

$$\Rightarrow \quad \frac{d}{dt}\left(\exp(\gamma t)\ e_m\right) = \exp(\gamma t) M^{-1}\left(\eta + \mu_m\right)$$

$$\Rightarrow \quad \int_0^t \frac{d}{dt}\left(\exp(\gamma t)\ e_m\right) dt = \int_0^t \exp(\gamma t) M^{-1}\left(\eta + \mu_m\right) dt\,.$$

By introducing the upper bound, given by $(\overline{\eta}_0)_j = \sup_t \left| M^{-1}\eta \right|_j$, and the *equivalency margin* $(\tilde{\mu}_m)_j = \sup_t \left| M^{-1}\mu_m \right|_j$, we arrive at

$$\int_0^t \frac{d}{dt}\left(\exp(\gamma t)\ e_m\right) dt \le \int_0^t \exp(\gamma t)\, dt \left(\overline{\eta}_0 + \tilde{\mu}_m\right),$$

$$\Rightarrow \quad \exp(\gamma t)\ e_m \le \left[\exp(\gamma t) - \mathrm{I}\right] \gamma^{-1} \left(\overline{\eta}_0 + \tilde{\mu}_m\right),$$

$$\Rightarrow \quad e_m \le \left[\mathrm{I} - \exp(-\gamma t)\right] \gamma^{-1} \left(\overline{\eta}_0 + \tilde{\mu}_m\right).$$

Define the isolation threshold vector $R_m(\theta, \dot{\theta}, \tau, t) = \left[r_{m_1}, r_{m_2}, \ldots, r_{m_n}\right]$ to be

$$R_m(\theta,\dot{\theta},\tau,t) \equiv \left[\ \mathrm{I} - \exp(-\gamma t)\ \right] \gamma^{-1} \left(\overline{\eta}_0 + \tilde{\mu}_m \right) .$$

Therefore, *for* $\boldsymbol{t_{dt} < t_{is}}$*, dynamics in the filter* $\boldsymbol{m}$ *in the* $\boldsymbol{j}^{th}$ *state are equivalent to the true dynamics within a margin* $\tilde{\mu}_{m_j}$ *if*

$$\left| e_{m_j} \right| \leq r_{m_j} \quad for\ \forall\ t \in [\ t_{dt}, t_{is}\ ] .$$

The above formulation provides a robust mechanism for successful fault isolation. In the absence of an acceptable equivalent, detection/approximation filter should be employed to accommodate the fault.

3.6 Accommodation

In the absence of faults, without any loss of generality a *PD-computed-torque* approach can be used to accomplish tracking [2]. Under healthy conditions, the nominal input torque $\tau = \tau_0$ is given by

$$\tau_0 = \underbrace{M(\theta)\ \left[K_p(\theta - \theta_d) + K_v(\dot{\theta} - \dot{\theta}_d) + \ddot{\theta}_d \right]}_{tracking} + \underbrace{V(\theta,\dot{\theta}) + G(\theta)}_{computed\ \ torque} ,$$

where $\theta_d, \dot{\theta}_d, \ddot{\theta}_d \in R^n$ are the vectors of desired joint positions, velocities, and accelerations, respectively, and $K_p \in R^{n\times n}$ and $K_v \in R^{n\times n}$ are negative definite matrices, which are designed, so that exponential convergence of the tracking errors is achieved.

Applying the proposed torque and stage-dependent fault models, the input should have the following structure

$$\tau = \begin{cases} \left(I + diag[H(t)]\right)^{-1}\left[\tau_0 - \hat{f}_\theta(\theta,\dot{\theta},t)\right] & if \ |1+h_i(t)| > \varepsilon \\ \tau_0 & if \ |1+h_i(t)| \le \varepsilon \end{cases},$$

where ε is some constant, whose value is dictated by the nominal input torque. The new input has capabilities to *self-correct* failures. The fault approximator will be able to mimic the faults and provide appropriate modifications to the input torque in order to accommodate them.

3.7 Idle-Monitoring

After a fault had been accommodated, in most situations it may disappear after certain period. Velocity and position-dependent faults may disappear from the system because the velocity or position reached regions where the fault is simply not present. There can be a multiple of other causes for a fault to become absent from the system. There is no need to spend resources on accommodation of something that is not present

anymore, plus there is no need to keep the system thinking that the fault is there, if in reality it is not there. This fact suggests a need for idle-monitoring system after the fault had been accommodated. It should be able to make a determination if the fault is just at low values or disappeared. If it did disappear, it should change the control, detection, and isolation scheme in order to monitor for its future occurrences. This can be accomplished by introducing *idle-monitoring* threshold $\rho \in R^n$. *The accommodated fault is declared absent, if*

$$\left| \sum_{i=1}^{s} c_{j_i}^{m} \right| < \rho_j \qquad for \quad t \geq t_{pr} \quad and \quad j = 1, 2, \ldots, n \,,$$

where $\boldsymbol{t_{pr}}$ *is the maximum idle time*. Once this happens, the control law is reconfigured so this fault is not accommodated any further, and the bank of isolation filters is updated so it includes this fault dynamics again (isolation it was removed from the isolation bank).

3.8 CDIA Performance Analysis

The performance of CDIA can be optimized with additional modifications. Some of them are described in this section.

As it has been presented in section 2.2.3, faults may occur in multiple concurrent combinations. In addition, faults may occur at different points in the change history, or for instance one combination may occur at t = 2 seconds and another at t = 11 seconds. If the most recent fault combination m (where $m = 1, 2, \ldots \left(2^N - 1\right)$) was successfully

isolated, then there is no need to observe for the types of faults that were a part of this combination. They are already present in the system, they were isolated, and trying to observe and isolate them is an unnecessary use of resources. At this stage, the bank of isolation filters should consist of $2^{N-m}-1$ filters.

As it was presented in Section 2.2.2, some faults do not have time dependent history. Their presence can depend on either one of the states, or a number of the states. For instance, some frictions occur only if velocity exceeds a certain value, so as long as the velocity is below some upper bound, this type of fault cannot occur. Consequently, if some fault had been detected in the system before this triggering parameter threshold had been reached, there is no need to activate the isolation filters for such faults. Therefore, the number of isolation filters can be reduced even more, thus reducing the number of the possible faults and increasing the efficiency of the scheme. This is one of the advantages of modeling fault history not as only time dependent, but as parameter-dependent.

4 SIMULATION

In this chapter the previously presented modeling techniques are applied to SCARA robotic system (Figure 9). This simulation study demonstrates that the presented scheme is effective when applied to a real life robotic system. The simulation was conducted using Matlab [50]. The sample of the Matlab code used is available in the Appendix.

4.1 SCARA Robot

The *Selective Compliance Assembly Robot Arm* (SCARA) robot was selected for the simulation studies because of its extensive use in the industry. Figure 4 depicts a general representation of the SCARA robot. This robotic system comes in many different configurations, and the presented configuration reflects its general structure. This system offers a considerable generality for the scheme simulation because it encapsulates both translational and rotational types of joint and its dynamics strongly depend on position,

velocity, acceleration, and time. Traditionally, SCARA robots have one translational vertical axis, two rotational axes that provide motion in the horizontal plane, and usually one additional axis for the tool rotation in the wrist. The overall SCARA robot structure is very rigid in both the vertical and horizontal axes, which allows very smooth and well guided motion of the links. It has the highest speed of any other robot configuration in the industry, which ranges in 2000-5000 mm/s. The repeatability rate is also very high, which explains its high popularity in the manufacturing industry. Successful application of the CDIA to the SCARA robot assures the generality of the modeling and control scheme proposed. Examples of robotic systems belonging to the general class of SCARA robot include the Adept One, the IBM 7545, the Intelledex 440, and the Rhino SCARA [4][51].

Figure 9 - SCARA robot.

The dynamic model of the SCARA robot can be represented with the same system of differential equations as any general robotic system presented in section 2.1, which is

$$M(\theta)\ddot{\theta}+V(\theta,\dot{\theta})+G=\tau, \tag{4.1}$$

where

$$\tau=\begin{bmatrix}\tau_1 & \tau_2 & \tau_3 & F_4\end{bmatrix}^T,$$

$$\theta=\begin{bmatrix}\theta_1 & \theta_2 & \theta_3 & x_4\end{bmatrix}^T,$$

$$G=\begin{bmatrix}0 & 0 & 0 & m_3 g\end{bmatrix}^T,$$

$$M(\theta)=\begin{bmatrix}
\begin{pmatrix}(m_1+m_2+m_3)l_1^2+\\+2(m_2+m_3)l_1l_2\cos(\theta_2)+\\+(m_2+m_3)l_2^2\end{pmatrix} & \begin{pmatrix}(m_2+m_3)l_2^2+\\+(m_2+m_3)l_1l_2\cos(\theta_2)\end{pmatrix} & I_{z3} & 0\\
\begin{pmatrix}(m_2+m_3)l_2^2+\\+(m_2+m_3)l_1l_2\cos(\theta_2)\end{pmatrix} & (m_2+m_3)l_2^2 & I_{z3} & 0\\
I_{z3} & I_{z3} & I_{z3} & 0\\
0 & 0 & 0 & m_3
\end{bmatrix},$$

$$V(\theta,\dot{\theta})=\begin{bmatrix}-(m_2+m_3)l_1 l_2\sin(\theta_2)\dot{\theta}_2\left(\dot{\theta}_2-2\dot{\theta}_1\right)\\(m_2+m_3)l_1 l_2\sin(\theta_2)\dot{\theta}_1^2\\0\\0\end{bmatrix}.$$

Detailed derivation is presented in Appendix 6.1, which is based on general model analyzed in [5]. Values of the parameters used during simulation of the SCARA robot are listed in the Table 1 below. These are reasonable estimates of the real robotic system.

Link Weights:	
m_1	50 kg
m_2	40 kg
m_3	30 kg
Link Dimensions:	
l_1	0.425 m
l_2	0.375 m
l_3 (radius of the shaft)	0.020 m
x_4	0.356 m
Maximum Ranges:	
p_1	$5/6\pi$ rad
p_2	$7/9\pi$ rad
p_3	$3/2\pi$ rad
p_4	0.200 m
Maximum Velocities:	
V_1	$10/3\pi$ rad/sec
V_2	5π rad/sec
V_3	$55/3\pi$ rad/sec
V_4	1.200 m

Table 1 - SCARA parameters

The presented model is an idealized representation of the real physical system. The following assumptions had been made: no friction, rigid links, rigid structure of the joints (rigid motor shafts, no backslashes, rigid gearing), no load at the end of the effecter, link masses are at distant ends, gravity is *g*, fault free operating conditions. This

model can be improved, which can lead to a better controller design. On the other hand, a more thorough model will have more complicated mathematical structure, which can make its analysis and controller design very difficult or even impossible.

In the SCARA dynamic model state 4 is decoupled from the other three states. One might ask why does it even have to be considered? If state 4 is ignored in the design of the CDIA and the fault does occur specifically in the state 4, then it will never be detected and accommodated for that matter. In addition, a complete model of the SCARA robot is being analyzed in this simulation. Ignoring either one of the states sets it apart from the true mechanical system, and we want the simulation to be as realistic as possible.

The best approach to determine the upper bound of unmodeled dynamics is through experimental study. Because this thesis includes only simulation study and no testing in the field was conducted, it had to be derived in an analytical fashion. It was established that joint velocities exert the largest effect on the magnitude of the unmodeled dynamics. Though, the maximum allowable by robot design join velocities were used as a base for the unmodeled dynamics upper bound vector multiplied by some factor. Running simulations and observing the newly designed upper bound verses the unmodeled dynamics helped to carefully adjust both the multiplication factor, and each value in the upper bound vector.

4.2 Fault Models

In the joints (components), the most common and ever present type of faults is friction. Friction has been extensively analyzed and varieties of models are available.

Friction models in the works by C. Canudas de Wit [23][24][25] provide an excellent reflection of friction in the real joint. The table below lists most common and noteworthy friction models.

Coulomb / Sticktion	$f(\dot{\theta}) = \alpha \operatorname{sgn}(\dot{\theta})$
Asymmetries	$f(\dot{\theta}) = \alpha_j \operatorname{sgn}(\dot{\theta}) + \beta_j \dot{\theta}$
Position Dependence	$f(\theta) = k_f \sin\left(w_0 \theta + \varphi \right)$
Downward Bend	$f(\dot{\theta}) = \left[\alpha_0 + \alpha_1 \exp\left(-\beta \left| \dot{\theta} \right| \right) \right] \operatorname{sgn}(\dot{\theta})$
Viscous	$f(\dot{\theta}) = \alpha_2 \dot{\theta}$

Table 2 - Component Fault Dynamics.

In SCARA manipulators, actuators are generally electric motors. Faults in rotating electric motors may be classified as electric faults, rotational faults and vibration faults. Rotational faults include windage, friction, brush friction, core faults, stray-load faults. Table 3 reflects most of the rotational faults in the motor. The electric faults in motors include the I^2R faults in the field circuits and armature circuits [8][9], and their mathematical model can be summarized with

Electric	$f(\tau) = \alpha\tau, \quad -1 < \alpha \leq K \leq \infty$

Table 3 - Actuator Fault Dynamics.

where *K* is some maximum value that α can reach. The class of vibration faults includes sub-synchronous, synchronous, and super-synchronous faults, vertical motor bearing faults, and critical speeds faults [8][9]. Because of the shear complexity of such faults, there are no adequate mathematical models available and the best available method for their determination is experimental measurements.

4.3 Numerical Study

The first stage of the numerical study analyzes performance of the detection/approximation (DA) observer. Figure 10 - Figure 15 demonstrate results of such study with an example of actuator and component fault detection and accommodation in a SCARA robot. The previously described fault dynamics are applied in this simulation. As shown in Figure 14 and other plots, the proposed scheme is able to detect *both* actuator and component faults, learn their dynamics and make appropriate modifications to the control law, which in turn accomplishes accommodation.

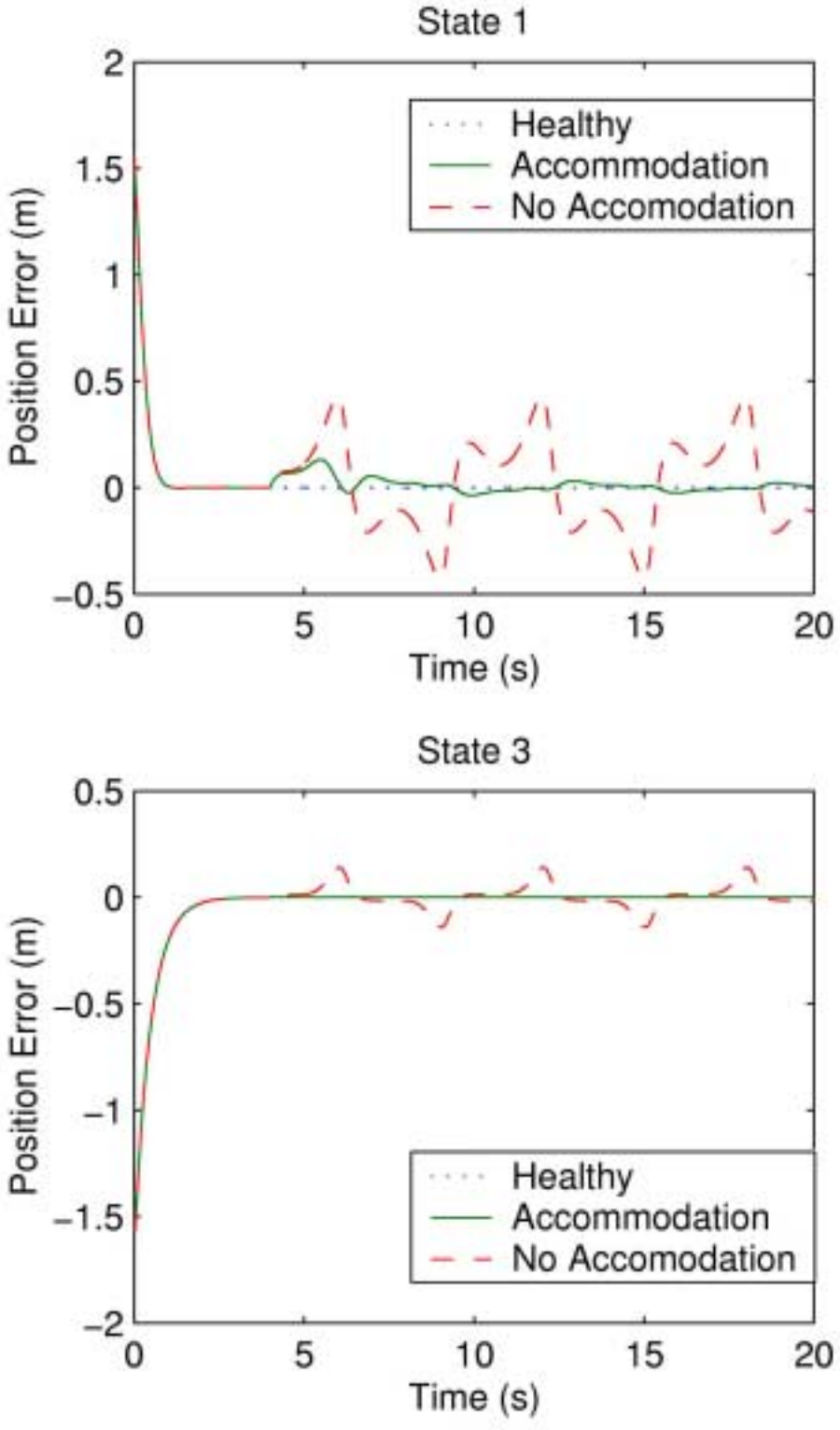


Figure 10 - DA observer: position error (States 1 & 3).

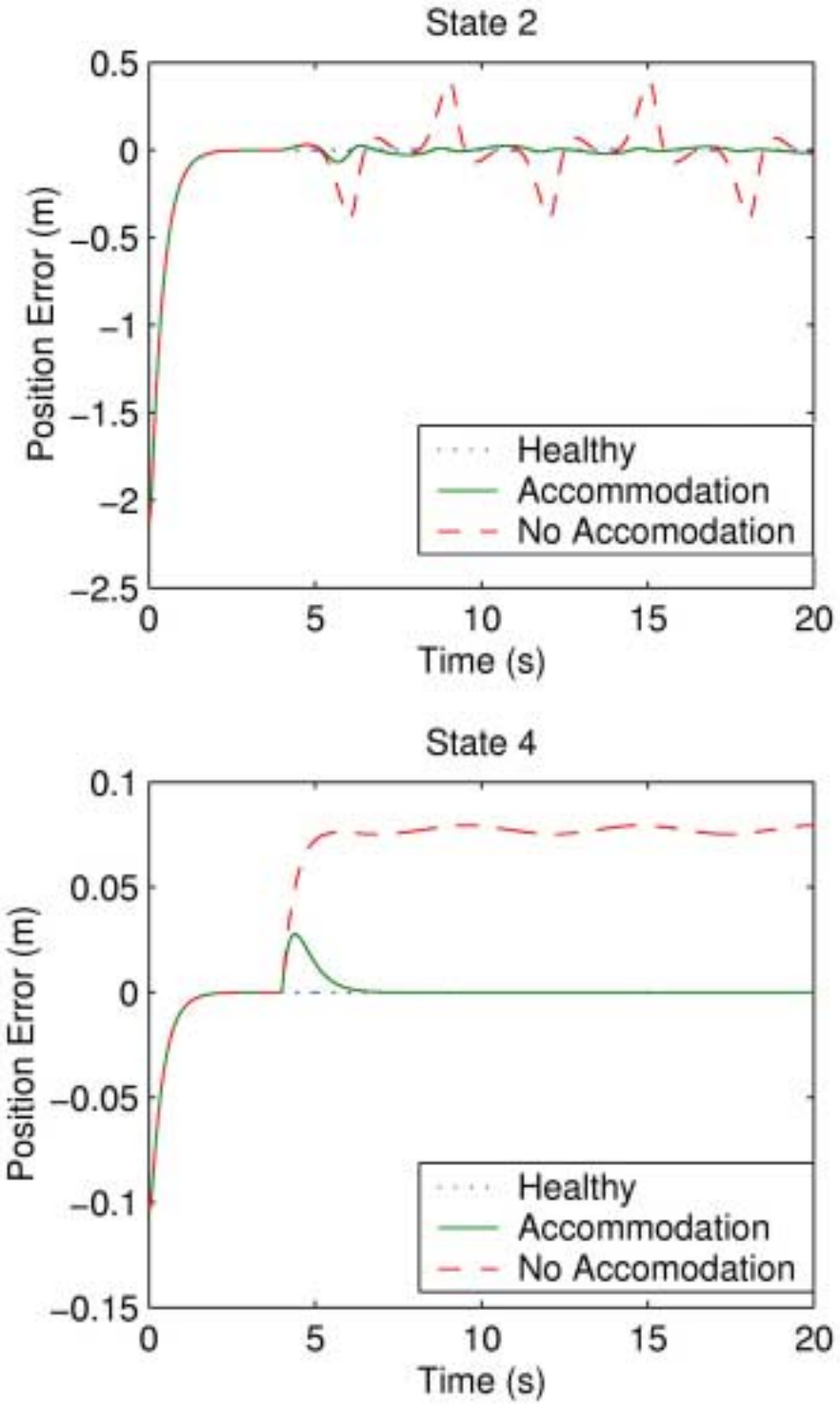


Figure 11 - DA observer: position error (States 2 & 4).

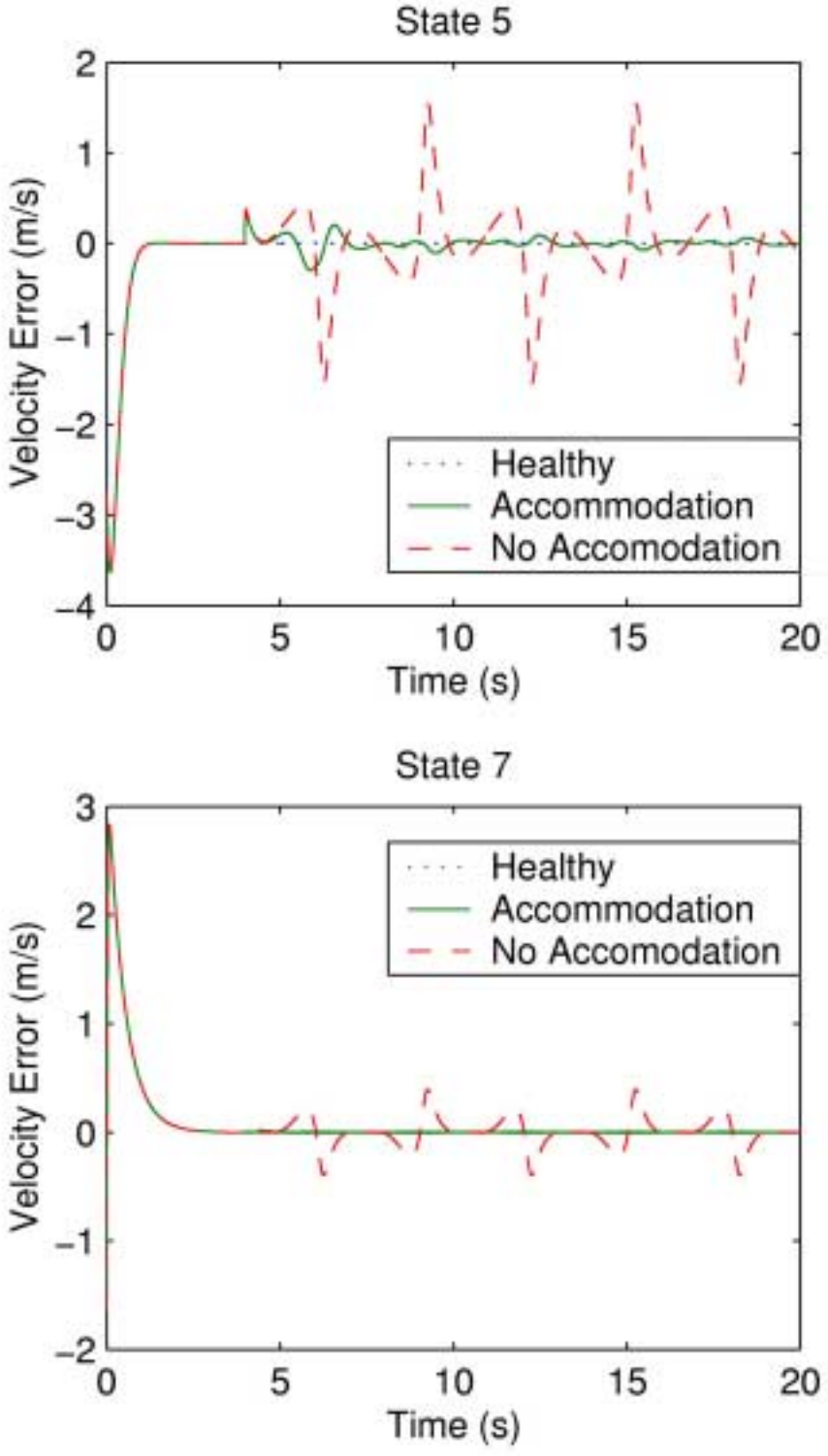


Figure 12 - DA observer: velocity error (States 5 & 7).

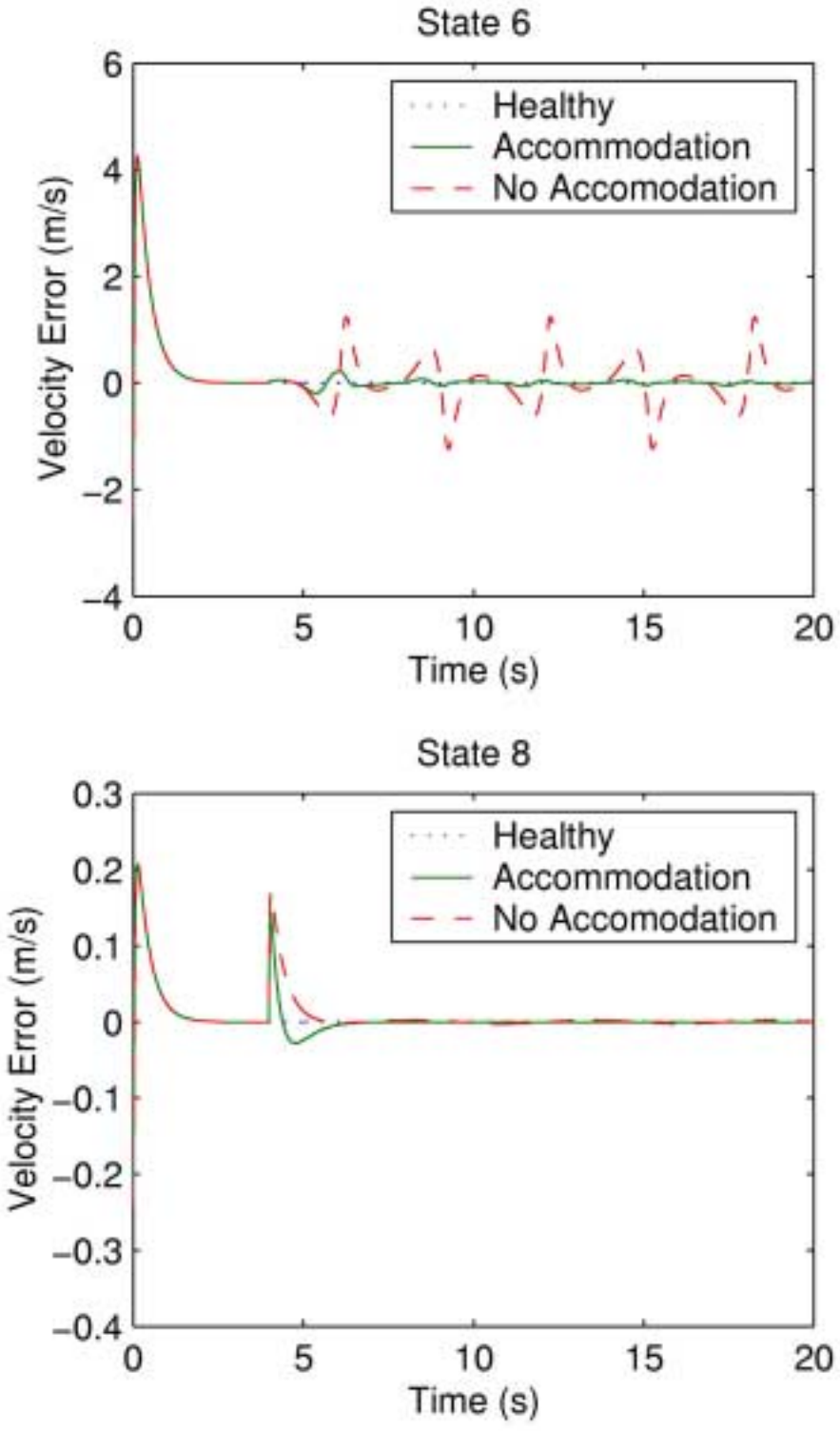


Figure 13 - DA observer: velocity error (States 6 & 8).

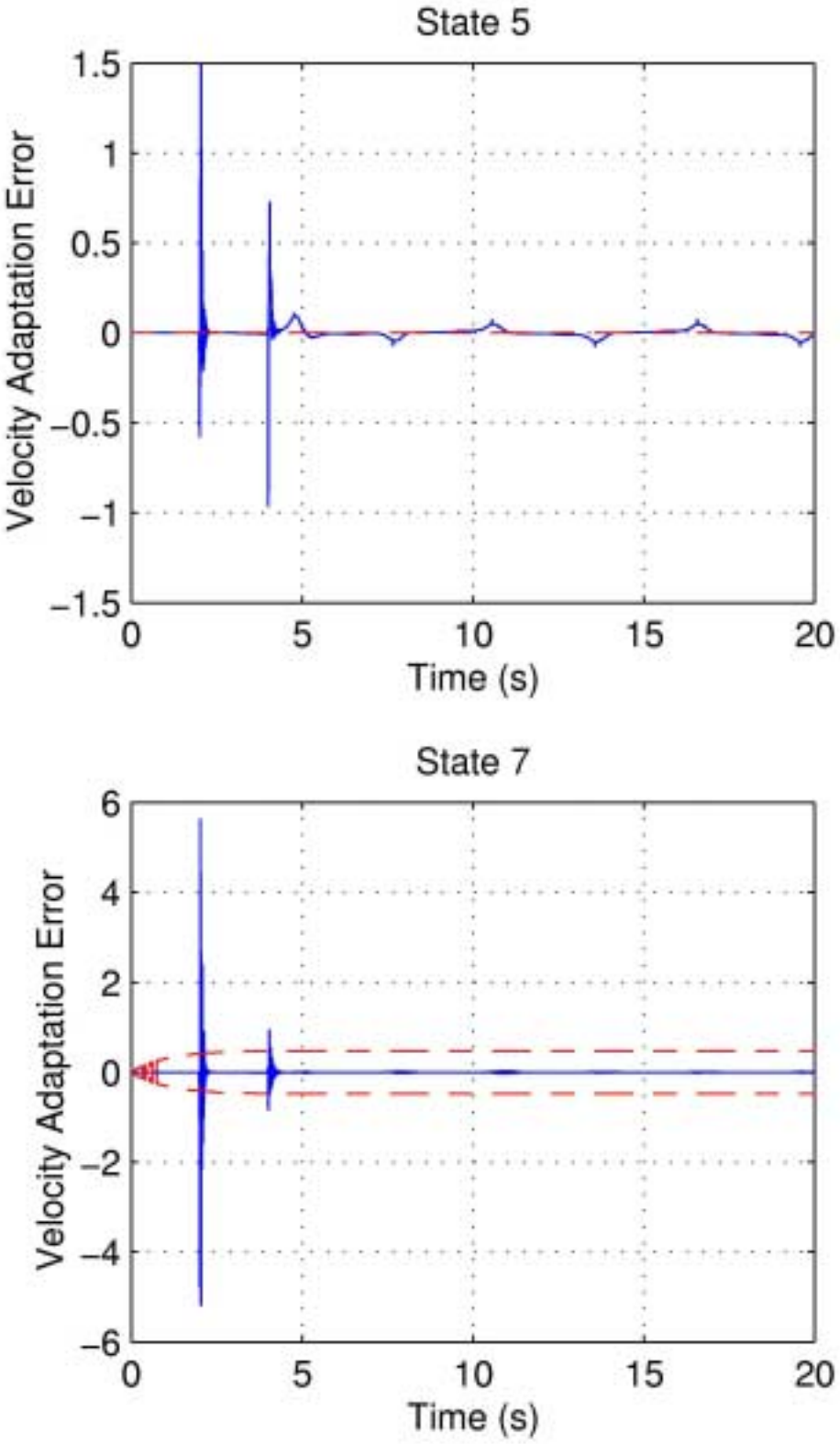


Figure 14 - DA observer: velocity estimation error (States 5&7).

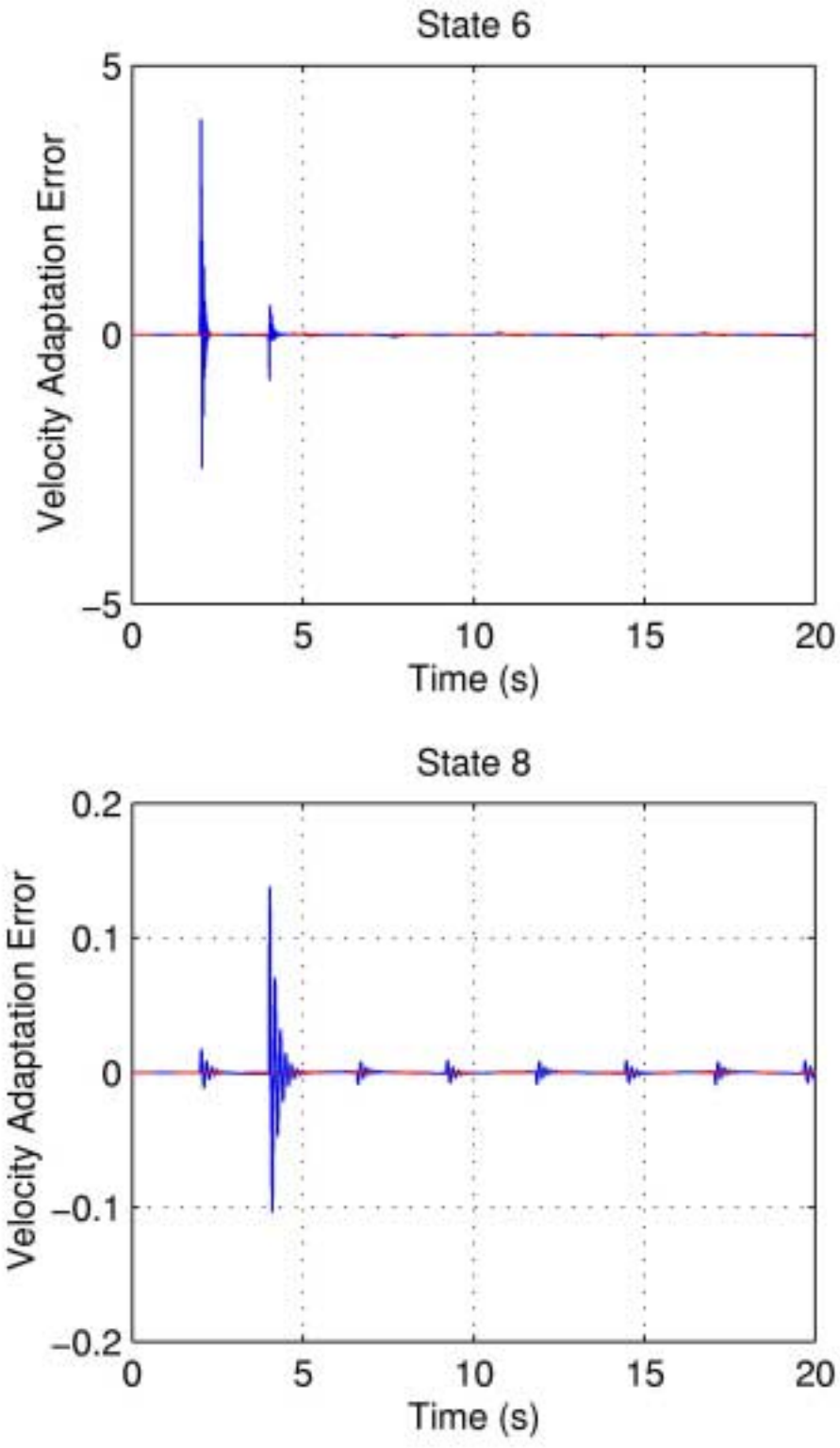


Figure 15 - DA observer: velocity estimation error (States 6&8).

During the second stage of the numerical study (Figure 16 - Figure 21), isolation performance of the CDIA scheme was analyzed. Three a priori known types of faults were included in the isolation filter bank, thus

$$\underbrace{2^3 - 1}_{\substack{\textit{Isolation} \\ \textit{filters}}} + \underbrace{\quad 1 \quad}_{\substack{\textit{Detection / Isolation} \\ \textit{Observer}}} = \underbrace{\quad 8 \quad}_{\substack{\textit{Total number} \\ \textit{of filters}}} .$$

Selected faults were I^2R, *Coulomb / Sticktion, Position Dependence*, which coincide with torque, velocity, and position dependent faults. Plots below present the simulation results, which point out the effectiveness of the scheme.

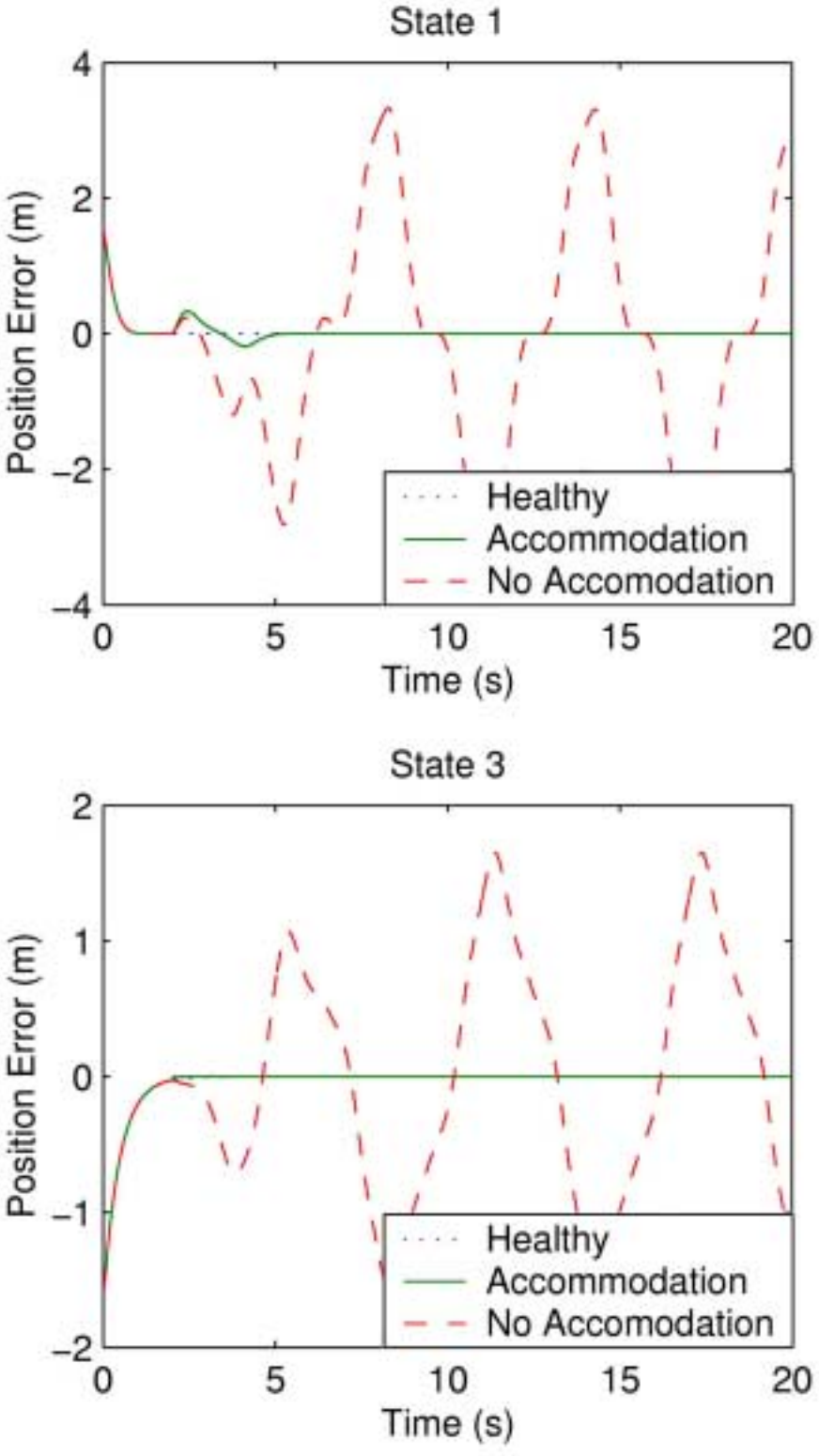


Figure 16 - Isolation: position error (States 1 & 2).

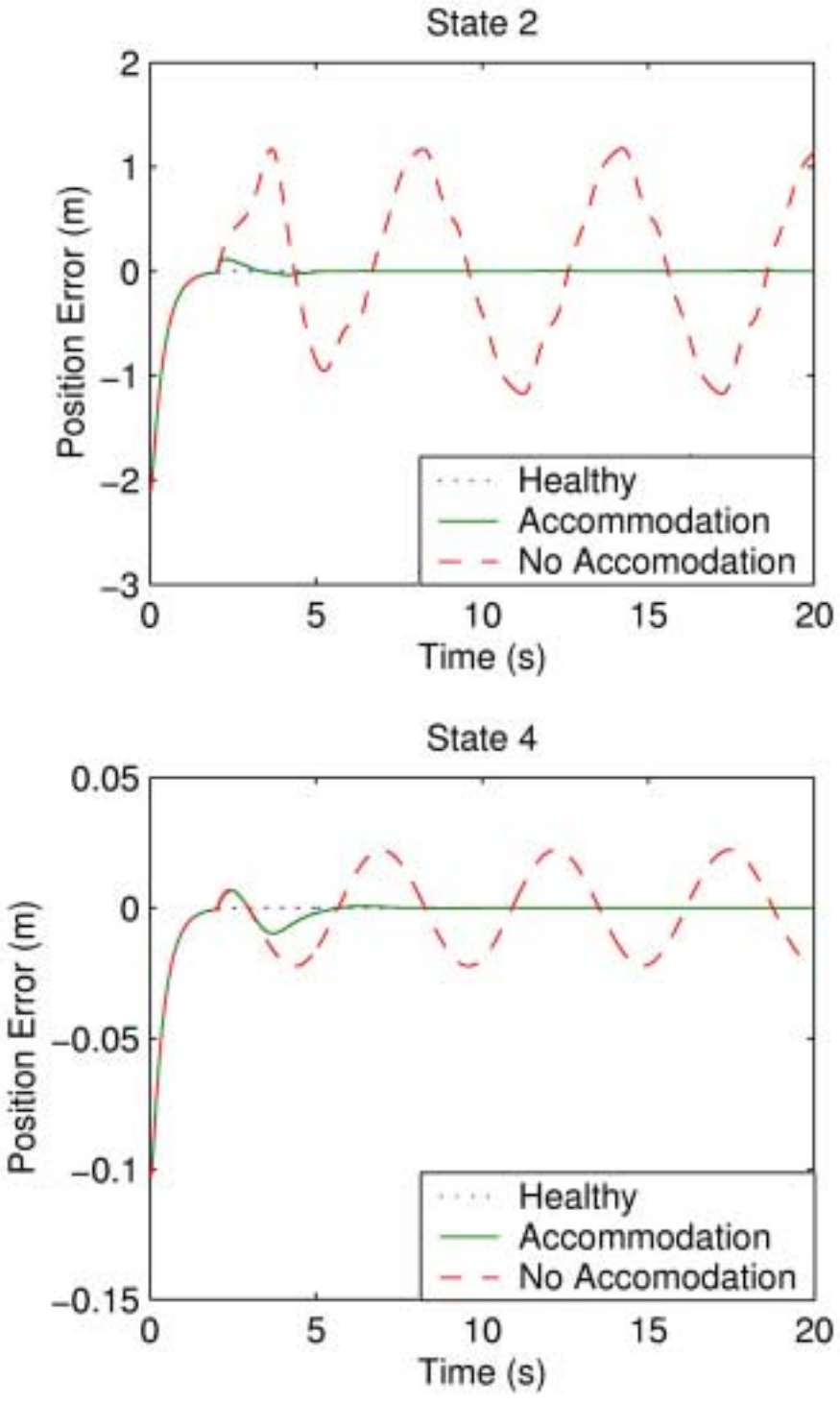


Figure 17 - Isolation: position error (States 2 & 4).

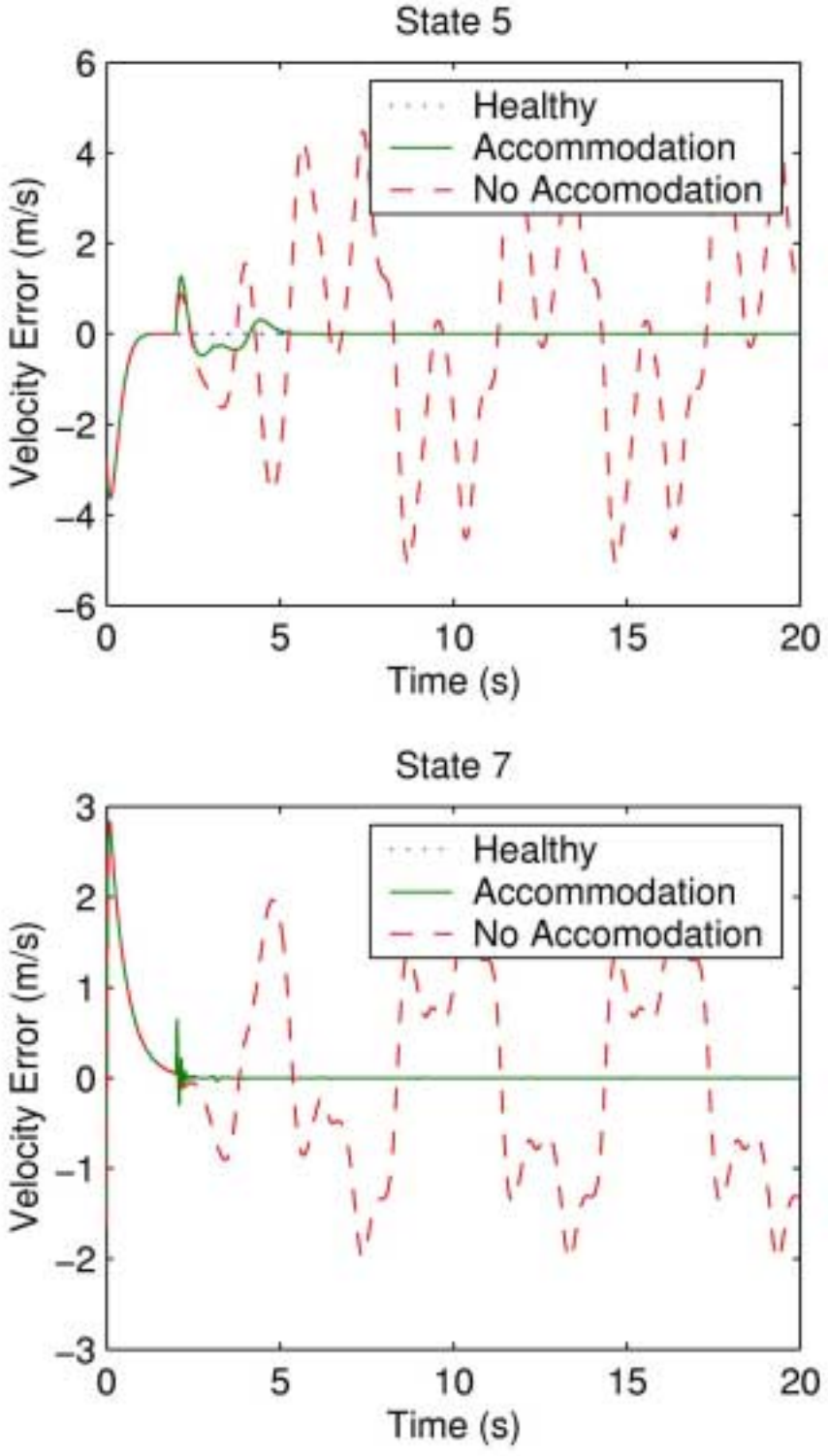


Figure 18 - Isolation: velocity error (States 4 & 6).

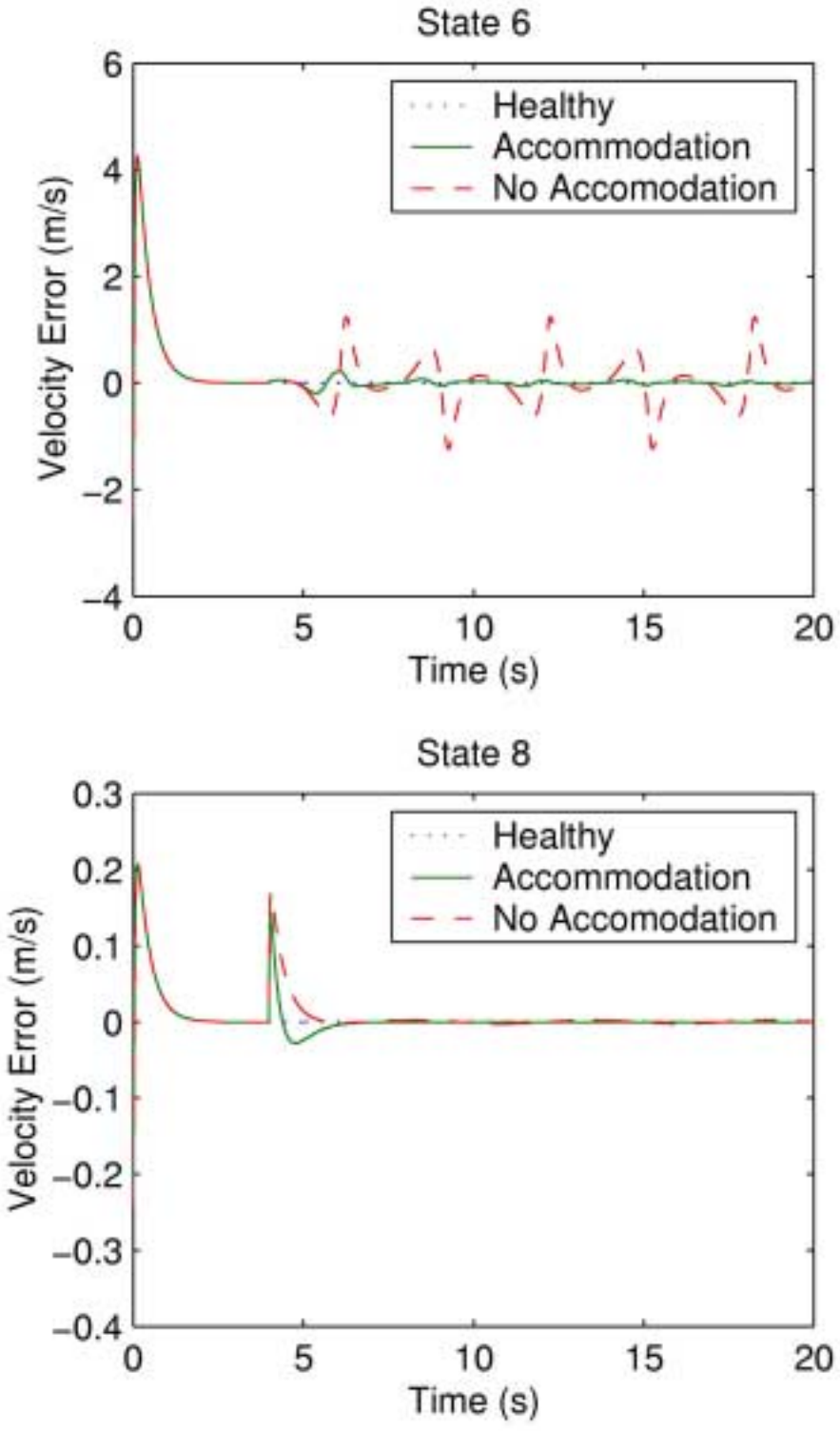


Figure 19 - Isolation: velocity error (States 6 & 8).

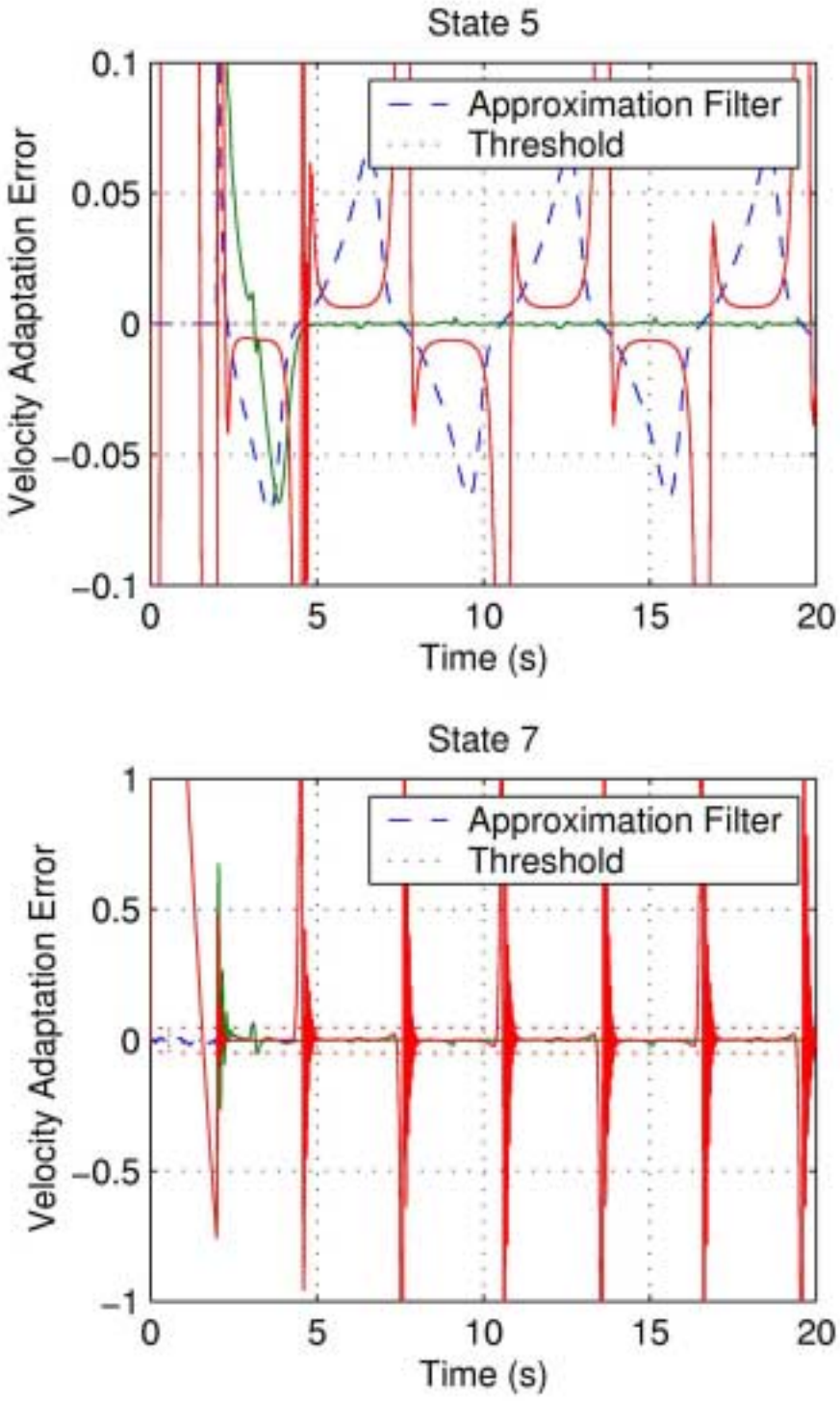


Figure 20 - Isolation: velocity estimation error (States 5 & 7).

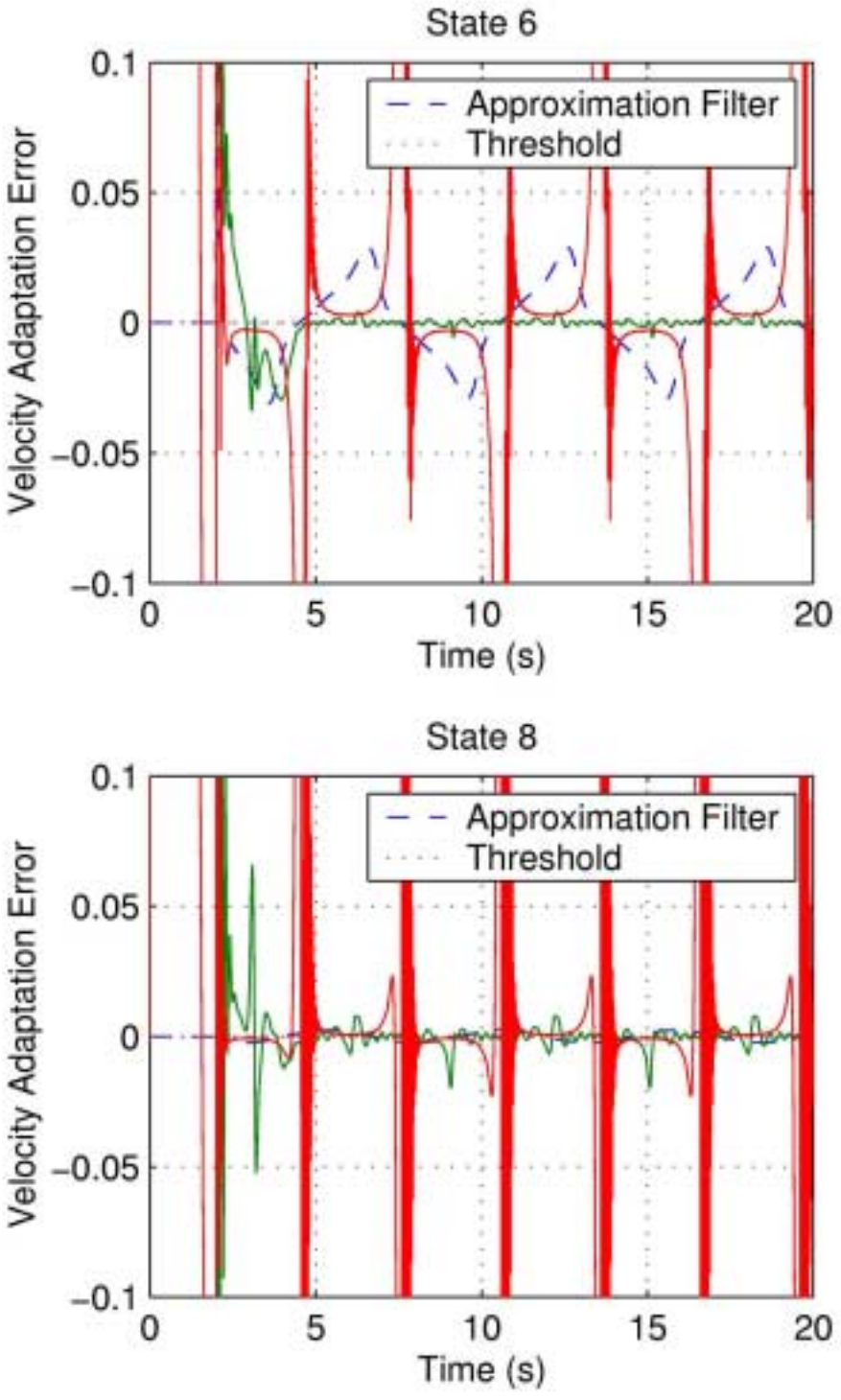


Figure 21 - Isolation: velocity estimation error (States 6 & 8).

5 CONCLUSIONS AND RECOMENDATION

Both internal and external changes (faults) can distort trajectory tracking, slow down a system's performance, decrease a system's capabilities, and even bring the system to a total halt. An innovative approach to model changes in non-linear systems was developed. Change (fault) profiles are modeled not only as time-dependent, but also as state-dependent. The new modeling technique was used to develop a very effective approach that both monitors the robotic system's health and its environment, and provides significant improvements to its performance. It is robust with respect to unmodeled dynamics, and torque dependent and state dependent changes. Change Detection, Isolation, and Accommodation (CDIA) can be easily reshaped to work with a wide variety of systems and changes. Its application requires minimal amount of additional hardware, and it also can be directly applied to already existing robotic systems. One of the great advantages of the approach is that it can be applied to

hydraulic, electrical or other types of robotic systems with minor modifications. This approach gives robotic system the tools to be aware of its constantly changing internal and external environment, identify or learn any changes, and accommodate them.

CDIA is an invaluable tool for autonomous systems. Examples are space, underwater technology, and hazardous environments. Maintenance is an important factor in the systems operation, especially in the areas where human access to the system is either limited or impossible. CDIA transforms regular robotic system to a much more intelligent system, capable of self-monitoring and self-correcting. It provides the system with tools to eliminate or decrease the need for maintenance for non-catastrophic changes. This has huge rewards not only in extreme environments. Maintenance is a very expansive exercise, and therefore the elimination of it provides operational cost cuts.

CDIA utilization is impossible without the use of the present day state of the art computational devices. The key idea of CDIA is its on-line in real-time execution. There are an enormous number of computational processes that have to be executed in real time in parallel to the operation of the real system. Therefore, CDIA received a significant attention in the last ten to fifteen years due to the advances in the DSP and other computer technologies. The tremendous leap in the computer technology of the recent years created opportunities for cheaper and better implementation of the CDIA technology. In addition to that, there has been a tremendous advances in neural networks and fuzzy logic, which also stimulated new researches and improvement in the CDIA.

A few recommendations, which directly follow from the presented work, can be made. This thesis analyzed full state feedback scenario, and the situations when feedback from not all of the state is available should also be investigated. Application of CDIA to

under-actuated robotic systems is yet another direction for research. In the future CDIA can be extended to other robotic systems (underwater for instance), and to general systems. The solid proof of the effectiveness and performance capabilities of the CDIA can be obtained by conducting a field test on the real robotic system.

The CDIA is a versatile base for the intelligent self-monitoring and correcting control systems that can grow on top of it. Work can be done in a number of directions to make it more advance and custom. It can be reshaped to work with other types of robotic systems that employ not only electric actuators, but hydraulic for instance. The CDIA can be applied to work not only with robots, but also with any control system where its self-correcting features are needed. Conducting a broader research on the dynamics of the changes can expand the bank of isolation filters and make it even more effective.

6 APPENDIX

6.1 SCARA Robot Dynamic Model

PARAMETERS:

i – link number,

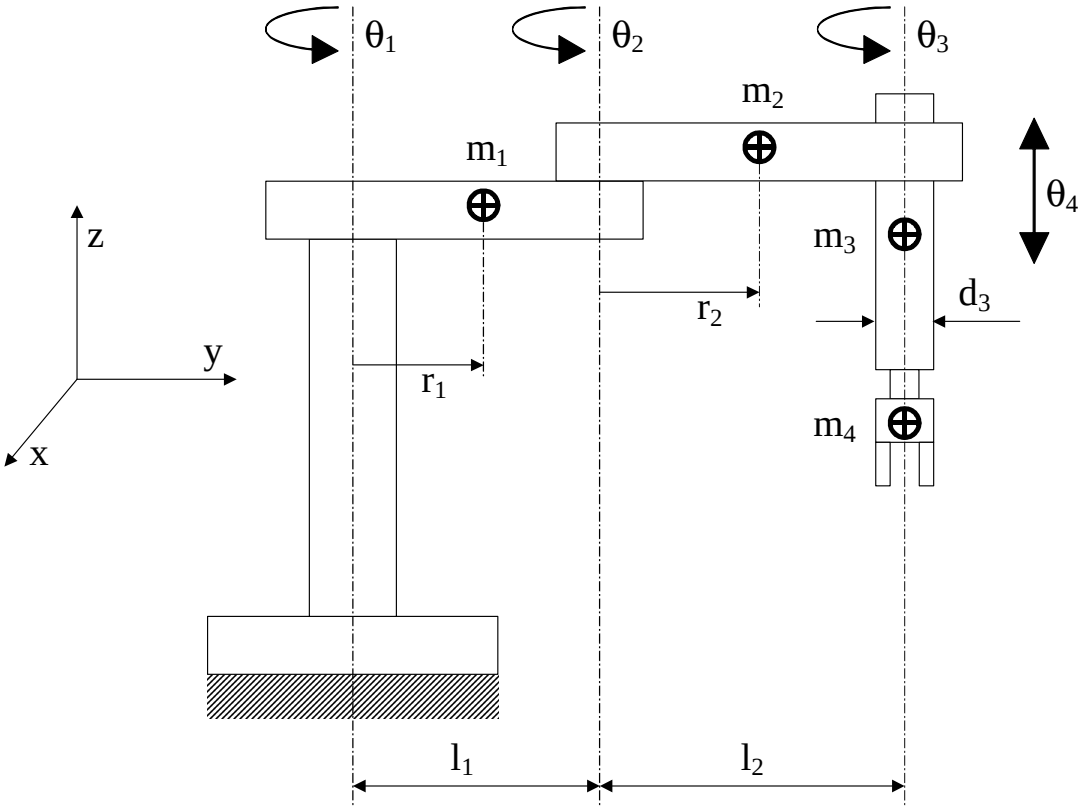
m_i – mass of the i^{th} link,

l_i – length of the i^{th} link,

θ_i – displacement of the i^{th} link,

r_i – distance from the joint to the center of mass of the i^{th} link

I_{zi} – moment of inertia of the i^{th} link in z direction relative to a frame attached at the center of mass of the link and aligned with the principle axes of the link.



GENERAL DYNAMIC MODEL:

$$M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G = \tau$$

where

$$\tau = \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 & F_4 \end{bmatrix}^T,$$

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & x_4 \end{bmatrix}^T,$$

$$G = \begin{bmatrix} 0 & 0 & 0 & m_3 g \end{bmatrix}^T,$$

$$M(\theta)=\begin{bmatrix} \alpha+\beta+2\gamma\cos(\theta_2) & \beta+\gamma\cos(\theta_2) & \delta & 0 \\ \beta+\gamma\cos(\theta_2) & \beta & \delta & 0 \\ \delta & \delta & \delta & 0 \\ 0 & 0 & 0 & m_4 \end{bmatrix},$$

$$V(\theta,\dot{\theta})=\begin{bmatrix} -\gamma\sin(\theta_2)\dot{\theta}_2\left(2\dot{\theta}_1+\dot{\theta}_2\right) \\ \gamma\sin(\theta_2)\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix},$$

$$\alpha = I_{z1}+r_1^2m_1+l_1^2m_2+l_1^2m_3+l_1^2m_4$$
$$\beta = I_{z2}+I_{z3}+I_{z4}+l_2^2m_3+l_2^2m_4+r_2^2m_2$$
$$\gamma = l_1l_2m_3+l_1l_2m_4+l_1m_2r_2$$
$$\delta = I_{z3}+I_{z4}.$$

ASSUMPTIONS:

1. Fault free operating conditions (no friction),
2. Rigid links,
3. Rigid structure of the joints (rigid motor shafts, no backslashes, rigid gearing),
4. Link 3 can be estimated to be a cylindrical rod, therefore

   $$I_{z3}=\frac{1}{2}m_3d_3^2,$$

5. Diameter of the link 3 (d_3) is much less then the length of the links 1 and 2 (l_1, l_2), therefore I_{z3} is negligible in comparison with l_1^2, l_2^2, and l_1l_2.
6. No load at the end effecter

   $\Rightarrow \quad m_4 = 0$
   $\Rightarrow \quad I_{z4} = 0,$

7. Link 3 has vertical translational motion

   $\Rightarrow \quad M_{44} = m_3$,

8. Centers of mass of links 1 and 2 are at the distant ends

   $\Rightarrow \quad r_1 = l_1$
   $r_2 = l_2$
   $I_{z1} = 0$
   $I_{z2} = 0.$

DYNAMIC MODEL:

$\alpha = l_1^2(m_1 + m_2 + m_3)$
$\beta = l_2^2(m_2 + m_3)$
$\gamma = l_1 l_2(m_2 + m_3)$
$\delta = I_{z3}$

$$M(\theta) = \begin{bmatrix} \begin{pmatrix}(m_1+m_2+m_3)l_1^2 + \\ +2(m_2+m_3)l_1 l_2\cos(\theta_2)+ \\ +(m_2+m_3)l_2^2\end{pmatrix} & \begin{pmatrix}(m_2+m_3)l_2^2 + \\ +(m_2+m_3)l_1 l_2\cos(\theta_2)\end{pmatrix} & I_{z3} & 0 \\ \begin{pmatrix}(m_2+m_3)l_2^2 + \\ +(m_2+m_3)l_1 l_2\cos(\theta_2)\end{pmatrix} & (m_2+m_3)l_2^2 & I_{z3} & 0 \\ I_{z3} & I_{z3} & I_{z3} & 0 \\ 0 & 0 & 0 & m_3 \end{bmatrix}$$

$$V(\theta,\dot{\theta}) = \begin{bmatrix} -(m_2+m_3)l_1 l_2\sin(\theta_2)\dot{\theta}_2(\dot{\theta}_2 - 2\dot{\theta}_1) \\ (m_2+m_3)l_1 l_2\sin(\theta_2)\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix}.$$

6.2 Simulation Code

Main

```
% SRMain
% Scara Robot - AdeptOne-XL
clear all
clc
global M Mi VG a1 a2 a3 Fa Fc FA FC Kpv Ke Ka Kc c sgm k n0 thh ind acc und td

% SETUP
%=============================================================================================
k=7;                              % Number of neurons per state
tf=2;                             % Simulation time

step=0.005;                           % Time step
tt=0:step:tf;                         %
m1=50;    m2=40;    m3=30;            % Weights of the links (kg)
l1=0.425; l2=0.375;                   % Lenghts of the links (m)
j3=m3*0.02^2/2;                       % Moments of inertia of the 3rd link
g=9.8;                                % Gravitational acceleration
P=pi*[5/6 7/9 3/2 0.2/pi];            % Maximum joint range
V=pi*[10/3 5 55/3 1.2/pi];            % Maximum joint speed
P0=pi*[1/2;-2/3;-1/2;-0.1/pi];        % IC(Initial conditions)-position
V0 =zeros(4,1);                       % IC-velocity
Ve0=zeros(4,1);                       % IC-velocity estimates
H0 =zeros(4,1);                       % IC-actuator fault weights
L0 =zeros(k*8,1);                     % IC-component neurons
X0 =[P0;V0;Ve0;H0;L0];                % IC-vector
n0=pi*[30;5;55/3;1.2/pi]*5e-5;        % Modeling uncertanty upper-bound
ind=[(17:4:(16+k*8));(18:4:(16+k*8));(19:4:(16+k*8));(20:4:(16+k*8))];%

a1=(m2+m3)*l2^2;                          % Inertia matrix
a2=(m2+m3)*l2*l1;                         %
a3=a1+(m1+m2+m3)*l1^2;                    %
M=zeros(4,4);                             %
M(1:3,1:3)=j3;                            %
M(2,2)=a1;                                %
M(4,4)=m3;                                %
VG=[0;0;0;g*m3];                          % Coriolis/centripetal/gravity matrix
Fa=zeros(4,1);                            % Initial actuator  faults
Fc=zeros(4,1);                            % Initial component faults
FA=-[7;10;10;0.9];                        % Actuator  faults
FC=[1e2;1e2;1e-2;1e1];                    % Component faults

fprintf('     Generating  DESIRED TRAJECTORY\n\n'); % Calculating desired trajectory
for i=0:(tf/step)                                  %
    xd(i+1,:)=srt(i*step)';                        %
end                                                %

Kpv=[diag([30 60 90 60]) diag([10 25 45 25])];     % Position/Velocity gains
Ke=[1e2;1e2;1e2;1e2];                              % Estimator error gains
Ka=[1e1;1e1;1e0;1e0];                              % Actuator adaptation gains
Kc=repmat([1e1;1e1;1e-1;1e5],k*2,1);               % Component neuron gains
c=[linspace(-P(1),P(1),k) linspace(-V(1),V(1),k);  % Neuron centers
   linspace(-P(2),P(2),k) linspace(-V(2),V(2),k);  %
   linspace(-P(3),P(3),k) linspace(-V(3),V(3),k);  %
   linspace(-P(4),P(4),k) linspace(-V(4),V(4),k)]; %
sgm=1e-3;                                          % Neuron weights
```

```
% SYSTEM SYMULATION
%=============================================================================================

options=odeset('JConstant','on','RelTol',1e-4,'AbsTol',1e-4);
disp('     Integrating HEALTHY SYSTEM');                    % > Healthy System integration
[t,x1]=ode23s('srh',tt,[P0;V0],options);                    %
disp('     Integrating FAULTY  SYSTEM');                    % > Faulty  System integration
[t,x3]=ode23s('srf3',tt,[P0;V0],options);                   %

Fa=zeros(4,1);Fc=zeros(4,1);
sys=ss(zeros(4,4),ones(4,4),ones(4,4),zeros(4,4));
thh=ones(4,1)*100;
time(1)=0;
x2(1,:)=X0';
for i=1:(tf/step)
   tl=(i-1)*step;                                           % Initial time of ith subinterval
   tr=i*step;                                               % Final time  of ith subinterval
   [t,x]=ode23s('srd6',[tl:(tr-tl)/2:tr],X0,options);  % Integration
   x2(i+1,:)=x(3,:);                                        % Sssign to vector x2 value @tr
   time(i+1)=tr;                                            % Save next entry in time vector
   X0=x2(i+1,:);                                            % Assign x@tr to be x@0 (IC) for
next time subinterval
   clc
   fprintf('\n\n   Integrating ACCOMODATED SYSTEM t=%.4f',time(i+1));
   u(:,i+1)=exp(Ke*tr).*(Mi*n0);
   th1=lsim(sys,u',time ,zeros(1,4));
   thh=exp(-Ke*tr).*th1(i,:)';
   thd(:,i+1)=thh;
end
t=time;

% Output
%=============================================================================================
figure(1)
subplot(221),plot(t,x1(:,1)-xd(:,1),':',t,x2(:,1)-xd(:,1),t,x3(:,1)-xd(:,1),'--')
title('State 1');ylabel('Position Error (m)');xlabel('Time (s)')
legend('Healthy','Accommodation','No Accomodation');%axis([0 20 -1 2]);
subplot(222),plot(t,x1(:,2)-xd(:,2),':',t,x2(:,2)-xd(:,2),t,x3(:,2)-xd(:,2),'--')
title('State 2');ylabel('Position Error (m)');xlabel('Time (s)')
legend('Healthy','Accommodation','No Accomodation');
subplot(223),plot(t,x1(:,3)-xd(:,3),':',t,x2(:,3)-xd(:,3),t,x3(:,3)-xd(:,3),'--')
title('State 3');ylabel('Position Error (m)');xlabel('Time (s)')
legend('Healthy','Accommodation','No Accomodation');
subplot(224),plot(t,x1(:,4)-xd(:,4),':',t,x2(:,4)-xd(:,4),t,x3(:,4)-xd(:,4),'--')
title('State 4');ylabel('Position Error (m)');xlabel('Time (s)')
legend('Healthy','Accommodation','No Accomodation');

figure(2)
subplot(221),plot(t,x1(:,5)-xd(:,5),':',t,x2(:,5)-xd(:,5),t,x3(:,5)-xd(:,5),'--')
title('State 5');xlabel('Time (s)');ylabel('Velocity Error (m/s)');
legend('Healthy','Accommodation','No Accomodation');
subplot(222),plot(t,x1(:,6)-xd(:,6),':',t,x2(:,6)-xd(:,6),t,x3(:,6)-xd(:,6),'--')
title('State 6');xlabel('Time (s)');ylabel('Velocity Error (m/s)');
legend('Healthy','Accommodation','No Accomodation');
subplot(223),plot(t,x1(:,7)-xd(:,7),':',t,x2(:,7)-xd(:,7),t,x3(:,7)-xd(:,7),'--')
title('State 7');xlabel('Time (s)');ylabel('Velocity Error (m/s)');
legend('Healthy','Accommodation','No Accomodation');
subplot(224),plot(t,x1(:,8)-xd(:,8),':',t,x2(:,8)-xd(:,8),t,x3(:,8)-xd(:,8),'--')
title('State 8');xlabel('Time (s)');ylabel('Velocity Error (m/s)');
legend('Healthy','Accommodation','No Accomodation');

figure(3)
subplot(221),plot(t,x2(:,5)-x2(:,9 ),t,thd(1,:),'r--',t,-thd(1,:),'r--')
title('State 5');grid;xlabel('Time (s)');ylabel('Velocity Adaptation Error');
subplot(222),plot(t,x2(:,6)-x2(:,10),t,thd(2,:),'r--',t,-thd(2,:),'r--')
title('State 6');grid;xlabel('Time (s)');ylabel('Velocity Adaptation Error');
subplot(223),plot(t,x2(:,7)-x2(:,11),t,thd(3,:),'r--',t,-thd(3,:),'r--')
title('State 7');grid;xlabel('Time (s)');ylabel('Velocity Adaptation Error');
subplot(224),plot(t,x2(:,8)-x2(:,12),t,thd(4,:),'r--',t,-thd(4,:),'r--')
title('State 8');grid;xlabel('Time (s)');ylabel('Velocity Adaptation Error');
beep
```

Trajectory Generator

```
% FUNCTION SRTRAJECTORY
function [xd]=srtrajectory(t)

pt=pi*[5/6 1/3 7/9 1/3 1/2 1/3 0.2/pi 1.2/pi];

xd=[pt(1)*sin(pt(2)*t);          pt(3)*sin(pt(4)*t);
    pt(5)*sin(pt(6)*t);          pt(7)*sin(pt(8)*t);
    pt(1)*pt(2)*cos(pt(2)*t);    pt(3)*pt(4)*cos(pt(4)*t);
    pt(5)*pt(6)*cos(pt(6)*t);    pt(7)*pt(8)*cos(pt(8)*t);
   -pt(1)*pt(2)^2*sin(pt(2)*t); -pt(3)*pt(4)^2*sin(pt(4)*t);
   -pt(5)*pt(6)^2*sin(pt(6)*t); -pt(7)*pt(8)^2*sin(pt(8)*t)];
```

Healthy System Simulator

```
% FUNCTION SRh
function xdot=SRh(t,x)
global M VG a1 a2 a3 U Kpv

% SYSTEM
%=============================================================================================
M(1,1)=a3+2*a2*cos(x(2));                   % Inertia matrix
M(1,2)=a1+a2*cos(x(2));
M(2,1)=M(1,2);
Mi=inv(M);                                  % Inertia matrix inverse
VG(1,1)=-a2*sin(x(2))*x(6)*(x(6)+2*x(5));   % Coriolis, centripetal
VG(2,1)=a2*sin(x(2))*x(5)^2;                % and gravity forces matrix

% CONTROLLER
%=============================================================================================
xd=srtrajectory(t);                         % Desired trajectory
epv=x(1:8)-xd(1:8);                         % Velocity/Position error
U=M*(xd(9:12)-Kpv*epv)+VG;                  % Input

xdot(1:4,1)=x(5:8);
xdot(5:8,1)=Mi*(U-VG);
```

System with the Faults

```
% FUNCTION SRMODELF
function xdot=SRMODELF(t,x)
global M VG a1 a2 a3 F FC Kpv

% SYSTEM
%=============================================================================================
M(1,1)=a3+2*a2*cos(x(2));                    % Inertia matrix
M(1,2)=a1+a2*cos(x(2));
M(2,1)=M(1,2);
Mi=inv(M);                                   % Inertia matrix inverse
VG(1,1)=-a2*sin(x(2))*x(6)*(x(6)+2*x(5));    % Coriolis, centripetal
VG(2,1)=a2*sin(x(2))*x(5)^2;                 % and gravity forces matrix

if t>4
   Fa=FA;                                    % Actuator faults
end
if t>2
```

```
   Fc=FC.*(-sign(x(5:8)).*(1+0.05*exp(-1e6*abs(x(5:8))))+20*x(5:8)+10*sin(0.2*x(1:4)+pi))
end
und=(cos(5*x(1:4))+sin(15*x(5:8))).*[1e-4;1e-4;1e-3;1e-6]; % Modeling uncertanty

% CONTROLLER
%================================================================================================
xd=srtrajectory(t);                                  % Desired trajectory
epv=x(1:8)-xd(1:8);                                  % Velocity error
U=M*(xd(9:12)-Kpv*epv)+VG;                           % Healthy system input
xdot(1:4,1)=x(5:8);                                  % System
xdot(5:8,1)=Mi*(U-F-VG-und);                         % System
```

Detection/Accommodation

```
% FUNCTION SRD1

function xdot=SRD1(t,x)
global M Mi VG a1 a2 a3 Fa Fc FA FC Kpv Ke Ka Kc c sgm k ind acc und thh n0 td

% SYSTEM
%================================================================================================
M(1,1)=a3+2*a2*cos(x(2));                    % Inertia matrix
M(1,2)=a1+a2*cos(x(2));
M(2,1)=M(1,2);
Mi=inv(M);                                   % Inertia matrix inverse
VG(1,1)=-a2*sin(x(2))*x(6)*(x(6)+2*x(5));    % Coriolis, centripetal
VG(2,1)=a2*sin(x(2))*x(5)^2;                 % and gravity forces matrix

if t>4
   Fa=FA;                                    % Actuator faults
end
if t>2
   Fc=FC.*(-sign(x(5:8)).*(1+0.05*exp(-1e6*abs(x(5:8))))+20*x(5:8)+10*sin(0.2*x(1:4)+pi))
end

n=(cos(5*x(1:4))+sin(15*x(5:8))).*[1e-4;1e-4;1e-3;1e-6]*und;% Modeling uncertanty

% CONTROLLER
%================================================================================================
QZ=exp(-([repmat(x(1:4),1,k) repmat(x(5:8),1,k)]-c).^2.*sgm);
fc=sum((QZ.*x(ind))')';

xd=srt(t);                                                  % Desired trajectory
epv=x(1:8)-xd(1:8);                                         % Velocity error
ea=x(9:12)-x(5:8);                                          % Adaptation error
U=M*(xd(9:12)-Kpv*epv)+VG;                                  % Nominal input
U=U+acc*(inv(diag(1-x(13:16)))*(U+fc)-U);                   % Full input

xdot(1:4,1)  =x(5:8);
xdot(5:8,1)  =Mi*((1-Fa).*U-Fc-VG-n);
xdot(9:12,1) =Mi*((1-x(13:16)).*U-fc-VG)-Ke.*ea;
if sum((abs(ea)>abs(thh)))>0
   xdot(13:16,1)=Ka.*U.*(Mi*ea);
   xdot(17:(16+k*8),1)=Kc.*QZ(:).*repmat((Mi*ea),k*2,1);
else
   x(13:(16+k*8),1)=zeros((4+k*8),1);
   xdot(13:(16+k*8),1)=zeros((4+k*8),1);
   td=t;
end
```

7 BIBLIOGRAPHY

[1] H. Assada, J. J. E. Slotime, *Robot Analysis and Control*, A Wiley Interscience, 1986.

[2] J. J. E. Slotine, W. Li, *Applied Nonlinear Control*, Prentice Hall, 1991.

[3] Y. H. Kim, F. L. Lewis, *High-Level Feedback Control with Neural Networks*, World Scientific Publishing Co. Pte. Ltd., 1998.

[4] Robert J. Schilling, *Fundamentals of Robotics: Analysis and Control*, Prentice Hall, 1990.

[5] Richard M. Murray, Zexiang Li, and S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulators*, CRC Press Inc., 1994.

[6] Felix L. Chernousko, Nikolai N. Bolotnik, and Valery G. Gradetsky, *Manipulation Robots: Dynamics, Control, and Optimization*, CRC Press Inc., 1994.

[7] R. Ortega, A. Loria, P. Nicklasson, H. Sira, *Passivity-based control of Euler-Lagrange systems: mechanical, electrical, and electromechanical applications*, Springer, 1998.

[8] D. P. Sen Gupta, J. W. Lynn, *Electrical Machine Dynamics*, The Macmillan Press Ltd, 1980.

[9] L. W. Matsch Late, J. D. Morgan, *Electromagnetic and Electromechanical Machines*, Haprer & Row Publishers Inc, 1986.

[10] B. J. Chalmers, *Electic Motor Handbook*, Butterworth & Co. Ltd., 1988.

[11] F. L. Lewis, S. Jagannathan, *Neural Network Control of Robot Manipulators and Non-Linear Systems*, Taylor & Francis, 1999.

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

[13] D. V. Richardson, *Rotating Electric Machinery and Transformer Technology*, Reston Publishing Company, Inc., 1982.

[14] R. P. Grimaldi, *Discrete and Combinatorial Mathematics: an Applied Introduction*, Addison-Wesley Publishing Company, Inc., 1994.

[15] C. De Persis, and Alberto Isidori, "A Geometric Approach on Nonlinear Fault Detection and Isolation," *IEEE Transactions on Automation Control*, vol. 46, no. 6, 2001

[16] X. Zhang, M. M. Polycarpou, and T. Parisini, "Fault Isolation in a Class of Nonlinear Uncertain Input-Output Systems," *Proceedings of the American Control Conference*, 2001.

[17] M. M. Polycarpou, and A. J. Helmicki, "Automated Fault Detection and Accommodation: A Learning Systems Approach," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 11, pp. 1447-1458, 1995.

[18] A. T. Vemuri, M. M. Polycarpou, and S. A. Diakourtis, "Neural Networks Based Fault Detection in Robotic Manipulators," *IEEE Transactions on Robotics Automations*, vol. 14, no. 2, pp. 342-348, 1998.

[19] M. Feemster, D. M. Dawson, A. Behal, and W. Dixon, "Tracking Control in the Presence of Nonlinear Dynamic Friction Effects: Robot Extension," *IEEE International Conference on Control Applications*, pp. 1169-1174, 1999.

[20] A. T. Vemuri, and M. M. Polycarpou, "Neural-Network-Based Robust Fault Diagnosis in Robotic Systems", *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1410-1416, 1997.

[21] X. Zhang, T. Parisini, and M. Polycarpou, "Robust Parametric Fault Detection and Isolation for Nonlinear Systems", *Proceedings of the 38th Conference on Decision and Control*, pp. 3102-3107, 1999.

[22] A. B. Trunov, and M. M. Polycarpou, "Robust Nonlinear Fault Diagnosis: Application to Robotic Systems," *IEEE International Conference on Control Applications*, pp. 1424-1430, 1999.

[23] C. Canudas de Wit, P. Noel, A. Aubin, B. Brogliato, and P. Drevet, "Adaptive Friction Compensation in Robot Manipulators: Low-Velocities", *IEEE*, pp. 1352-1357, 1989.

[24] C. Canudas de Wit, H. Olsson, K. J. Astrom, P. Lischinsky, "A New Model for Control of Systems with Friction", *IEEE Transactions on Automatic Control*, vol. 40, no. 3, 1995.

[25] C. Canudas de Wit, and S. S. Ge, "Adaptive Friction Compensation for Systems with Generalized Velocity/Position Friction Dependency," *IEEE Proceedings of the 36th Conference on Decision & Control*, pp. 2465-2470, 1997.

[26] M. M. Polycarpou, A. B. Trunov, "Learning approach to non-linear fault diagnosis: Detectability Analysis", *IEEE Transactions on Automatic Control*, vol. 45, no. 6, 2000.

[27] J. Anshul, M. A. Demetriou, "A Neural Network Based Actuator Fault Detection and Diagnostic Scheme for a SCARA Manipulator," *IEEE Proceedings of the 15th International Symposium on Intelligent Control*, pp. 297-302, 2000.

[28] M. A. Demetriou, M. M. Polycarpou, "Incipient Fault Diagnosis of Dynamical Systems Using Online Approximators," *IEEE Transactions on Automatic Control*, vol. 43, no. 11, 1998.

[29] B. Armstrong-Helouvry, "Dynamic Friction in Control of Robot", *IEEE*, pp. 1202-1207, 1994.

[30] M. H. Terra, and R. Tinos, "Fault Detection and Isolation in Robotic Systems via Artificial Neural Networks," *Proceedings of the 37th IEEE Conference on Decision & Control*, 1998.

[31] M. H. Terra, and R. Tinos, "Fault Detection and Isolation in Robotic Manipulators via Neural Networks: A Comparison Among Three Architectures for Residual Analysis," *Journal of Robotic Systems*, pp. 357-374, 2001.

[32] A. A. Jain, and M. A. Demetriou, "A Neural Network Based Actuator Fault Detection and Diagnostic Scheme for a Scara Manipulator," *Proceedings of the 15th IEEE International Symposium on Intelligent Control*, pp. 297-301, 2000.

[33] F. Caccavale, "Experiments of Observer-based Fault Detection for an Industrial Robot," *Proceeding of the IEEE International Conference on Control Applications*, pp. 480-484, 1998.

[34] F. Caccavale, and I. D. Walker, "Observer-based Fault Detection for Robot Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.

[35] P. Tomei, "Robust Adaptive Friction Compensation for Tracking Control of Robots," *Proceedings of the IEEE International Conference on Control Application*, pp. 875-879, 1999.

[36] A. Alessandri, M. Caccia, and G. Veruggio, "A Model-based Approach to Fault Diagnosis in Unmanned Underwater Vehicles,"*IEEE*, 1998.

[37] A. B. Trunov, and M. M. Polycarpou, "Automated Fault Diagnosis in Nonlinear Multivariable Systems Using a Learning Methodology," *IEEE Transactions on Neural Networks*, vol. 11, no. 1, 2000.

[38] M. A. Demetriou, and M. M. Polycarpou, "Fault Detection, Diagnosis and Accommodation of Dynamical Systems with Actuator Failures via On-Line Approximators," *Proceedings of the American Control Conference*, pp. 2879-2883, 1998.

[39] K. Kiguchi, and T. Fukuda, "Fuzzy Neural Friction Compensation Method of Robot Manipulation During Position/Force Control," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 372-377, 1996.

[40] M. C. Pan, H. V. Brussel, and P. Sas, "Intelligent Joint Fault Diagnosis of Industrial Robots," *Mechanical Systems and Signal Processing*, pp. 571-588, 1998.

[41] P. Goel, G. Dedeoglu, S. I. Roumeliotis, G. S. Sukhatme, "Fault Detection and Isolation in a Mobile Robot Using Multiple Model Estimation and Neural Network," *Proceeding of the IEEE International Conference on Robotics & Automation*, 2000.

[42] J. J. Gonzalez, and G. R. Widmann, "A New Model for Nonlinear Friction Compensation in the Force Control of Robot Manipulators," *Proceeding of the IEEE International Conference on Control Application*, pp. 201-203, 1997.

[43] C. Canudas de Wit, "Comments on "A New Model for Control of Systems with Friction"," *IEEE Transactions on Automatic Control*, vol. 43, no. 8, 1998.

[44] Yi Xion, and Mehrdad Saif, "Robust Fault Isolation Observe Design", *Proceedings of the American Control Conference*, pp. 2077 – 2081, 1999.

[45] Krishna K. Busawon, and Mehrdad Saif, "A State Observer for Nonlinear Systems", *IEEE Transactions on Automatic Control*, vol. 14, no. 11, 1999.

[46] Marius Ghetie, and Mehrdad Saif, "On-line Fault Detection and Isolation Using Unbalanced Residual", *Proceedings of the 38th Conference on Decision and Control*, pp. 4480 – 4485, 1999.
[47] Gang Tao, Shuhao Chen, S. M. Joshi, "An Adaptive Control scheme for systems with unknown actuator failures", *Proceedings of the American Control Conference*, pp. 1115-1120, vol. 2, 2001.
[48] A. Taware, Gang Tao, C. Teolis, "Neural-hybrid control of systems with sandwiched dead-zones", *Proceedings of the American Control Conference*, pp. 594 –599, vol.1, 2001.
[49] A. T. Vermuri, M. M. Polycarpou, and A. R. Ciric, "Fault Diagnosis of Differential-Algebraic Systems," *IEEE Transaction on Systems, Man, and Cybernetics – Part A: Systems and Humans*", vol. 31, no. 2, 2001.
[50] Matlab (2001). Natick, MA: Mathworks.
[51] www.adaptec.com, 2001.
[52] www.ci.nyc.ny.us, 2001.
[53] www.adapt.com, 2001.