A Hybrid Approach to Fault Diagnosis in Teams of Autonomous Systems

Hanieh Agharazi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

May 2013

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Hanieh Agharazi

Entitled: "A Hybrid Approach to Fault Diagnosis in Teams of Autonomous Systems"

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

Master of Applied Science

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

Signed by the final examining committee:

_____ Chair
Dr. R. Raut

_____ Examiner, External
Dr. A. Dolatabadi (M.I.E.) To the Program

_____ Examiner
Dr. A. Aghdam

_____ Supervisor
Dr. S. Hashtrudi Zad

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

_____20_____

_____
Dr. Robin A. L. Drew
Dean, Faculty of Engineering and
Computer Science

ABSTRACT

A Hybrid Approach to Fault Diagnosis in Teams of Autonomous Systems

Hanieh Agharazi, M.A.Sc.

Concordia Unviersity, 2013

Discrete event systems (DES) are dynamical systems equipped with a discrete state set and an event driven state transition structure. An event in a DES occurs instantaneously causing transition from one state to another. DES models have emerged to provide a formal treatment of many man-made systems such as automated manufacturing systems, computer systems, communication networks and air traffic control systems.

In this thesis, we study fault diagnosis in teams of autonomous systems. In particular, one consider a team of two spacecraft in deep space. The spacecraft cooperate with each other in leader-follower formation flying. Formation flying demonstrates the capability of spacecraft to react to each other in order to maintain a desired relative distance autonomously without human intervention. In the system considered here, instruments (actuators and sensors) may fail and cause error. Because of the communication delays in deep space, each entity should be able to diagnose the failure and decide how to reconfigure itself.

Basically, fault diagnosis in such systems requires information exchange between the autonomous elements of the team. The exchanged information for example may include position and velocity data. Our goal in the thesis is to propose a method for fault diagnosis with reduced information exchange. One solution is to transmit only discrete event information between autonomous systems. Transmission of discrete event data occurs less frequently than the transmission of continuous streams of data. The discrete event data may include high level supervisory commands issued every now and then and discretized values of continuous data that

are transmitted only when a continuous-variable data (such as angle or acceleration) crosses the threshold. The fault diagnosis scheme proposed in this thesis is an adaptation of hybrid fault diagnosis for distributed autonomous systems.

This system is simulated using MATLAB/SIMULINK Software and DECK Toolbox. We examined different maneuvers for spacecraft and investigated the effect of faults on the overall system and the performance of our designed fault diagnoser.

To my parents and my husband Hamid

for their constant support and unconditional love

ACKNOWLEDGEMENTS

This work would not be possible without the help and encouragement I received from several people. First and foremost, I would like to express my most sincere gratitude to my supervisor, Dr. Shahin Hashtrudi Zad for his continuous patience, support and guidance in the development of this thesis.

I would also like to thank all my friends who have helped, supported and made my time enjoyable in Concordia University. In particular, I would like to thank Dr. Rasul Mohammadi for providing me with invaluable discussion and sharing the wealth of knowledge and experience.

Most of all, I would like to thank my dear family whom have always been present through out my life in spite of the distance. At last, but definitely not least, I would like to thank my amazing husband, Hamid, for his love, support, patience and encouragement throughout my studies.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

DES	Discrete Event System
EDESA	Extended Discrete Event System Abstraction
FSA	Finite State Automaton
FD	Fault Detection
FDI	Fault Detection and Isolation
DECK	Discrete Event Control Kit
POE	Planetary Orbital Environments
DS	Deep Space

Chapter 1

Introduction

Fault Diagnosis plays an important role in protecting life and property, and increasing reliability and productivity [28]. Therefore, it is of great importance in machinery and management systems such as transportation systems (such as aerospace, automobile), industrial production facilities (i.e. power plant, water treatment plant), household appliances (washer, dryer). In these systems, there are extensive amount of sensors with different types of signals and large number of operational modes. This makes them computationally complex systems and also makes fault diagnosis a really challenging problem. The main issue is the to develop systematic fault detection and isolation techniques to increase the accuracy and reliability along with reducing the cost of maintenance and revisions. As a result, a considerable amount of research has been conducted on fault diagnosis (e.g. [29], [30], [31], [33], [34]).

The behavior of many complex systems, such as spacecraft, can be described in terms of continuous and discrete modes. As a result, they require modeling tools that take into account both of these charactristics and their interactions. Hybrid system models have been developed extensively for modeling such complex systems.

In hybrid systems, the dynamics in every mode evolve continuously until a transition takes the system to a different mode of operation. This transition may

take place autonomously as a result of the continuous evolution of system variables or because of a discrete event such as a supervisory command. These systems have been used extensively by researchers in different engineering fields, as a modeling tool, for developing algorithms in many domains such as control, data management and fault diagnosis (see, e.g. [35], [36], [37], [38], [39]).

In this thesis, we consider fault diagnosis in teams of autonomous systems and in particular a team of two spacecraft in formation flying. We would like the spacecraft to maintain a desired formation as they are deployed in deep space. The components in the spacecraft may fail. In this distributed system, in order to increase the autonomy, each spacecraft should be able to do the diagnosis itself and decide its next move based on the controller commands and diagnosis results. For the purpose of fault diagnosis, a huge amount of information may have to be exchanged between the members of the formation flying. In this work, we propose a new method to decrease the volume of information exchange by transmitting *only* the discrete event information to the other members of the team. For example, the discretized acceleration and orientation data are to be sent to the other spacecraft to be used for fault diagnosis. This will reduce the amount of exchanged data since we transmit acceleration and orientation data when they cross certain thresholds.

In the following, we briefly review the research conducted in the literature as related to the work in this thesis.

1.1 Literature Review

1.1.1 Formation Flying

Formation flying is used to describe the behavior of a set of more than one spacecraft that collaboratively work together as an alternative to a single, larger and

more expensive spacecraft. Cooperative smaller set of spacecraft has many benefits over a single one including simpler designs, higher redundancy due to cheaper replacements and also improved reliability, reduced cost and increased mission performance. However, these benefits come with a new set of challenges such as relative navigation, control and fault diagnosis [9]. The concept of formation flying has been studied extensively in the literature with applications to the coordination of multiple robots [10], [11], unmanned aerial vehicles [12], [13] and satellites [8].

Formation flying can be divided into two main categories based on the ambient dynamic environment: **Deep Space (DS)** and **Planetary Orbital Environments (POE)**. In deep space the formation is in heliocentric orbit rather than earth orbit. In this case, the formation flying control focus on the tracking of spacecraft relative position and attitude [16]. On the other hand, in POE, the spacecraft are around the earth orbit and thus they are subject to significant environmental disturbances. In our work, we consider formation flying in deep space and assume that the earth gravity is negligible.

The control architecture plays a key role in the performance of the formation. According to [17] in spacecraft formation flying at least two of them use an active control scheme to maintain the relative positions. An alternative definition is given in [8] where formation flying is defined as a set of more than one spacecraft in which any of spacecraft dynamic states are coupled through a common control law. This definition is completed with two conditions: at least one spacecraft must (I) track a desired state profile relation to another member, and (II) the associated control law should, at the minimum, depend upon the state of this other member. This common control law can be understood as the formation flying control.

The formation flying control architectures have been classified in the literature into five main categories based on the topology of communication between the spacecraft controllers as follows [8].

- **Multi-Input/Multi-Output**: Controllers are designed based on the dynamic model of the entire formation. Therefore, the formation is considered as a multi-input, multi-output plant.

- **Leader-Follower**: In this architecture one spacecraft, referred to as the leader, moves based on its own absolute position and the others, known as the followers, move based on their relative position with respect to the leader. This architecture is also known as Chief-Deputy, Master-Slave or Target-Chase.

- **Virtual Structure**: This architecture considers all spacecrafts as rigid bodies embedded in a large virtual body. The whole motion of the system takes into account each individual spacecraft's motion in order to determine the whole rigid body motions.

- **Behavioral Architecture**: In this method , there is no globally accepted definition of a *primitive behavior* according to [18]. Instead, each spacecraft is designed to have its own objective and behavior. The outputs of multiple controllers are then combined together to form a control signal with consideration of the behavioral difference among the spacecraft.

- **Cyclic**: The formation is similar to the Leader-Follower architecture in the sense that each spacecraft has its individual controller but it is different in the sense that they are not in a hierarchical arrangement. The motion of each spacecraft is controlled with respect to its neighbors, not with respect to the leader.

Leader-Follower is the most studied formation flying architecture among the methods mentioned above. In our work, we consider a group of spacecrafts in formation flying with this control architecture.

In practice, it is desired to minimize the amount of communication among the spacecraft, as the data transmission and communication delay are grave issues in

deep space applications. Therefore, it is better to have some form of decentralization in control structure which has a lower communication requirement.

On the other hand, one of the main issues in all the mentioned architectures is the autonomy and robustness to faults in the formation. The system needs to first detect and isolate the presence and location of any faults, and then recover by reconfiguring the controllers.

One of the common methods of Fault Detection and Isolation (FDI) is the design of detection filters or using observers such as '*Luenberger Observers*' in a deterministic setting whose residuals change when a fault occurs in the system [19], [20], [21]. In the next section an overview of fault diagnosis and the various methods of FDI are presented.

1.1.2 Fault Diagnosis

The term ***fault*** refers to a non permitted derivation of the behavior of components of the system. A valve becoming stuck-closed, a bias in the sensor readings or a loose connection in an electric circuit are examples of faults. Faults can be either **permanent** or **non-permanent**. After the occurrence of permanent faults, the system remains in faulty condition indefinitely while for non-permanent faults, the system may recover and return to the normal condition. A broken valve can be an example of a permanent fault and a loose wire in an electrical system may cause a non-permanent fault.

With the possibility of failure occurrence, **Fault Diagnosis (FD)** is used to improve the reliability of a system. A typical fault diagnosis system detects the faults and isolate the source of failure before it causes a disaster in the system. Fault diagnosis techniques are classified into **model-free** and **model-based** methods.

Expert systems and **hardware redundancy** are two commonly used techniques in model-free methods in which it is difficult to obtain a model for the plant.

In expert systems, experience and knowledge of experts are stored as rules and then an inference engine is used for fault diagnosis. Gathering the required expertise and information for building an expert system is usually difficult and time-consuming [23], [24]. As a result, there will be no guarantee for the completeness of the resulting diagnostic rules. However, in cases where models are not easy to develop expert systems can be very effective. In hardware redundancy, however, multiple sensors are used to measure the same variable. Their outputs are then compared and the final value for that variable is determined by a *voter*. In case of failure, the faulty sensor can be detected by comparing its value with other sensor values. Although, this method is simple and fairly reliable, it is expensive. Moreover, it is not suitable for detecting common-cause sensor failures.

In addition to the above model-free methods, several model-based techniques for fault diagnosis have also been proposed in the literature. In a model-based method, the observed behavior of the system is compared with the expected behavior from system model. The condition of the system, normal or faulty, is then concluded from this comparison. A large class of model-based techniques rely on parameter estimations and state estimations for continuous variable systems. In these methods, the system is modeled using differential and difference equations [25], [26] and [27]. In the following, we briefly review the model-based techniques which are suitable for diagnosing failures in **Discrete Event Systems (DES)**.

Fault Diagnosis in Discrete Event Systems

A discrete event system is a dynamic system with discrete input and output, whose behavior can be described in terms of discrete state transitions. Fault diagnosis in these systems was prompted by some works in Automatic Control Systems [40], [41], [42] and Artificial Intelligence [43]. In [40], Lin proposed a discrete event approach for fault diagnosis assuming that each component has some normal and faulty states,

and uses the output at each state for fault diagnosis. In this **state-based approach**, a sequence of control commands are issued and the current condition of the system, normal or faulty, is determined by observing the output of the system. The system is said to be *online-diagnosable* if there exists a control sequence that diagnoses the system. This method is an *active state-based* diagnosis approach as a sequence of control commands are generated as the input to the system.

In [31], Hashtrudi Zad et al. proposed a *passive on-line* method for fault diagnosis in DES systems by constructing a fault diagnoser using a state-based approach. The objective is to use the output sequence to determine the current condition of the system; normal or faulty. This diagnoser can detect and isolate the failure by assuming that a failure is diagnosable if it occurs before the diagnoser initialization. To reduce the number of diagnoser states and the computational complexity of the diagnoser design, a model reduction method has also been introduced.

In contrast to the above-mentioned methods, there is an **event-based approach** proposed by Sampath et al. in [42], [44]. In this method, the diagnoser does not generate any control commands as the inputs to the system and only relies on event observations. The faults are assumed to be unobservable. The diagnoser then acts like a sensor to detect these unobservable faults.

Fault Diagnosis in Continuous-Variable Systems

Models with continuous variables are also studied in literature (see, e.g. [22], [46], [45], [47], [48], [5]). Most of these approaches rely on *Analytical Redundancy* where the expected outputs of the system are obtained analytically based on the mathematical model of the system and then compared with sensors measurements. The resulting difference is called the **Residual**. Probing the value of residuals, one can determine the presence of a fault. The residual is zero, or close to zero, in the absence of faults while it will be nonzero if a fault happens [49].

Parity Space and **Fault Detection Filters** are two approaches that are used frequently. Parity approach is based on the consistency test of parity equations and are built using the mathematical model of the system and the sensor measurements (see, e.g. [22], [50], [51], [52], [46], [53], [54], [55]). In [50], Chow and Willsky derived the parity equations from the state space model of the system. One can detect the fault from the inconsistency of parity equations. In FD filter approach, special filters are constructed for fault detection and isolation based on both linear (see [5], [56]) and nonlinear system models [6]. In order to decouple the faults, the effects of different faults are mapped into different directions/planes and stored in the residual vector space. Unique fault isolation can then be obtained independent of their magnitude or the modes (time functions) [22].

Fault Diagnosis in Hybrid Systems

The term hybrid refers to a mixture of two fundamentally different forms of dynamics, discrete-event and continuous variable (e.g. differential equation). Some conventional approaches for fault diagnosis in hybrid systems are based on discrete abstraction of the continuous dynamics. For instance, in [60], the continuous state of the system is quantized and discrete methods are applied for fault diagnosis.

In [32], Hashtrudi Zad et al. extended their diagnosis method for DES in [31] to hybrid automata. They examine the question of whether or not a high-level DES model contains enough information about the low-level hybrid model by introducing the notion of consistency and defining a set of sufficient conditions for that. These models are called consistent if the analysis and design based on the high-level and low-level models yield to the same result. An output, from a set of symbols, is assigned for each state of the hybrid automata and are assumed to be constant at any discrete mode.

In [57], the hybrid fault diagnosis is performed by hybrid structure hypothesis

testing. In this case, the occurrence of a fault can be sensed by measuring system variables at the time the faults occur and signaling this occurrence by observable events. Two diagnosers are designed for continuous and DES levels of the system. If an event is observed, the DES level diagnoser generates a discrete state estimate of the system. The continuous level diagnoser will then perform hypothesis tests, for example residual tests, of the discrete state estimate and generates sub-diagnosis statements regarding the faults at the continuous dynamics. The final diagnosis statement is then produced by a decision logic unit. In [57], it is implicitly assumed that discrete events occurring in the system are observable which cannot be held in many control applications.

As an example of a two level hybrid fault diagnosis, consider the system in [58]. The authors developed a hybrid fault diagnosis method for a team of UAVs using the cooperative characteristics of a team of agents to detect and isolate the faults. Their system consist of two fault diagnosis units: a low-level, agent level, which uses the classical diagnosis techniques and a high-level, team level, which is formulated in the DES framework. First, the state space equations for aircrafts are linearized. Linear observers are then formulated and residuals are generated such that when there are no faults, the residuals generated by the observer will approach to zero asymptotically. In their method, they have used semi-decentralized observers for a more accurate fault detection. However, depending on the team structure, when the agents' locations change, some of these semi-decentralized observers might no longer perform better than fully decentralized ones. Because of this shortcoming, as a remedy, a high-level DES supervisor uses different sets of semi-decentralized observers to detect the faults according to the existing team structure.

Other approaches for hybrid fault diagnosis are, but not limited to, abstracting the systems with a Timed Petri-Net Model in [38], abstractions of the continuous dynamics by Temporal Causal Graphs in ([59], [61]) and fault diagnosis based on

Sequential Monte Carlo (SMC) in [39].

In this work, we focus on the problem of fault detection and isolation in hybrid distributed systems. In this thesis, we intend to develop a framework for fault diagnosis of these systems with minimum communication load between their entities.

Reduction of the communication load has been studied in literature. For instance, the authors in [14] propose an optimal decentralized fault detection scheme for a class of discrete time large scale systems with limited network communications among subsystems by taking the advantage of network communications. Due to the limited bandwidth of the network, the communications should be limited and usually only one subsystem can access the network at the same time. As another example, in [15], the communication load between the entities is lowered by restricting the communication to only local information in continuous format to improve the time and cost. In our work, we would like to do the fault diagnosis with minimum information exchange while using both local and global information.

1.2 Thesis Contributions

As mentioned before, the problem of fault diagnosis in a team of autonomous systems is studied with the objective of reducing the communication among the members of the team. A hybrid approach is proposed where local continuous information along with discrete event information transmitted from other members of the team are used for fault diagnosis. Transmitting discrete information requires less communication load than sending continuous variable signals. This has the advantage of reducing the rate of information exchange. The exchanged information could include supervisory commands and discretized sensor readings (i.e. reporting sensor reading when it crosses a threshold).

One of the applications of this method could be in '*Safe Mode*' when minimal

communication and energy consumption is desirable. Another application is in systems with restricted communication because of the environmental conditions. In this thesis, this method is applied to a group of two spacecraft in a *Leader-Follower Formation Flying* format with possible failures in the sensors and actuators. To our knowledge, this approach to reduce the communications has not been explored in the literature.

1.3 Thesis Outline

In Chapter 2 we describe the method for Hybrid Fault Diagnosis with a simple example. We start this chapter by a brief introduction on Discrete Event Systems and Hybrid Modeling. After that, we describe the method of hybrid fault diagnosis and build a hybrid diagnoser by integrating the continuous-variable and discrete-event information.

In Chapter 3 we propose our hybrid fault diagnosis framework. Then, we introduce our system of Formation Flying Spacecraft with translational motion in one axis and describe the system components and review the dynamics of the system. The DES models of the system are built and the proposed hybrid fault diagnoser is made. We test the performance of the proposed method by applying different maneuvers to the system.

In Chapter 4 we consider the case when the spacecraft in the previous chapter also have rotational movement around a fixed axis. The new system is modeled and the proposed hybrid diagnosis method is applied to this system. Finally, in Chapter 5, we present a summary of our results and discuss directions for future research.

Chapter 2

Fault Diagnosis in Hybrid Systems

In this chapter, we apply the fault diagnosis method in hybrid systems in [7]. In this approach diagnosis is based on both continuous and discrete information. Diagnosis purely based on discrete model cannot isolate the failures that result in small (continuous) changes in the system output. So they have limitations in many complex systems [2]. On the other hand, a purely continuous approach may lead to very complex nonlinear relationships that are difficult to analyze. Moreover, due to the limitations on sensor implementation, some continuous variables may not be measurable and therefore, fault diagnosis based on purely continuous dynamics may not be always possible or necessary. In a hybrid approach, the information available at the DES level are integrated with the information coming from the continuous dynamics through the continuous sensors.

In this chapter, we explain the method of hybrid fault diagnosis in [7] using a running example. We first begin this chapter by a brief introduction to Discrete Event Systems (DES) in Section 2.1. We will then present a hybrid system using an example in Section 2.2 and perform the hybrid fault diagnosis on it in the following section. We will conclude this chapter by presenting a summary.

2.1 Discrete Event Systems

Discrete event systems (DES) are dynamical systems equipped with a discrete state set and an event driven state transition structure. An event in a DES occurs instantaneously causing transition from one state to another. DES models have emerged to provide a formal treatment of many man-made systems such as automated manufacturing systems, computer systems, communication networks, air traffic control systems, integrated transport systems and healthcare systems.

There are several approaches to model a DES such as Finite State Automata (FSA), Petri Nets, Queuing Networks and Pseudo Codes. Automata theory, [3], provides one of the most comprehensive sets of mathematical tools for studying discrete event systems. In this dissertation, we use Moore finite state automata to model DES. First we review some basic definitions and operations on automata.

Languages and Finite State Automata

The dynamics of a discrete event system can be represented by **sequences** of events. A sequence is also called a string or a word. The empty string, denoted as ϵ , is a string without any events. If we consider the set of events as an **alphabet**, then a sequence will be a language with words formed from the alphabet. Moreover, the behavior of a discrete event system can be interpreted as the language that it speaks.

Definition 1. *A **language** is any set of finite-length strings, including the empty sequence, from the events in alphabet* Σ*.*

Given an alphabet Σ, we denote the language that includes the empty string and any finite-length string built from alphabet Σ as Σ^*. A language is a set of sequences; therefore the basic set operations of two languages L_1 and L_2 such as intersection ($L_1 \cap L_2$), union ($L_1 \cap L_2$), complement (L_1^{co}) and relative complement ($L_1 - L_2$) apply to them.

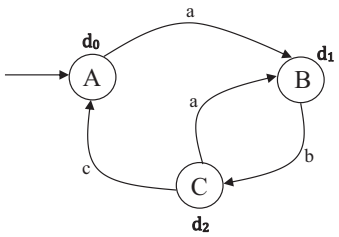


Figure 2.1: A simple FSMA.

Languages can be used as a representation for DES behavior. However, it is difficult to analyze their behavior based on languages. In this case **automata** are a better choice.

Definition 2. *A **finite-state Moore automaton** (FSMA) G is a six-tuple:*

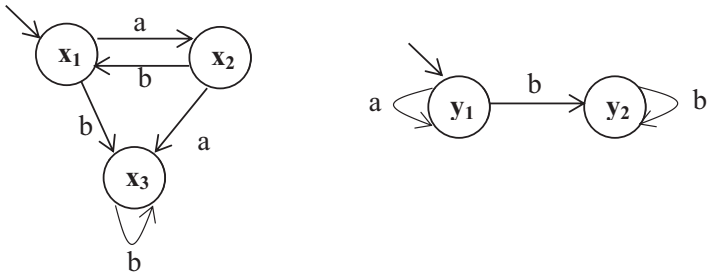
$$G = (Q, \Sigma, T, D, \lambda, q_0) \tag{2.1}$$

where Q is the state set; q_0 is the initial state; Σ is the non-empty event set; $T : Q \times \Sigma \times Q$ is the set of transitions; D is the set of discrete outputs and $\lambda : Q \rightarrow D$ is the output map.

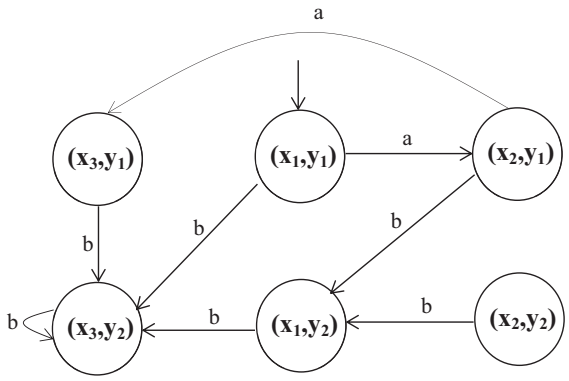
Example 1. *A simple FSMA is shown in figure 2.1. Here,* $Q = \{A, B, C\}$, $\Sigma = \{a, b, c\}$, $q_0 = A$, $T = \{(A, a, B), (B, b, C), (C, a, B), (C, c, A)\}$, $D = \{d_0, d_1.d_2\}$, $\lambda(A) = d_0$, $\lambda(B) = d_1$ *and* $\lambda(C) = d_2$.

Synchronous Product

The **synchronous product** of two automata G_1 and G_2 denoted by $G_1||G_2$ or **sync**(G_1,G_2) models the joint operation of automata ([62], [63]). This is a commutative and associative operation. The synchronous product of three automata G_1,



(a)



(b)

Figure 2.2: Synchronous Product Example: (a) two automata G_1 and G_2, (b) **sync**(G_1,G_2)

G_2 and G_3 is defined as:

$$\mathbf{sync}(G_1, G_2, G_3) = \mathbf{sync}(G_1, \mathbf{sync}(G_2, G_3)) = \mathbf{sync}(\mathbf{sync}(G_1, G_2), G_3) \qquad (2.2)$$

The synchronous product of more that three automata can be defined similarly. Figure 2.2 illustrates the synchronous product of two automata G_1 and G_2.

2.2 Hybrid System Modeling

In this section, we explain the definition of **Hybrid Automata** and then describe a hybrid system with some faults. A hybrid system is a system which is described

Figure 2.3: A simple spacecraft with two thrusters.

by continuous and discrete dynamics. This system can be modeled as a hybrid automata along with its low-level continuous dynamics and high-level DES model. The following hybrid automata definition is a modified form of the definition in [4] as explained in [7].

Definition 3. *A **Hybrid Automata** is a 14-tuple of the form*

$$H = (Q, X, U, Y, FT, Init, S, \Sigma, T, G, \rho, D, \lambda, q_0) \tag{2.3}$$

where Q is the set of finite discrete states; $X \subseteq R^n$, $U \subseteq R^p$ and $Y \subseteq R^r$ are the set of vector spaces of continuous state, control input and output, respectively; FT is the set of m fault types $f^1,\ldots,f^m$ with $f^i(t) \in R$ for $1 \leq i \leq m$; $Init \subseteq X$ is the set of initial continuous states; $S = \{S_q | q \in Q\}$ is the set of dynamic models defining the continuous dynamics of the system; Σ is a set of symbols representing the discrete events labeling the transitions among discrete states; $T \subseteq Q \times \Sigma \times Q$ is the set of discrete transitions; $G: T \times X \times U \rightarrow \{True; False\}$ is the set of guard conditions; $\rho: T \times X \rightarrow X$ is a reset map; D is the set of discrete output symbols; $\lambda: Q \rightarrow D$ is the discrete output map and q_0 is the initial discrete state.

Figure 2.3 shows a spacecraft with two thrusters. These thrusters could be *on* or off moving the spacecraft to the right or left direction on x-axis. They may fail, thruster T_1 may become *stuck-on* and T_2 may become *stuck-off*.

The hybrid automaton for this spacecraft is depicted in Figure 2.4. There are four discrete states for the normal behavior of the thrusters and four discrete

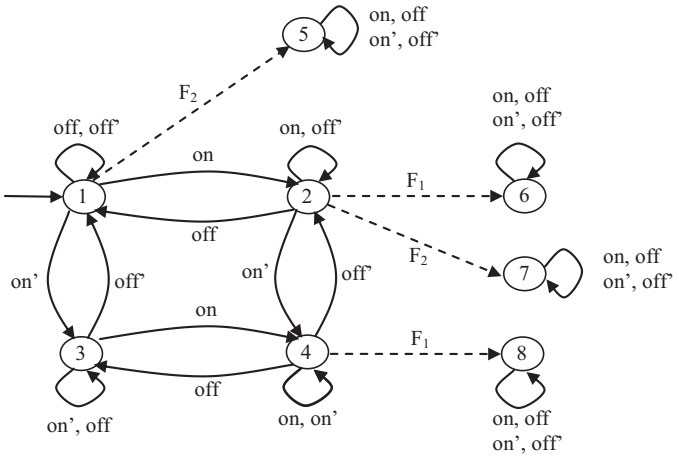


Figure 2.4: A hybrid automaton modeling a spacecraft with two faults.

states for the faulty behavior. Events on/off and $o\acute{n}/of\acute{f}$ correspond to the power condition of thrusters T_1 and T_2 respectively. F_1 and F_2 represent T_1's $stuck-on$ and T_2's $stuck-off$ faults. We assume that the faults are permanent, i.e. the system stays in the faulty state indefinitely.

Assume that S is the set of linear systems modeling the spacecraft dynamics at each discrete state. We can then write S_q, the continuous dynamics at the discrete states q of the hybrid automaton model, as

$$S_q = \begin{cases} \dot{X}(t) = A_q X(t) + B_q U(t) + L_q F(t) \\ Y(t) = C_q X(t) \end{cases} \tag{2.4}$$

where

$$X(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \ , \ Y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \tag{2.5}$$

In the above equation, $U(t)$, $F(t)$ and L_q represent the matrices of thruster power level (zero/one for on/off conditions), thrusters failure and the vector of the fault signatures in the discrete state q, respectively. Also, $x(t)$ is the position of the

spacecraft and $\dot{x}(t)$ is the spacecraft velocity. The output $y_1(t)$ is the position and the output $y_2(t)$ is the velocity of the spacecraft. Therefore, overall, the hybrid model contains an automaton (DES abstraction) in Figure 2.4 plus the continuous variable models in Eq. (2.4) and (2.5).

2.3 Diagnosis of Hybrid Automata

In this section, we describe a hybrid fault diagnosis method for hybrid automata [7]. In this method, the diagnosis is performed by integrating the information from the continuous sensors of the system with the information from discrete sensors at the DES level. In this approach, based on the continuous dynamics of the system, a bank of **Residual Generators** (isolators) are designed to isolate the fault types at the continuous level by producing a residual using the continuous input and output of the system. In [5] and [6], the solvability conditions for the existence of residual generators for fault isolation in linear and nonlinear dynamical systems have been studied.

The framework for designing the diagnoser for this method of fault diagnosis is depicted in Figure 2.5. First, the DES abstraction of residual generator is generated. It will then be integrated with the DES abstraction of the system to construct the **Extended DES Abstraction (EDEAS)** of the system and isolators. The hybrid diagnoser will be then designed based on the EDESA as described in the following steps:

1. Residual generators (isolators) are described by DES models.

2. Appropriate self-loop transitions are added to the DES abstraction of the system (H_{abs}) to make the transitions in isolator's DES model consistent with the system's transition model.

3. We assume that the system stays at each discrete state long enough so that

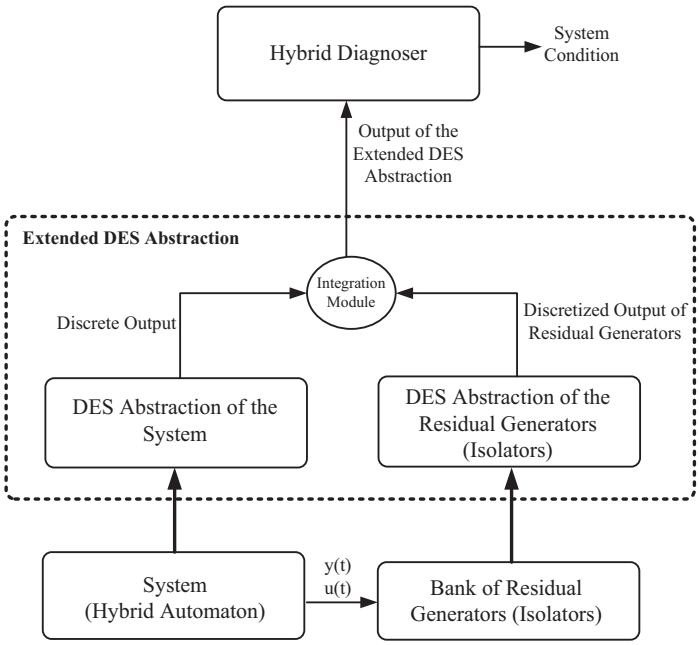


Figure 2.5: The schematic of the hybrid diagnosis framework

the transient response dies out. This assumption is enforced by having each isolator event happen between the occurrence of any two consecutive events in the system.

4. EDESA is constructed by synchronous product of the DES models in steps 1 to 3.

5. Finally, the diagnoser is built based on the EDESA from the previous step using a diagnoser design for discrete-event models.

In the following, we apply these steps to our previous example of the simple spacecraft with two thrusters.

Step 1 - Modeling of Isolators

The isolators are designed by taking the continuous input and output of the system and producing a residual vector to isolate some fault types from the others while

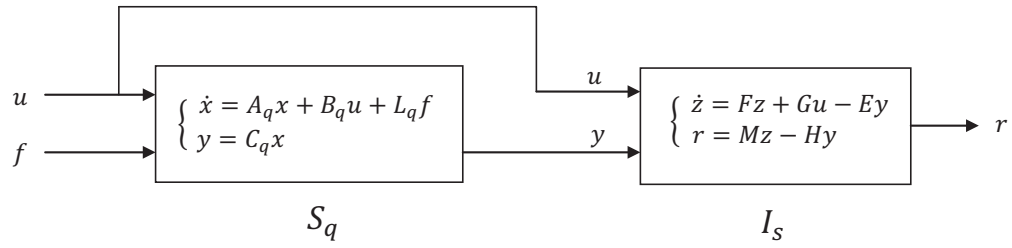
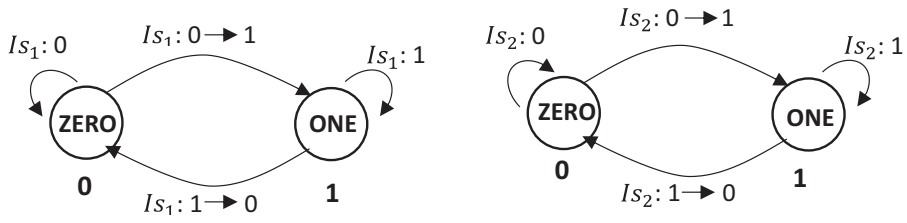


Figure 2.6: System dynamics and Isolator in state q

the system is in one of the discrete states. This is shown in the block diagram in Figure 2.6. This figure shows the continuous dynamics of the system in discrete state q along with the isolator I_s for that state. We assume that at the beginning the isolators are initialized at zero. They will remain at zero as long as no fault is present in the system. If a failure occurs, they will change to a nonzero signal. The discretized value of the isolators is 0 when normal and 1 when the fault has occurred. The design procedure of isolators is outside the scope of this thesis. The reader could refer to [5], [6] and [7] for more details.

The designed isolators are then modeled as a finite state Moore automaton with two states of ZERO and ONE. In our spacecraft example, there are two faults of F_1 and F_2. Therefore, there will be two isolators, Is_1 and Is_2, used for detection and isolation of the respected faults. Figure 2.7 shows the FSA model of both isolators. The events 'Is_i:0→1' and 'Is_i:1→0' represent the transition of residual signal from state ZERO to ONE and from state ONE to ZERO, respectively. The self-loops '$Is_i : 0$' and '$Is_i : 1$' are unobservable events added for design consistency.



(a) (b)

Figure 2.7: The FSA modeling the isolators: (a) Is_1, (b) Is_2

Step 2 - Consistency between the DES Model of System and Isolator

The abstract DES models for the system and isolators capture the interactions between them, i.e. any changes in the system such as mode changes and the occurrence of the faults should lead to the changes in the output of isolators. The DES model of the system is then modified to enforce this consistency requirement by adding appropriate self-loop transitions.

Let H_{abs} be the abstract DES model of the spacecraft example (shown in Figure 2.8). The continuous dynamics of the system are defined at each state in terms of Eq. (2.4).

The modified DES abstraction of the spacecraft example, $\hat{H}_{abs}$, is constructed from H_{abs} by adding certain self-loop transitions at each discrete state as shown in Figure 2.9. For instance, at state q_0, the self-loop transitions are '$Is_1 : 0$', 'Is_1:1→0', '$Is_2 : 0$' and 'Is_2:1→0'. This implies that the isolators Is_1 and Is_2 cannot have the transition from ZERO to ONE as this is a healthy state of the system with no present faults. D_i for i=0,...,3 are the outputs of the DES model at each state.

Step 3 - Enforcing the Assumption by DES Model

As described earlier in this section, we need to introduce a new automaton enforcing our assumption of the time between the occurrence of two consecutive events. The

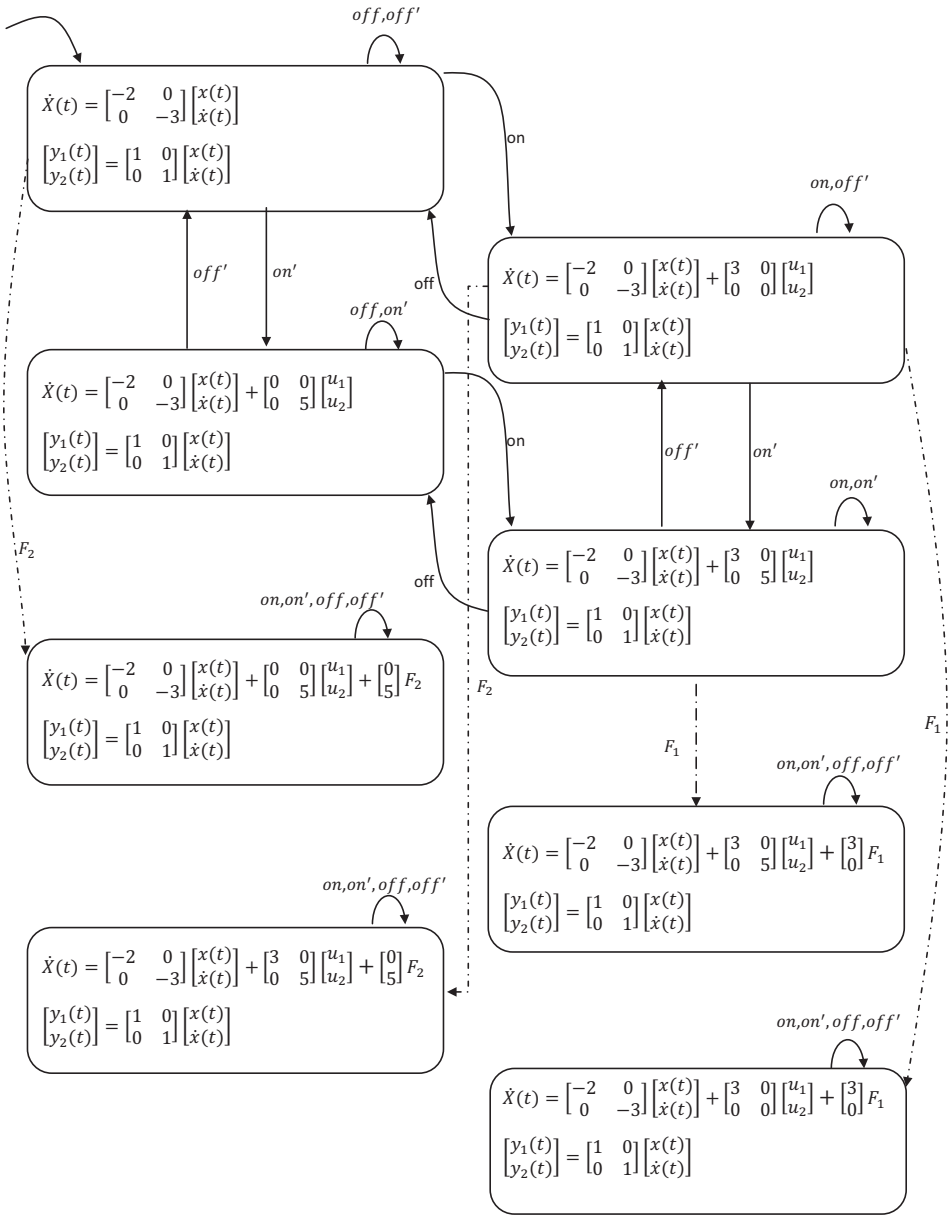


Figure 2.8: H_{abs} - The hybrid automaton model for the spacecraft example.

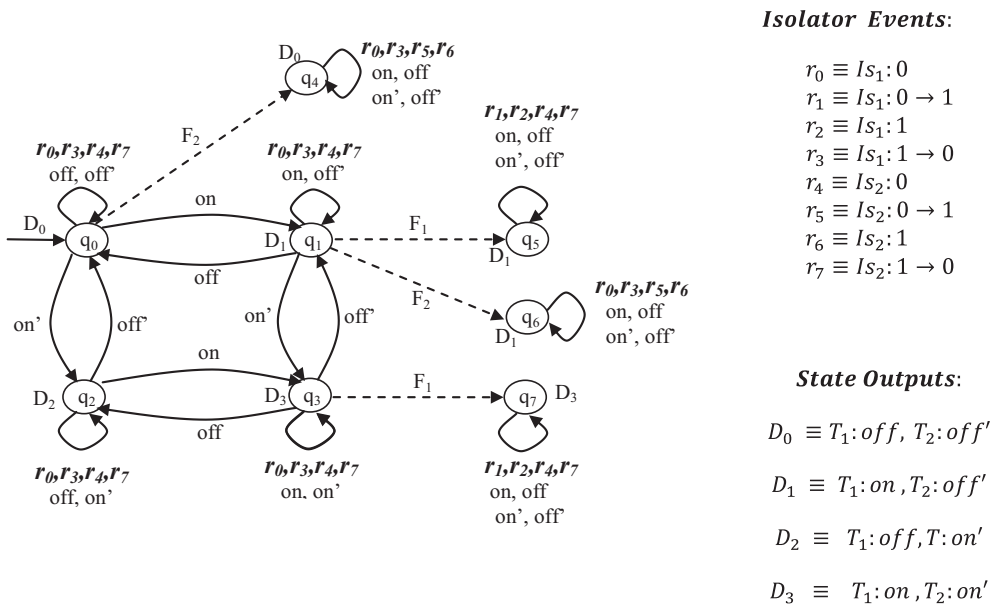


Figure 2.9: $\hat{H}_{abs}$ - The modified DES abstraction for the spacecraft example.

automaton ASM_{Is} shown in Figure 2.10 models the above assumption.

Step 4 - Constructing EDESA

The EDESA of the hybrid system and isolators, denoted as $\tilde{H}$, is defined as the synchronous product of $\hat{H}_{abs}$, the modified DES abstraction, (Is_1,Is_2), the automata modeling the isolators, and ASM_{Is}, the automaton enforcing the assumption. This

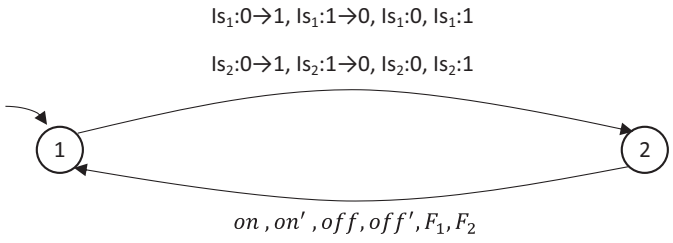


Figure 2.10: ASM_{Is} - The FSA enforcing the assumption.

is shown in Eq. (2.6).

$$\tilde{H} = \mathbf{sync}(\hat{H}_{abs}, Is_1, Is_2, ASM_{Is}) \tag{2.6}$$

Figure 2.11 shows the EDESA of the spacecraft example. The output at each state is the 3-tuple [D_x,r_1,r_2] where D_x was given in Figure 2.9 and r_1 and r_2 are the discretized output of the isolators.

Step 5 - Building the Diagnoser

A part of the diagnoser is designed and shown in Figure 2.12 where the failure mode F_2 is detected. The discrete outputs of the EDESA are observed and based on the EDESA model, the possible states of EDESA and consequently the health condition of the model are estimated. We continue probing the outputs of the model until the condition of the model is defined. Overalls, the diagnoser combines discrete information (thrusters on/off status) and continuous information (obtained from sensors, processed to generate residuals) to arrive at a diagnoser.

2.4 Summary

In this chapter, a method for hybrid fault diagnosis is described with a simple example. This approach develops a systematic method to integrate the information in continuous variable data, such as pressure sensor in a valve, with discrete value data, such as threshold sensors, to achieve more accurate diagnosis.

A bank of residual generators are first constructed to detect and isolate the faults. They will then be abstracted by DES models and integrated with the DES abstraction of the system to construct an extended DES model. Finally, this extended model is used to build the fault diagnoser.

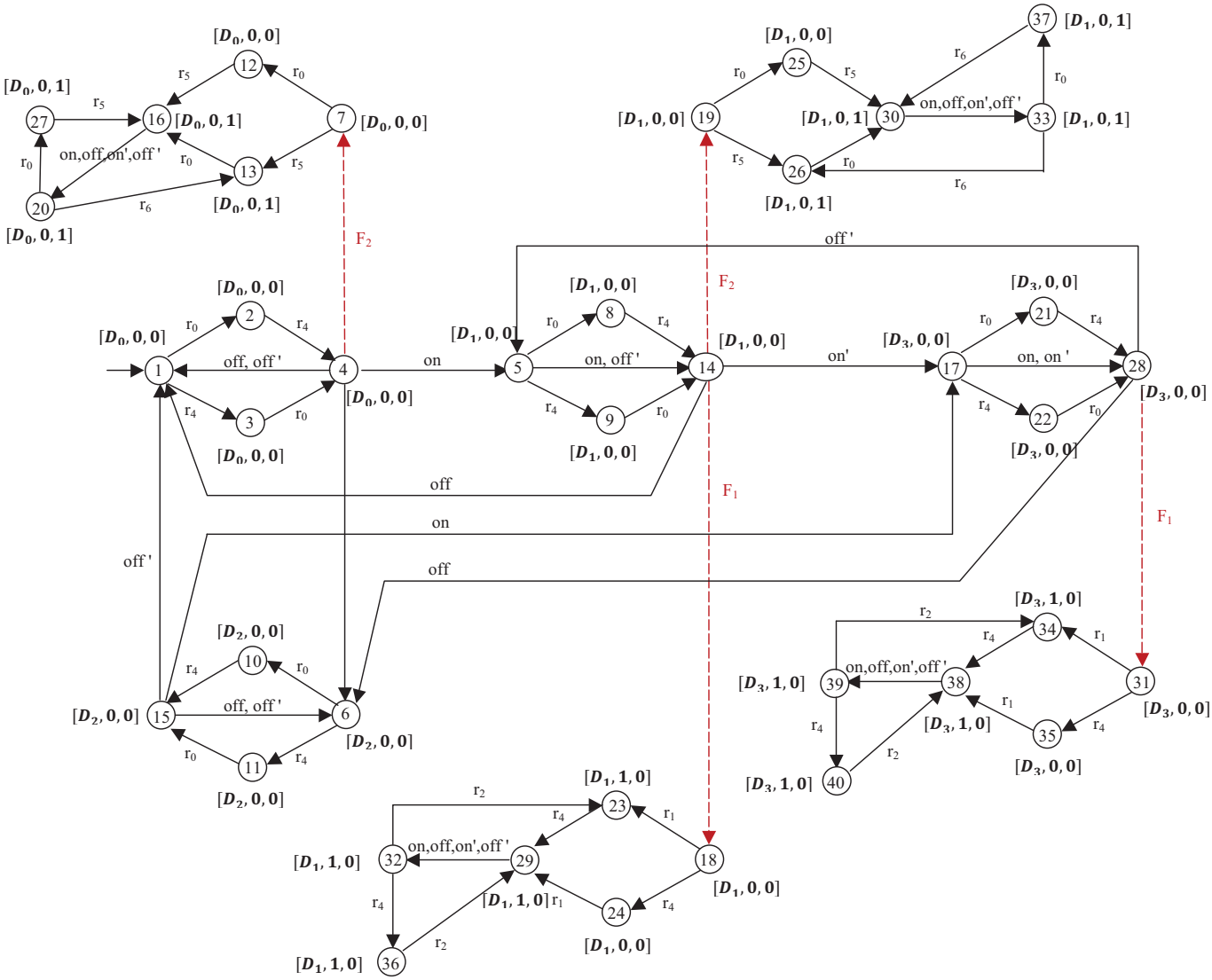


Figure 2.11: *EDESA* - EDESA Model of the spacecraft.

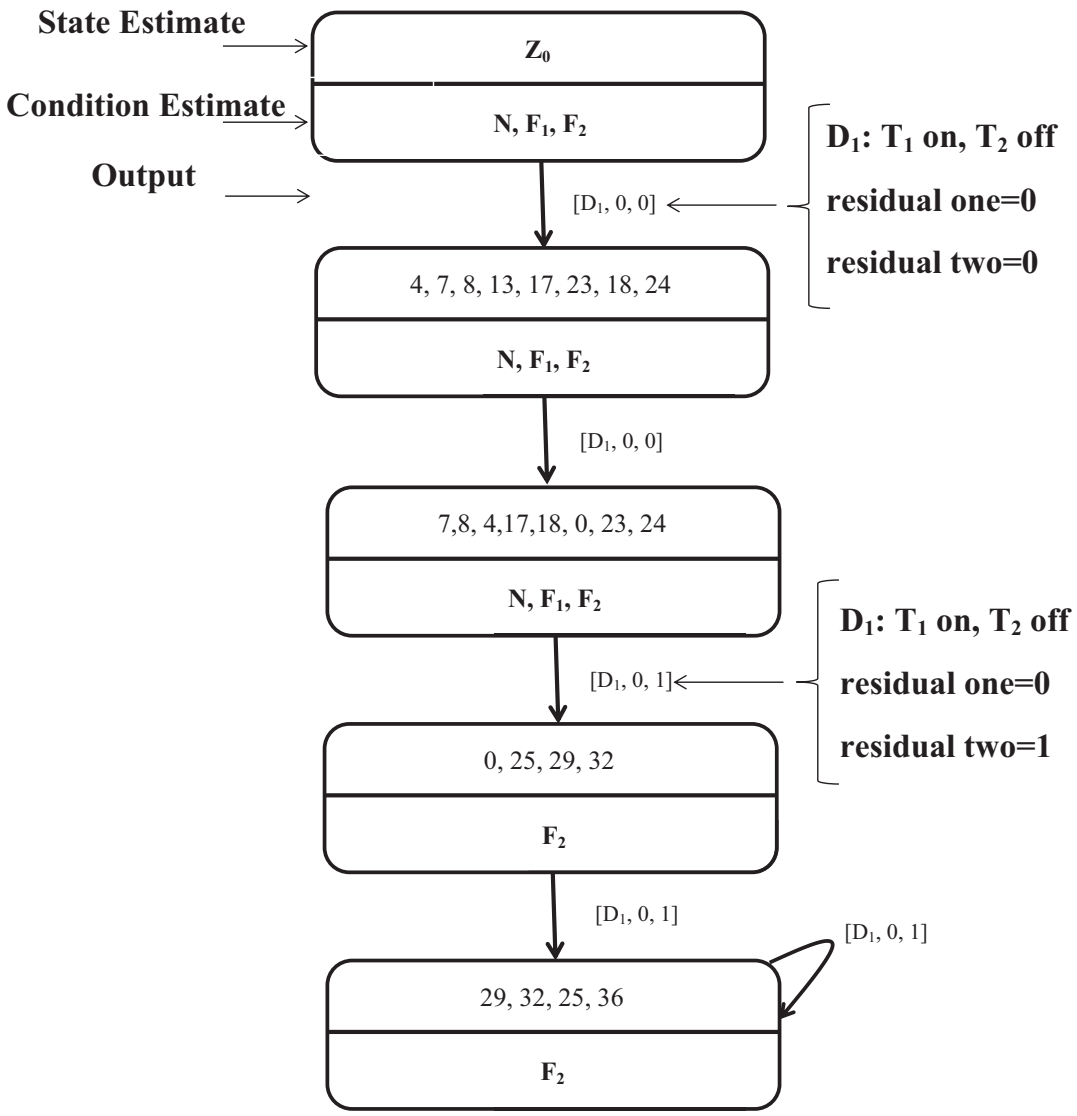


Figure 2.12: Part of the diagnoser constructed for the spacecraft example based on EDESA model (Fig. 2.11)

Chapter 3

Hybrid Fault Diagnosis in Leader-Follower with One-Dimensional Translational Motion

In this chapter, we propose a decentralized implementation of the hybrid diagnosis method described in the previous chapter. This method is applied for fault diagnosis in decentralized systems that are distributed geographically such as spacecraft formations and teams of robots.

3.1 A Decentralized Hybrid Fault Diagnosis Design System

The proposed method in this thesis is illustrated in Figure 3.1. In this scheme, diagnosis is performed by local diagnosers who have access to both local continuous data (e.g. sensor readings) and discrete data (e.g. output of threshold sensors,

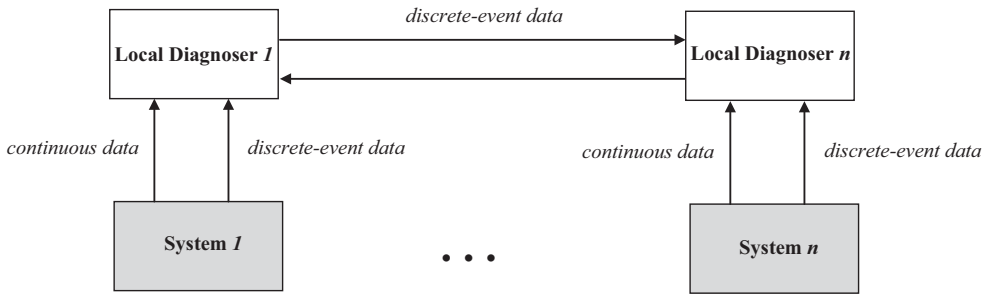


Figure 3.1: Proposed Method Scheme - A Decentralized Hybrid Diagnosis with Discrete Information Exchange

supervisory commands to change mode of operation). In order to reduce the amount of communicated data, the local diagnosers only exchange the discrete data or the discretized versions of the continuous data.

This hybrid approach is applied to a system consisting of two spacecraft in ***Leader-Follower Formation*** to investigate faults in actuators and sensors in the system. Specifically, we explore the design of the local diagnoser for the follower spacecraft. The orientation of spacecraft in space can be described by continuous state space equations. On the other hand, the dynamics of the spacecraft components such as thrusters and accelerometer sensor can be described in terms of discrete transitions, and then can be presented by DES models. We develop a hybrid automaton model representing the behavior of the system and perform a hybrid diagnosis to detect and isolate the faults of the follower spacecraft. As discussed previously, the only data that is to be exchanged between the spacecraft has to be discrete-event data to limit communications. In this study, we explore the possibility and limitations of decentralized fault diagnosis subject to the aforementioned constraint on data exchange between the spacecraft.

The remainder of this chapter is organized as follows. In Section 3.2, we explain our system of a Leader-Follower spacecraft with equations and relationships

between the components. In Section 3.3, we develop a hybrid model for our system. Fault diagnosis based on the constructed hybrid model is explained in Section 3.4. In Section 3.5, we investigate different scenarios for the system and employ the diagnoser to detect and isolate possible existing faults.

3.2 Leader-Follower Formation Flying Spacecraft

In our work, we consider the formation flying spacecraft in deep space, i.e. far from the Earth. In deep space, the gravity is small and almost the same for both spacecraft. There its contribution to equations of motion is small. To maintain the performance of the system, it is important to keep the orientation and relative position of the spacecraft in a desired range.

The fundamental devices for spacecraft stabilization in the space are **Actuators** and **Sensors**. Angles, velocity and acceleration are measured by sensors to determine the position and orientation of spacecraft with respect to a reference system. Some of the most commonly used sensors are *Sun Sensor*, *Earth Sensor*, *Star Sensor*, *Magnetometer*, *Gyroscope*, *Accelerometer* and *Laser Relative Distance*. On the other hand, actuators generate the required torque for the spacecraft to push them around the space. Some of the well-known actuators are *reaction wheel*, *magnetotorquer* and *thrusters*.

In our work, we consider a group of two spacecraft in a Leader-Follower formation as shown in Figure 3.2. We study fault diagnosis in the follower. The leader's components are subject to failure and are monitored by a separate local diagnosis system. So, from the follower's diagnosis perspective, the leader is fault-free. For the purpose of maintaining the relative distance between the spacecraft, we have chosen some of the mentioned actuator and sensors as listed below.

- **Thrusters**, 'on-off' thrusters are used. These thrusters, shown as T_i's in the

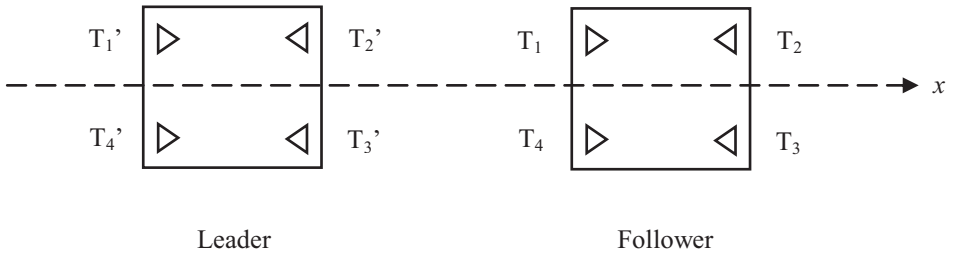


Figure 3.2: Two spacecraft in Leader-Follower formation on a line.

model, are either '*on*' or 'off'. While active, they push the spacecraft to the right or left direction at a constant speed. Four thrusters are placed in four corners as illustrated in Figure 3.2. Leader thrusters are assumed **fault-free** while follower thrusters may fail. We have considered two types of failures for follower thrusters: thruster T_1 may become stuck-on or all thrusters may become stuck-off at the same time. We also assumed that the follower thrusters are more powerful than the leader thrusters to enable it to track the leader's maneuvers.

- **Distance Sensor**, a laser sensor which measures the relative distance between leader and follower spacecraft. In our problem, this sensor is in the follower and notifies the follower supervisor if the relative distance becomes *low*, *high* or *normal*. We assume that the health of this sensor is monitored using hardware redundancy and from the diagnoser's perspective, this sensor is fault-free.

- **Accelerometer**, measures the acceleration of spacecraft. Accelerometer may fail and show the value of the acceleration with a bias. Here we consider negative bias, but positive bias can be handled similarly.

In this chapter, we consider translation motion in one axis for spacecraft. In the next chapter we consider the case in which the spacecraft also have the rotational

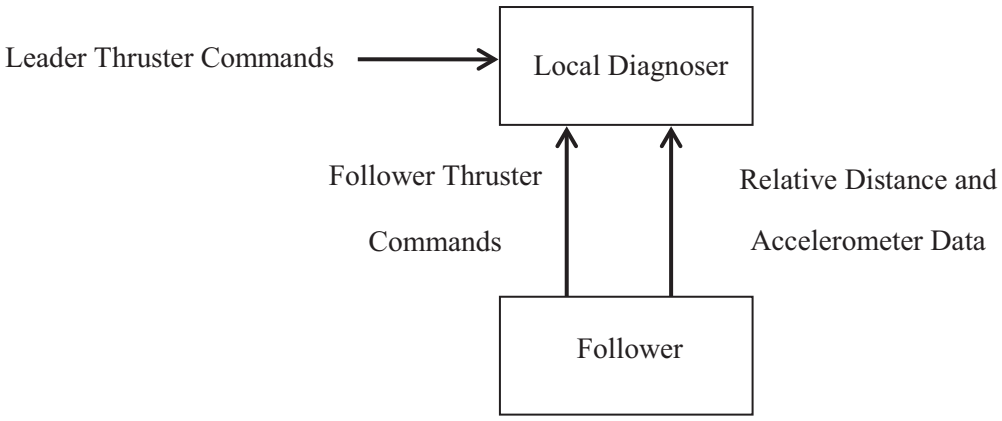


Figure 3.3: The follower and local diagnoser

motion in addition to translational motion.

The objective of this chapter is to develop a hybrid fault diagnoser for the follower spacecraft. We assume that the leader maneuver commands constitute the discrete data which is transmitted from the leader to the follower. The follower supervisor observes the distance sensor output and issues the necessary commands based on this value to maintain the required distance. This is illustrated in Figure 3.3.

Assuming that the spacecraft are initially deployed far from each other, the follower attempts to decrease the distance and moves towards the leader. When the relative distance is detected to be in the desired range, a '*stop*' command will be issued for the follower. On the other hand, if the relative distance becomes less than the desired value, the '*increase distance*' command will be issued. In the meantime, the leader may change its position.

In the presence of a failure, the system shows the same behavior which may lead to a different result. As an example, assume that while the follower is moving, the thruster T_1 fails, stuck-on. If the relative distance becomes low, the '*stop*' and shortly after that the '*increase distance*' commands will be issued for the follower.

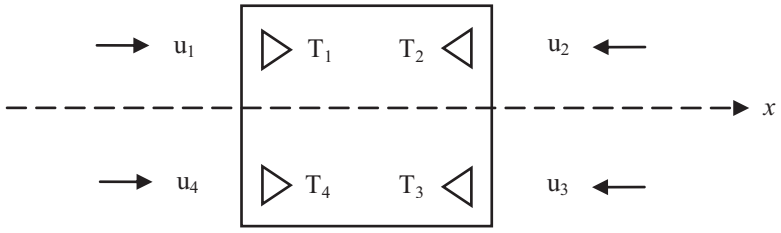


Figure 3.4: The Follower spacecraft

But the follower will continue moving in the previous direction because of the stuck-on T_1. As the faults are assumed to be *permanent*, the system remains in the faulty condition indefinitely.

Based on what we discussed so far, let us model the dynamical behavior of the system. Consider the follower spacecraft model in Figure 3.4. The laws of motion for this spacecraft are

$$\dot{x} = v \tag{3.1}$$

$$M\dot{v} = F \tag{3.2}$$

where x is the absolute position of follower, v is the velocity, M is the total mass of the spacecraft and F is the net force applied to it.

In Figure 3.4, 'u_i' refers to the applied force to the spacecraft resulting from thruster T_i. The arrows show the direction of the force. Therefore, we have

$$F = u_1 - u_2 - u_3 + u_4 \tag{3.3}$$

On the other hand, the gravity is negligible as the system operates in deep space. For simplicity, we assume that the spacecraft mass is normalized to 1. Hence, we have

$$\ddot{x} = F = u_1 - u_2 - u_3 + u_4 \tag{3.4}$$

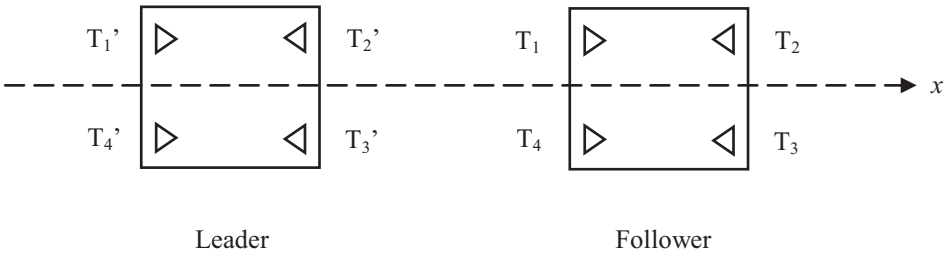


Figure 3.5: Leader and Follower Formation.

3.3 System Model

We first start by modeling individual components, controller and interaction among the components and then use MATLAB/SIMULINK Software along with DECK Toolbox [1] to build the final model.

3.3.1 DES Model of System Components

Based on the discrete behavior of the system components, we can model the whole system by discrete-event models. Our leader-follower system is shown in Figure 3.5.

Follower Thrusters

As spacecraft move to the right or left, we can consider follower thrusters (T_1,T_4) or (T_2,T_3) as single components that push the spacecraft to right or left. These thrusters can be either normal or faulty. Two fault types may occur in the follower thrusters: all thrusters are stuck-off or only T_1 is stuck-on. We have assumed single failure scenario for the system, i.e., no two faults happen at the same time.

The automaton modeling follower thrusters is shown in Figure 3.6. There are seven states and five events 'T_1, T_4-on', 'T_2, T_3-on', 'off', '$all-fail$' and 'T_1-fail'

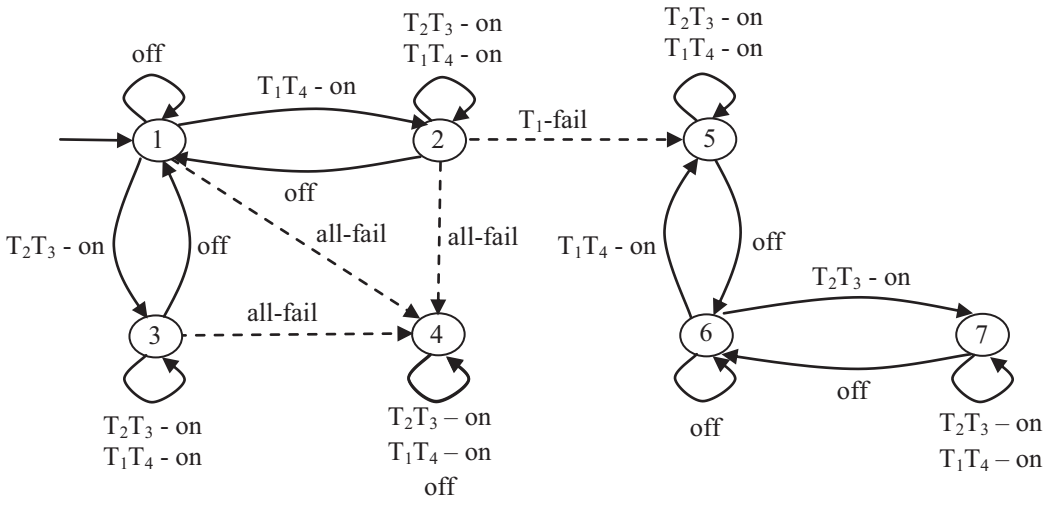


Figure 3.6: FT - Follower Thrusters

representing the commands *on*, off and failures *all stuck* $-$ off and T_1 *stuck* $-$ *on*. For simplicity, fault events are shown by dashed-lines. Event '$all - fail$' enters the system in the mode that all thrusters are stuck-off at the same time. On the other hand, event '$T_1 - fail$' means that thruster T_1 has become stuck-on. We assume, for simplicity, that *all stuck* $-$ off failure may happen when thrusters are either on or off while T_1 *stuck* $-$ *on* can only happen when the pair (T_1,T_4) is on.

We assume that in our system leader components are fault-free. The automaton modeling the leader thrusters is shown in Figure 3.7.

Relative Distance Laser Sensor

This laser sensor measures the relative distance between the leader and follower. In our model, the distance is divided into three discrete levels: **high**, **normal** and **low**. The automaton in Figure 3.8 models the changes in the output of the sensor. Each event represents a transition in the sensor output from one discrete level to another. For example, event '$r : H2N$' is generated when the sensor output changes from

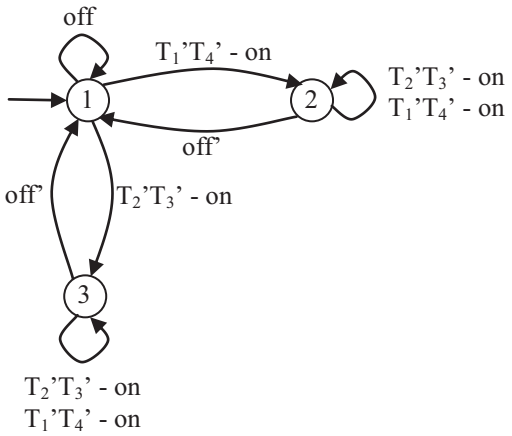


Figure 3.7: *LT* - Leader Thrusters

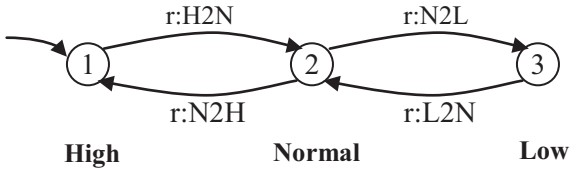


Figure 3.8: *RDS* - Relative Distance Sensor

High to *Low*. We assume that the measured distance is reliable and the sensor is fault-free.

The relative acceleration between leader and follower is calculated by taking the second derivative of relative distance and is used for fault diagnosis. This will be further discussed in section 3.4.1. The relative acceleration can be **zero**, **positive** or **negative**. The discrete event model for acceleration is shown in Figure 3.9. Similar to the relative distance, event '$\ddot{r}$: +20' is generated when the acceleration changes from a positive value to zero.

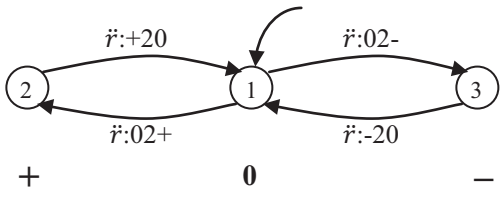


Figure 3.9: RA - Relative Acceleration

Accelerometer

There is an accelerometer in the follower spacecraft that measures its acceleration. For fault diagnosis purpose, the acceleration is divided into 5 levels: **negative**, **very negative**, **positive**, **very positive** and **zero**. It shows *positive* when the spacecraft is moving to the right direction but only thruster T_1 is firing. On the other hand, *very positive* acceleration happens when both thrusters T_1 and T_3 are active and push the spacecraft to the right direction.

Figure 3.10 illustrates the automaton model for the follower accelerometer. There are eight events '$acc : P2Z$', '$acc : Z2P$', '$acc : P2PP$', '$acc : PP2P$', '$acc : N2Z$', '$acc : Z2N$', '$acc : N2NN$' and '$acc : NN2N$' representing the transition of accelerometer output between five levels. For example, event '$acc : Z2P$' is generated when the accelerometer output changes from *zero* to *positive* and so on. The accelerometer may encounter loss of effectiveness as defined by the event '$acc - fail$'. It means that there may be a *bias* in the measured values. We assume that the accelerometer shows a value less than the actual value. For example, it may show *positive* instead of *very positive*.

Single-Failure Scenarios

In this thesis, the system is examined in a single-failure mode, i.e. only one fault at a time could happen. We assume that faults are permanent. This means that if the

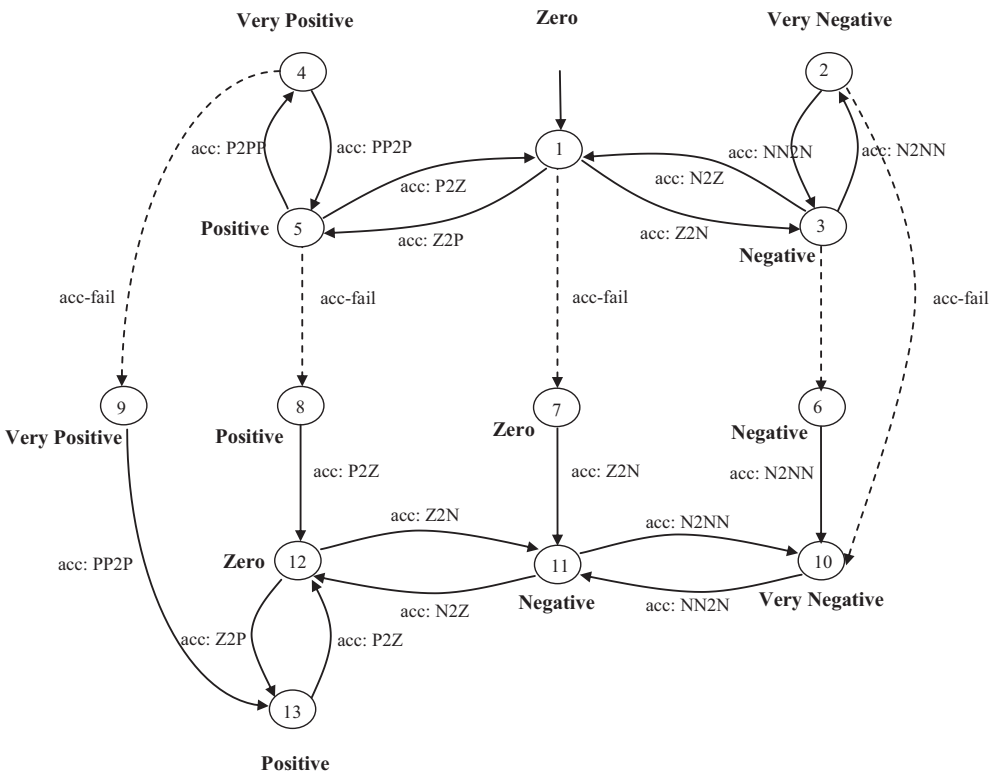


Figure 3.10: *ACC* - Follower Accelerometer

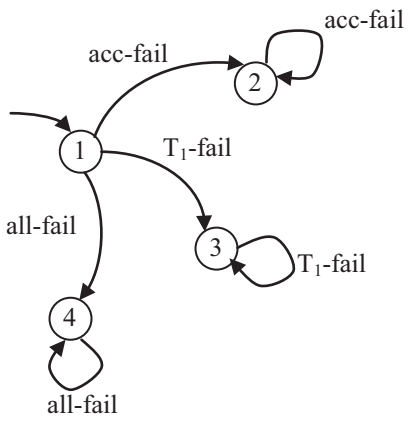


Figure 3.11: $FSCN$ - Single Failure Scenarios

system enters a failure mode, it stays in that failure mode forever. This assumption can be included in the model using the discrete event model in Figure 3.11.

3.3.2 Interactions Among the Components

Due to the interactions among the components, changes in status of one may affect other's behaviors. The interactions considered in our system are:

- Interactions among follower thrusters and accelerometer,
- Interactions among leader thrusters, follower thrusters and their relative acceleration.

Interactions among follower thrusters and accelerometer

As discussed in the previous section, the acceleration measured by accelerometer depends on the number of active thrusters and their health status. The interaction model is obtained by first forming the synchronous product of the DES model of

accelerometer, follower thrusters and fault scenarios

$$INTFACC = \mathbf{sync}(FT, ACC, FSCN) \tag{3.5}$$

where FT is the model of follower thrusters, ACC is the model of accelerometer and $FSCN$ is the single failure scenario model.

Next, selfloops are added to the interaction model as shown in Figure 3.12. For example, in the state $(2, H)$, the thruster pair (T_1, T_4) is active and the accelerometer is healthy. In this case, the measured acceleration is *very positive*. The selfloops represent the events in which the accelerometer output changes to *very positive* or PP.

Interactions among follower thrusters, leader thrusters and relative acceleration

The relative acceleration is calculated by taking the second derivative of relative distance between leader and follower and is a function of thruster pairs (T_1, T_4), (T_2, T_3), ($T_1{'}$, $T_4{'}$) and ($T_2{'}$, $T_3{'}$) and their health status. The relative acceleration can be zero, positive or negative depending on the leader and follower moving direction with respect to each other. We define this value as the acceleration of follower with respect to the acceleration of the leader. For example, if the leader is accelerating to the right direction while the follower is accelerating to the left, the relative acceleration will be negative. The relative acceleration with respect to the state of leader and follower thrusters is shown in Table 3.1. For example, in state (2,1), where the leader thruster pair ($T_1{'}$, $T_4{'}$) is on and all the thrusters in the follower are off, the relative acceleration is negative.

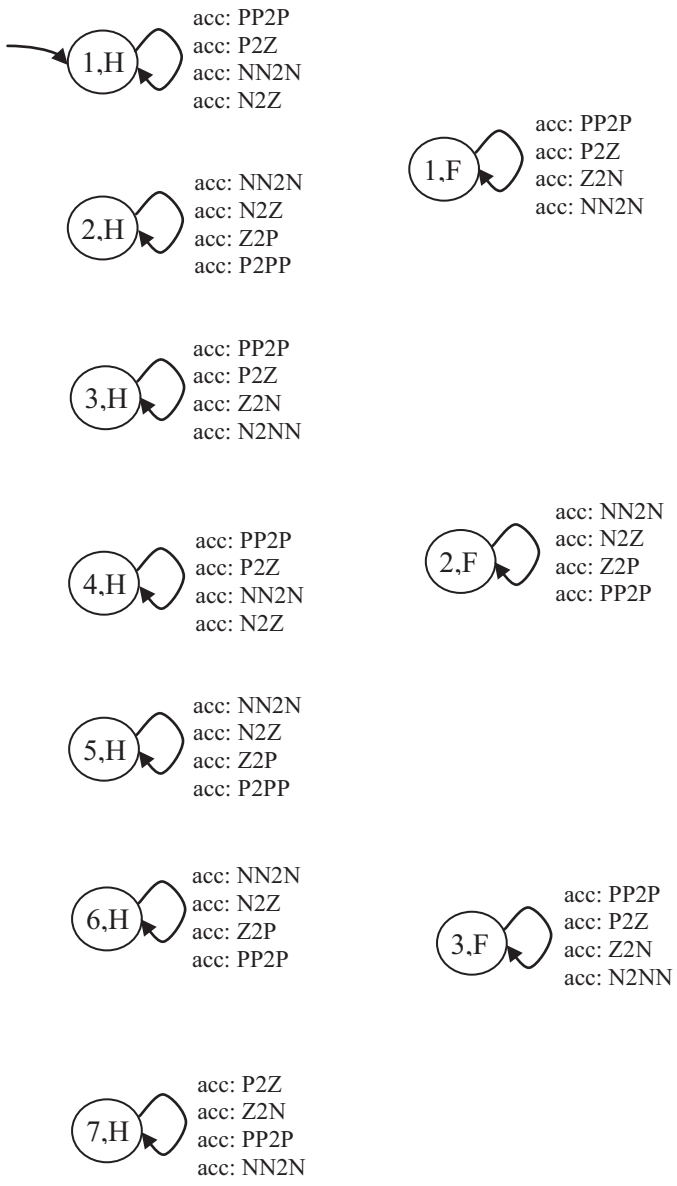


Figure 3.12: Selfloops for Interactions among Follower Thrusters and Accelerometer

Table 3.1: The Relative Acceleration with respect to the Follower and Leader Thrusters

State Number (L,F)	**Leader State**	**Follower State**	**Relative Acceleration**
(1,1)	All Thrusters Off	All Thrusters Off	0
(1,2)	All Thrusters Off	(T_1, T_4) On	+
(1,3)	All Thrusters Off	(T_2, T_3) On	-
(1,4)	All Thrusters Off	All Thrusters Fail	0
(1,5)	All Thrusters Off	T_1 - fail and (T_1, T_4) On	+
(1,6)	All Thrusters Off	T_1 - fail and other Thrusters Off	+
(1,7)	All Thrusters Off	T_1 - fail and (T_2, T_3) On	-
(2,1)	($\acute{T}_1$, $\acute{T}_4$) On	All Thrusters Off	-
(2,2)	($\acute{T}_1$, $\acute{T}_4$) On	(T_1, T_4) On	+
(2,3)	($\acute{T}_1$, $\acute{T}_4$) On	(T_2, T_3) On	-
(2,4)	($\acute{T}_1$, $\acute{T}_4$) On	All Thrusters Fail	-
(2,5)	($\acute{T}_1$, $\acute{T}_4$) On	T_1 - fail and (T_1, T_4) On	+
(2,6)	($\acute{T}_1$, $\acute{T}_4$) On	T_1 - fail and other Thrusters Off	-
(2,7)	($\acute{T}_1$, $\acute{T}_4$) On	T_1 - fail and (T_2, T_3) On	-
(3,1)	($\acute{T}_2$, $\acute{T}_3$) On	All Thrusters Off	+
(3,2)	($\acute{T}_2$, $\acute{T}_3$) On	(T_1, T_4) On	+
(3,3)	($\acute{T}_2$, $\acute{T}_3$) On	(T_2, T_3) On	-
(3,4)	($\acute{T}_2$, $\acute{T}_3$) On	All Thrusters Fail	+
(3,5)	($\acute{T}_2$, $\acute{T}_3$) On	T_1 - fail and (T_1, T_4) On	+
(3,6)	($\acute{T}_2$, $\acute{T}_3$) On	T_1 - fail and other Thrusters Off	+
(3,7)	($\acute{T}_2$, $\acute{T}_3$) On	T_1 - fail and (T_2, T_3) On	+

3.3.3 Follower Supervisory Controller

The relative distance between leader and follower should remain in a certain range. The supervisory controller generates a sequence of events for the system in order to maintain this distance. For example, it commands the follower to increase its distance from the leader when the relative distance becomes low. The controller events for maintaining this distance are shown in Figure 3.13. The events '*decrease_distance*' and '*increase_distance*' are generated whenever the relative distance sensor output changes from normal to high and normal to low, respectively. Suppose the relative distance becomes low. The supervisory controller commands the follower to increase the distance. The method of firing the thrusters is ***bang-bang***. It means that to increase the distance, the sequence of generated events are:

1. Turn on the thrusters T_1 and T_4

2. wait for time τ

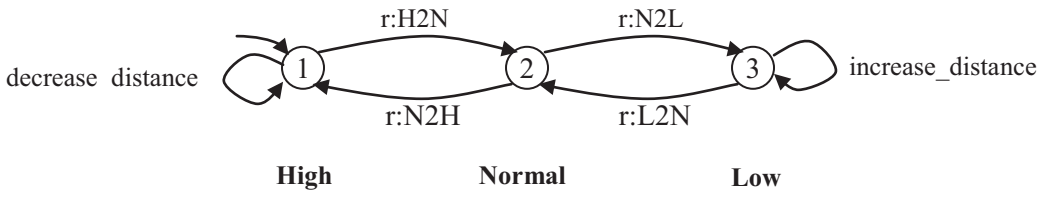


Figure 3.13: *FollowerCommands* - Follower Commands

3. Turn off the thrusters

4. wait for time τ

5. Turn on the thrusters T_2 and T_3

6. wait for time τ

7. Turn off the thrusters

where 2τ is the length of thruster firing. Suppose that δ>0 is the desired accuracy in adjusting the relative distance, τ is then chosen so that the distance traveled through the above sequence is less than δ. The automata enforcing these sequences is presented in Figure 3.14.

The automaton modeling the sequence controller is obtained by integrating the automata of Figure 3.13 and Figure 3.14 using the synchronous product operator. Let $FollowerCommands$ and $FollowerSequences$ denote the automata modeling the controller events of Figure 3.13 and controlling sequences of Figure 3.14, respectively. The $FollowerControllerDES$ can be obtained as

$$FollowerControllerDES = \mathbf{sync}(FollowerCommands, FollowerSequences) \tag{3.6}$$

As depicted in Figure 3.14, there is an event '*pulse*' between each '*on*' and 'off' command. This represents the duration in which the thruster pairs (T_1,T_4) or

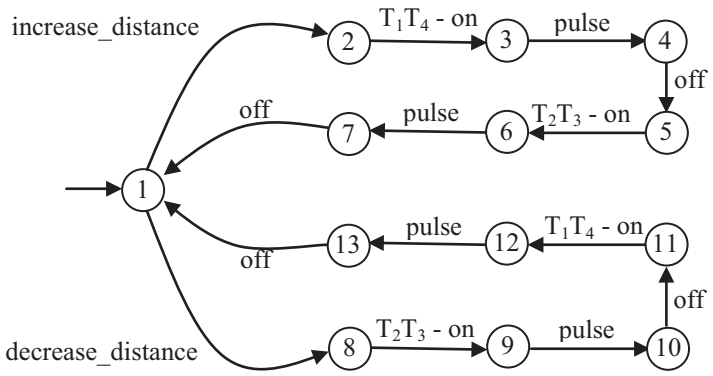


Figure 3.14: *FollowerSequences* - Follower Sequences

(T_2,T_3) are on.

3.3.4 Hybrid Modeling

The DES abstraction of the follower can be formed using the synchronous product of the automata modeling the components, the automata modeling their interactions and the sequence controller automata. Let the automaton

$$SystemDES_{abs} = (Q_{abs}, \Sigma_{abs}, T_{abs}, D_{abs}, \lambda_{abs}, q_{abs,0}) \tag{3.7}$$

be the DES abstraction of the leader-follower system. We have

$$SystemDES_{abs} = \mathbf{sync}(ComponentDES, InteractionsDES, SequenceController, FSCN) \tag{3.8}$$

where

$$ComponentDES = \mathbf{sync}(FT_1T_4, FT_2T_3, LT, RDS, RA, ACC) \tag{3.9}$$

and

$$InteractionsDES = \mathbf{sync}(INTFACC, INTLFA) \tag{3.10}$$

The hybrid automaton of the system can be written as

$$System = (Q, X, U, FT, Y, Init, S, \Sigma, T, D, \lambda, q_0) \tag{3.11}$$

where

$$\begin{gathered} X, U, FT, Y, Init \subset \mathbb{R}^2; \\ Q = Q_{abs}; \\ \Sigma = \Sigma_{abs}; \\ T = T_{abs}; \\ D = D_{abs}; \\ \lambda = \lambda_{abs}; \\ q_0 = q_{abs,0}; \end{gathered} \tag{3.12}$$

The system has a total of 6 components. Therefore, each discrete state of the system can be describes as a 6-tuple q=(q_1,...,q_6), where q_i represents the state of each component. The hybrid automaton model of the system has 5074 discrete states and 33306 transition

The faults in the system can be modeled by additive fault signals. We can write S_q, the dynamics in state q, as

$$S_q = \begin{cases} \dot{X}(t) = A_q X(t) + B_q U(t) + L_q F(t) \\ Y(t) = C_q X(t) \end{cases} \tag{3.13}$$

where

$$X(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \ , \ B_q = L_q \tag{3.14}$$

In the above equation, $U(t)$, $F(t)$ and L_q represent the matrices of applied forces to the follower, follower failures and the vector of the fault signatures in the discrete

state q, respectively. Also, $x(t)$ is the position of the follower and $\dot{x}(t)$ is the follower velocity. Therefore, the dynamics in state q can be rewritten as,

$$\begin{cases} \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} \\ \qquad + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{bmatrix} \\ y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \end{cases} \tag{3.15}$$

Note that the accelerometer bias failure is not included in these equations. We can simply show the accelerometer failure by adding the bias value to the output of the accelerometer.

In equation (3.15), the output $y(t)$ is the velocity of the follower while the input $U(t)$ is the applied force to it from the thrusters in which $u_i(t) = 1.2$ if thruster T_i is *on* and $u_i(t) = 0$ if thruster T_i is off. The values in matrix $F(t)$ are chosen based on the follower failure mode. For example, if the system enters the failure mode in

which all thrusters are stuck-off, equation (3.15) can be written as

$$\begin{cases} \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} \\ \qquad + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} -u_1(t) \\ -u_2(t) \\ -u_3(t) \\ -u_4(t) \end{bmatrix} \\ y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \end{cases} \tag{3.16}$$

Also, the dyanamics in state q when thruster T_1 is stuck-on will be as

$$\begin{cases} \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} \\ \qquad + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1.2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \end{cases} \tag{3.17}$$

In this equation, $f_1(t) = 1.2$ is the applied force to the follower from thruster T_1 which is stuck-on.

3.4 Fault Diagnoser

In this section, we design the residual generators (or isolators) for the system. Then, the hybrid fault diagnoser for the system will be constructed.

3.4.1 Residual Generator Design

There are three approaches for residual generators: ***parity space***, ***observer-based*** and ***parameter estimation***. In this work we apply the parity space approach. This approach is based on the '*parity check*', i.e. testing the parity equations that are based on the system equations and measured signals. If there is any inconsistency

in these parity equations, the occurrence of a failure can be estimated.

As mentioned in Section 3.2, the acceleration of the follower is measured by the accelerometer. We also calculate the ***Expected Acceleration*** from equation (3.4). The parity check is then applied by comparing the measured and expected accelerations. If there is an inconsistency between these two values, we can say that a failure has happened. The parity residual signal, P_D, can be written as

$$P_D = Expected\ Acceleration - Measured\ Acceleration \tag{3.18}$$

P_D is then used for ***Fault Detection***; as $P_D = 0$ if the system is in normal mode of operation and $P_D \neq 0$ if the system is in faulty mode of operation.

As mentioned in Section 3.2, there are three fault types in the system: accelerometer bias failure, thruster T_1 stuck-on and all follower thrusters stuck-off. A proper fault diagnosis includes ***Fault Isolation*** as well as fault detection. Therefore, after detecting the presence of a failure, we have to distinguish the active fault. Note that single failure scenario is assumed.

Fault isolation is performed by observing the parity signal P_I. The parity equation compares the ***Expected Relative Acceleration*** with the ***Actual Relative Acceleration***.

$$P_I = sgn(Expected\ Relative\ Acceleration) - sgn(Actual\ Relative\ Acceleration) \tag{3.19}$$

where $sgn(.)$ is the Sign Function. The actual relative acceleration is calculated by taking the second derivative of the relative distance sensor output as

$$Actual\ Relative\ Acceleration\ (t) = \frac{d^2r(t)}{dt^2} \tag{3.20}$$

In Equation (4.11), $sgn(Actual\ Relative\ Acceleration)$ is calculated by probing the

sign of the actual relative acceleration using Equation (3.21) below.

$$sgn(Actual\ Relative\ Acceleration) = \begin{cases} -1 & if Actual\ Relative\ Acceleration < 0, \\ 0 & if Actual\ Relative\ Acceleration = 0, \\ 1 & if Actual\ Relative\ Acceleration > 0. \end{cases} \tag{3.21}$$

The sign of expected relative acceleration is listed in Table 3.1 with respect to the status of leader and follower thrusters. If this sign is consistent with the sign of actual relative acceleration in Equation (3.21), we can say that this is the accelerometer that shows the wrong acceleration and therefore it is faulty. On the other hand, if there is an inconsistency between these values the failure is from thrusters.

Finally, isolation between thruster faults is done by probing the accelerometer sensor output. If all thrusters are stuck-off, the accelerometer will not pick any acceleration. The above comparison process (including P_D, P_I) is performed by the hybrid diagnoser. In the next section, we will discuss more about the fault detection and isolation procedure in the follower.

3.5 Simulation Results

We use the **MATLAB/SIMULINK** software to simulate the system behavior. ***Discrete Event Control Kit (DECK)*** software is also used for constructing the DES model of the system. DECK is a toolbox written in the programming language of MATLAB for the analysis and design of supervisory control systems based on discrete-event models [1]. In this section, we assume different maneuvers for the leader spacecraft and analyze the response of the follower in both normal and faulty modes of operation.

Figure 3.15 shows the position of leader and follower spacecraft in normal and

faulty modes of operation in normalized time units. The solid line shows the leader position on the x-axis at each time with the origin being its initial position and the dashed line represents the follower position. The decrease in this values mean that the spacecraft is accelerating to left, while an increase means it is accelerating to the right direction. We set the desired relative distance between the spacecraft to be 12 distance units.

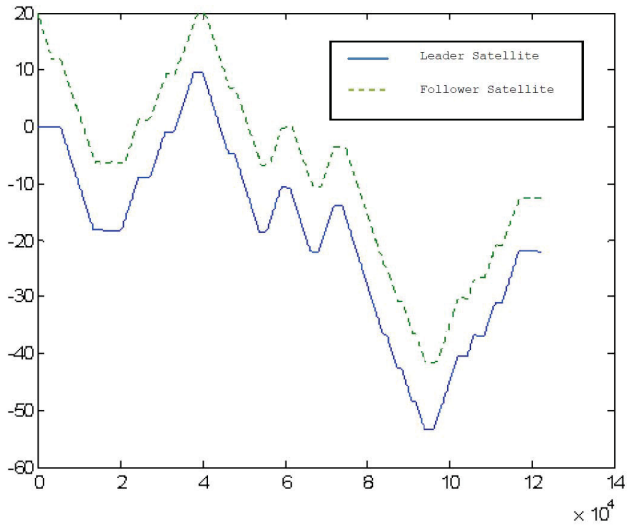
In Figure 3.15(a), at the beginning, the spacecraft are deployed 20 units apart from each other which is '*high*'. That makes the follower decrease its distance with the leader and accelerates to the left direction till the distance falls in the '*normal*' range. A similar scenario will happen if the relative distance becomes less that 10 units or '*low*'. Figure 3.15(b) shows the same maneuver for the leader except that at the time around $t = 65$ all the follower thrusters become '$stuck - off$'. This makes the follower continue moving in the same direction at a constant speed.

In the next section, different maneuvers are defined for the leader and the reaction of the follower and the values of fault detection and isolation signals will be explained in both normal and faulty conditions.

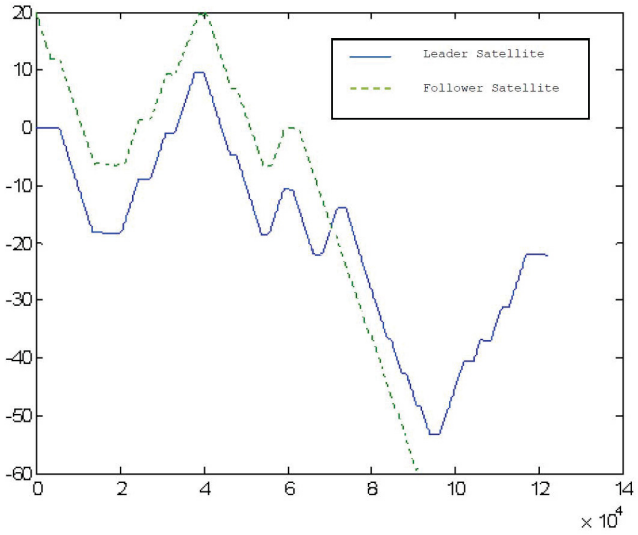
3.5.1 Normal Mode of Operation

In this scenario, the system operates in a fault-free mode. The follower follows the leader maneuver while keeping the relative distance at a desirable range; 12 units. Figure 3.16 demonstrates the spacecraft maneuver in normal mode of operation when the initial relative distance is '*high*'.

Figure 3.16 shows the position of the leader and follower spacecraft on a line with the origin being the initial position of the leader, fault detection signal P_D and fault isolation signal P_I, respectively. We mentioned in Section 3.4 that the P_D signal is used for detecting the presence of the faults. We also assumed that the ***Bang-Bang Control Policy*** is used to command the spacecraft to move. This



(a)



(b)

Figure 3.15: Position of Leader and Follower spacecraft in: (a) Normal Mode of Operation, (b) Faulty Mode of Operation

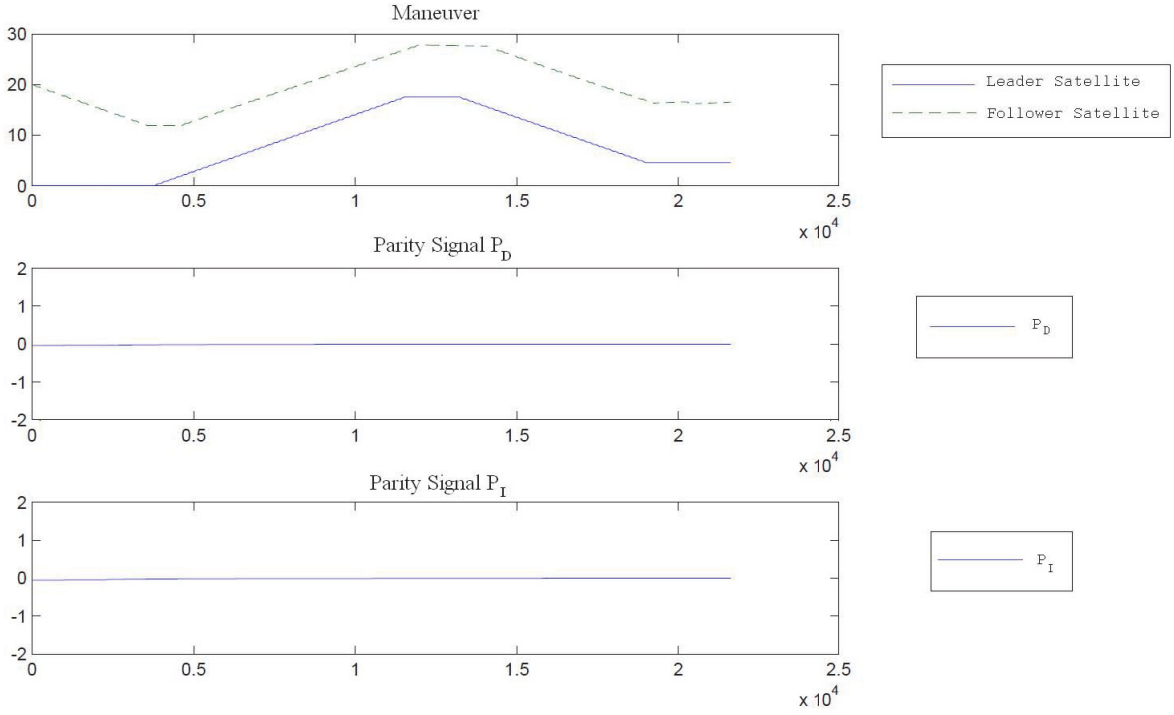


Figure 3.16: Normal Operation Mode

means that when follower receives the command to move towards the right, it first accelerates to the right and then to the left. The same can be said for going to the left direction; first it accelerates to the left and after that to the right. This makes the accelerometer pick up both positive and negative accelerations while moving in a single direction. It is observed that the accelerometer output signal has several sudden variations, as depicted in Figure 3.17.

The reader may note that if the '$decrease - distance$' command is activated during the bang-bang firings, the follower will not decrease its distance till the end of bang-bang firings. This may cause some delays in the response of the follower as shown in Figure 3.16.

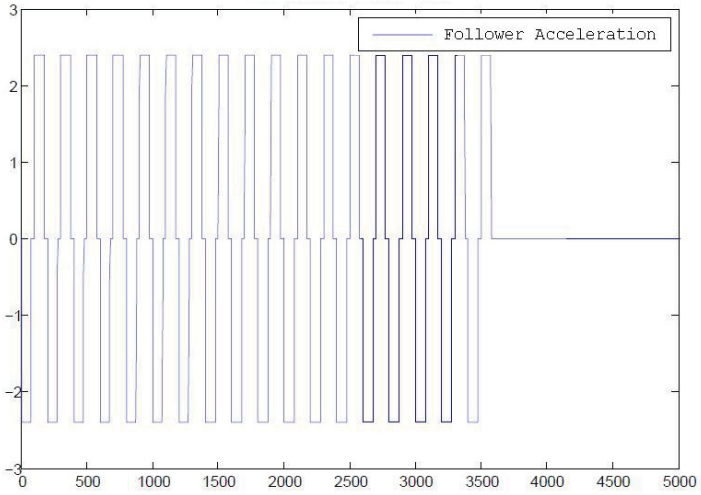
The parity signal P_D in Eq. (3.18) is the difference between the expected and measured accelerations. The expected acceleration is the total force applied to the follower as defined in Eq. (3.4) and can be 0, ± 1.2 or ± 2.4. The measured acceleration is , however, the output of accelerometer and is shown in Figure 3.17. From Eq. (3.18) and this figure, we expect the parity signal P_D to have similar sudden variations. Because of these fluctuations, it is reasonable to use the mean value of the signal for fault diagnosis. The same is true for P_I signal in fault isolation. The mean of P_D and P_I signals are shown in Figure 3.16. Based on these plots, one can say that the system evolves in the normal mode and no failure has happened.

3.5.2 Faulty Mode of Operation

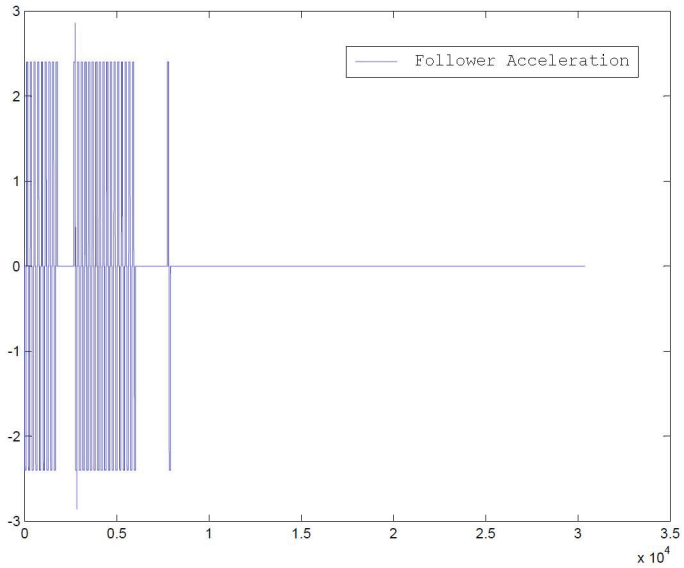
In this section we assume different failures for the follower and observe its behavior. The leader is assumed to be **fault-free**.

Accelerometer Bias Failure

At time $t = 3 \times 10^4$ the accelerometer fails and develops a bias of '+2' in its readings, i.e. showing more acceleration. The *maneuver*, P_D and P_I parity signals are shown



(a)



(b)

Figure 3.17: Follower Accelerometer Output for: (a) $0 \leq t \leq 5000$, (b) $0 \leq t \leq 35000$

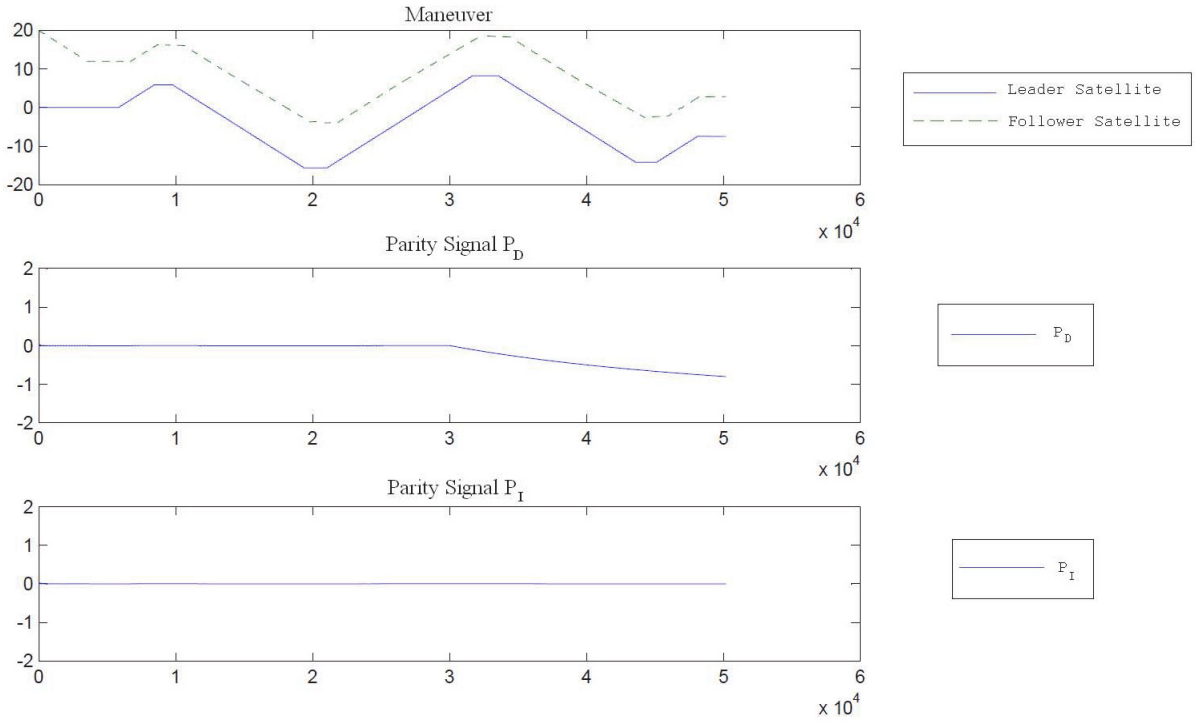


Figure 3.18: System Performance with Accelerometer's Positive Bias Failure

in Figure 3.18. As one can see from this figure, the follower continues its maneuver correctly even after the occurrence of failure. We can say that the accelerometer failure does no affect the maneuver of the follower. This was expected, because controlling the maneuver of the follower is only based on the relative distance.

On the other hand, the parity signal P_D decreases after accelerometer fails. As the signal is no longer *zero*, we can say that a failure has happened. Now that we have detected the occurrence of a failure, we have to isolate it, i.e. see whether it is from thrusters or accelerometer. We observe that the parity signal P_I, depicted in Figure 3.18, is approximately zero. Therefore, the failure is from accelerometer.

Figure 3.19 shows the spacecraft maneuvers and fault diagnosis signals for a '-2' bias in the accelerometer output.

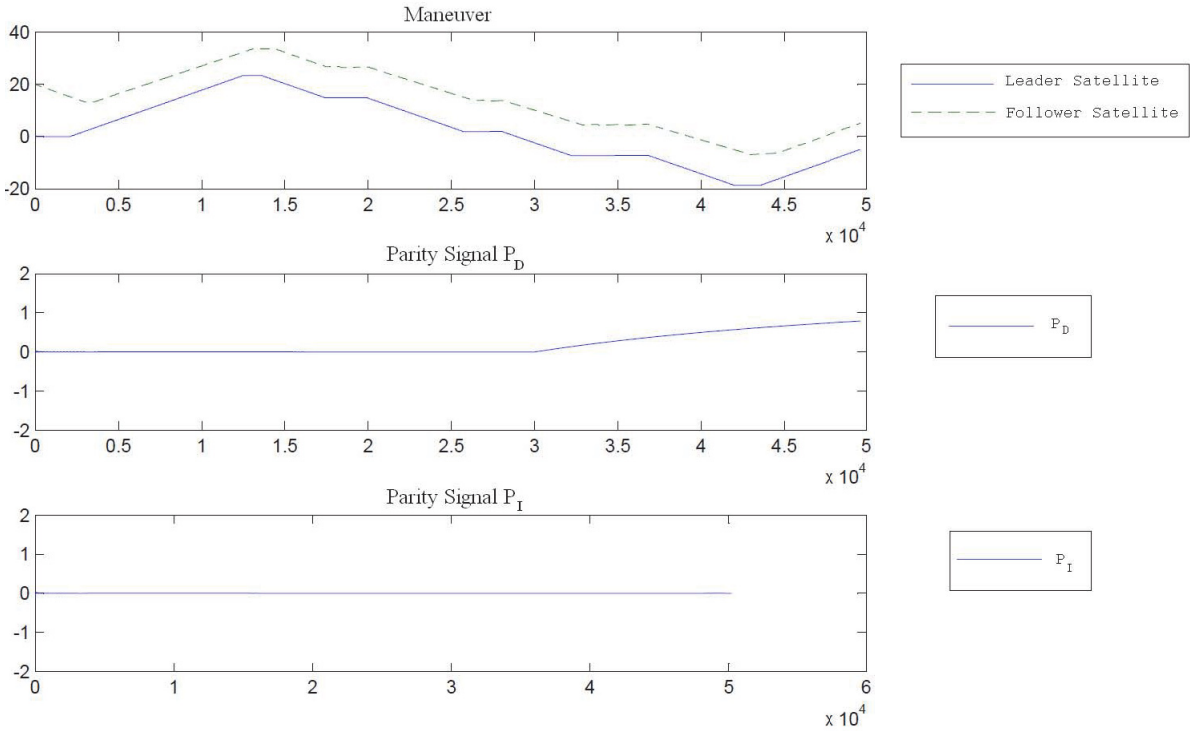


Figure 3.19: System Performance with Accelerometer's Negative Bias Failure

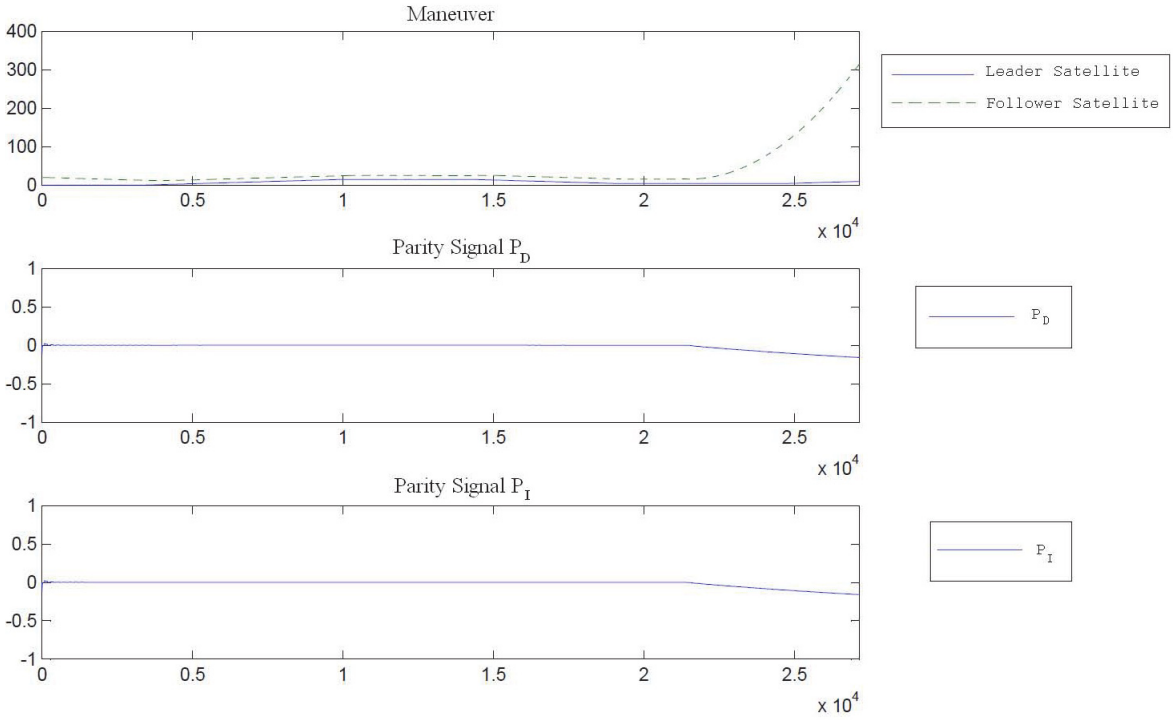


Figure 3.20: System Performance with Thruster T_1 becomes stuck-on

Thruster T_1 stuck-on

In this part, we consider the failure of thruster T_1. The spacecraft maneuver and parity signals are shown in Figure 3.20. At time $t = 22 \times 10^3$ thruster T_1 fails and follower continues moving right at a rapidly increasing pace. Note that the parity signals P_D and P_I become nonzero after failure happens.

All Thrusters stuck-off

Figure 3.21 illustrates the system performance when all thrusters become stuck-off before starting the operation. This may happen because of a common failure in the actuation system of the spacecraft.

As mentioned in section 3.3.1, thrusters may become stuck-off when they are either *on* or off. Therefore, the follower may stop or continue moving based on

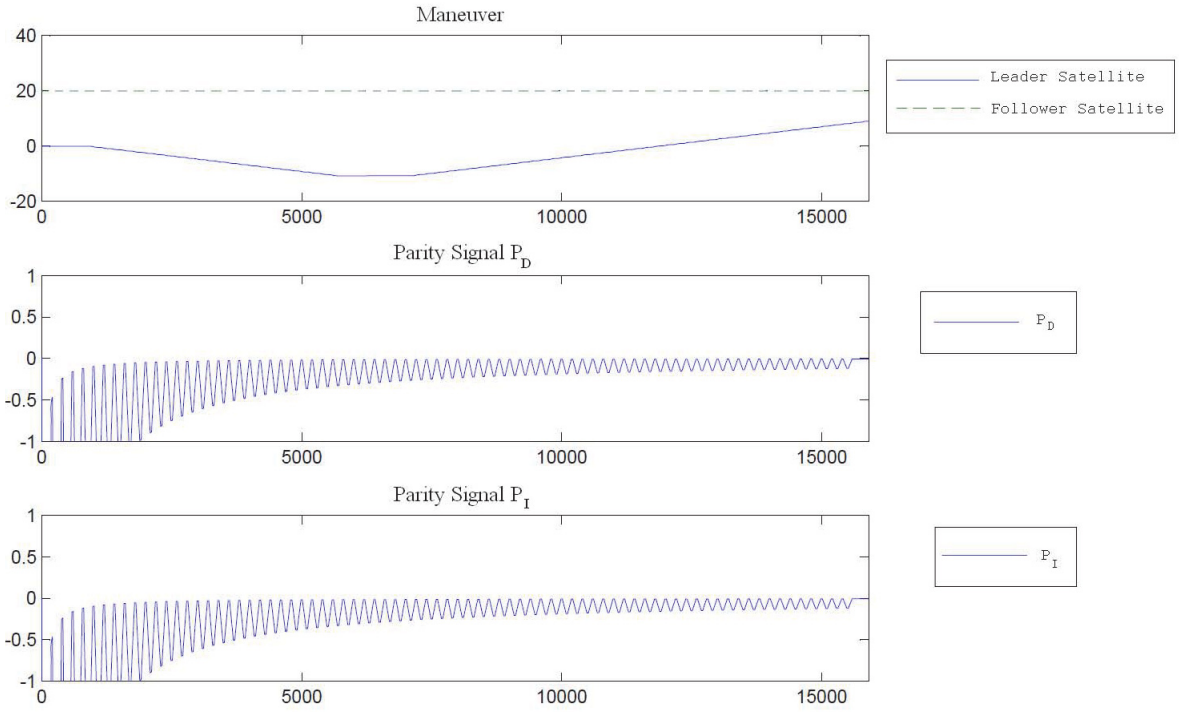


Figure 3.21: System Performance with all Thrusters become stuck-off

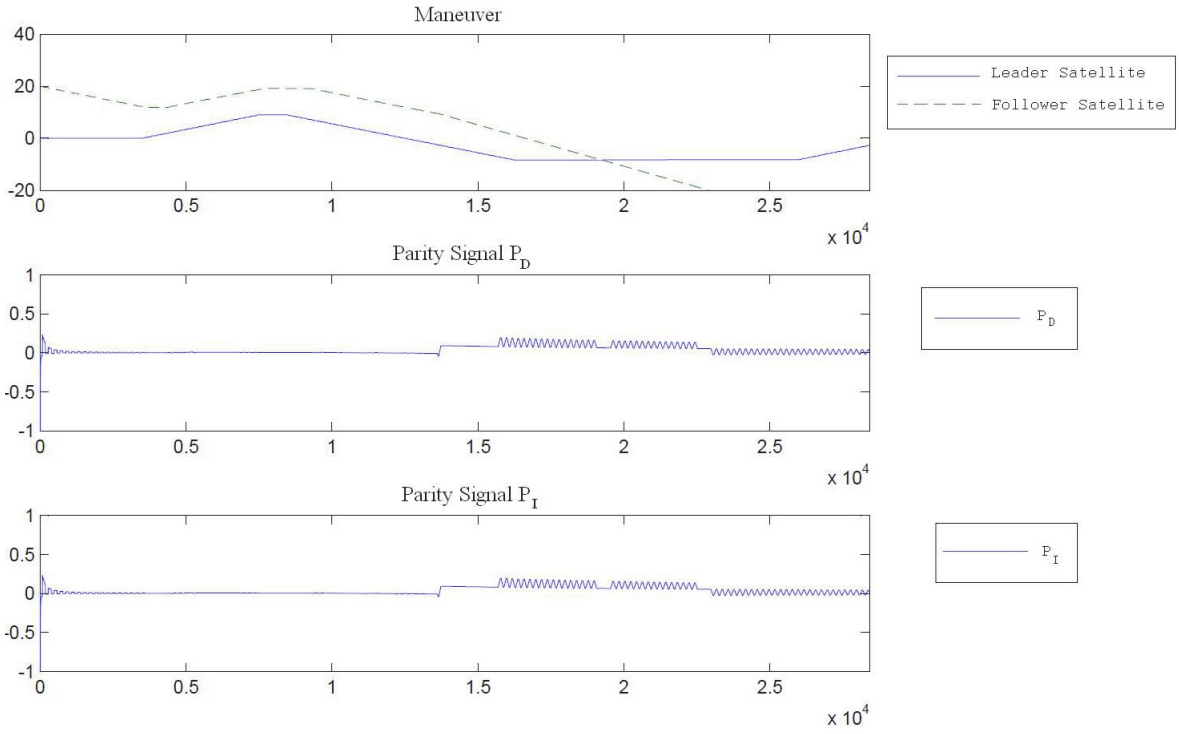


Figure 3.22: System Performance with all Thrusters become stuck-off

the last received command. In Figure 3.22, the failure happens when the follower was moving to the left direction. As we can see, it continues going left even when the relative distance becomes low. Moreover, Figure 3.23 demonstrates the system behavior in which the follower was going right before the failure happens.

Note that the parity signals, P_D and P_I, are both non-zero when the failure is active. This is similar to the case where only thruster T_1 was stuck-on. To differentiate these two faults, one may refer to the mean of the accelerometer output signal as shown in Figure 3.24.

Discarding the initial fluctuations due to the transients in sensor measurements in Figure 3.24(a), we can see that the mean of measured acceleration in the presence of thruster T_1 failure is nonzero. This signal is almost zero when all thrusters become stuck-off, as shown in Figure 3.24(b).

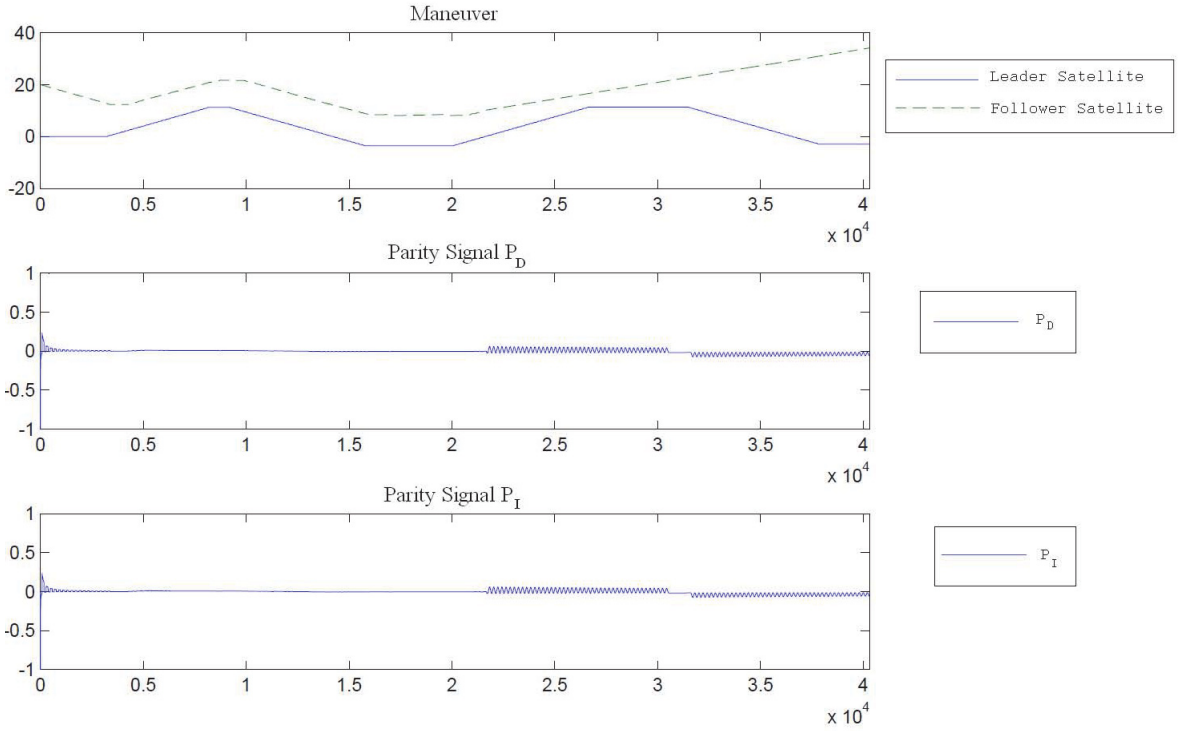
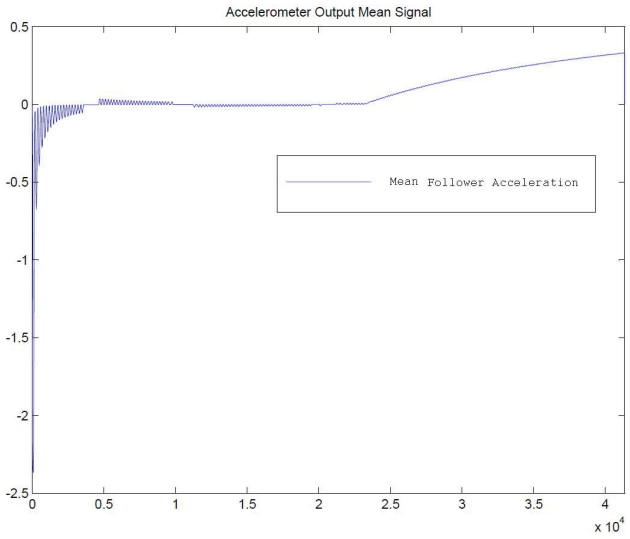
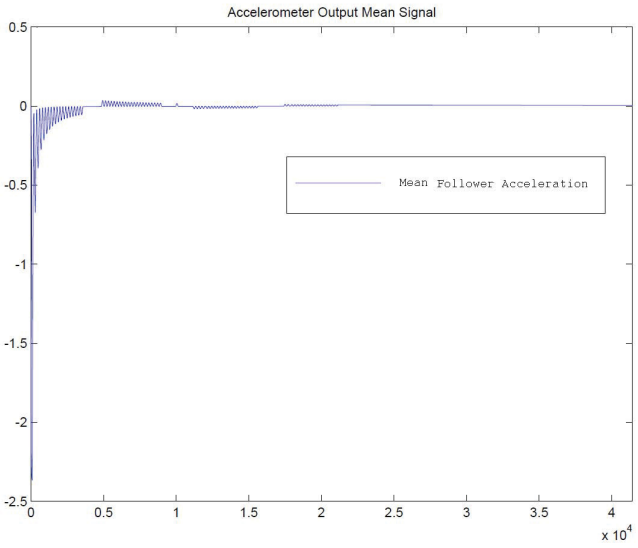


Figure 3.23: System Performance with all Thrusters become stuck-off



(a)



(b)

Figure 3.24: Accelerometer Output Mean Signal: (a) Thruster T_1 stuck-on, (b) All Thrusters stuck-off

3.5.3 Remarks

Remark 1 - The impact of discretization

In this chapter, the output of the relative distance and accelerometer sensors are discretized into 3 and 5 levels, respectively. These levels of discretization do not have any impacts on the accuracy of the diagnosis in this particular system. The reason is that we have used parity signals P_D and P_I for fault diagnosis and neither signal is affected by the discretizations.

Remark 2 - Spacecraft mass

In Eq. (3.2) we assumed that the mass of the spacecraft is constant and we normalized its value to 1. In the case of variable mass, the construction of parity signals would not change. Therefore, the diagnoser will perform similarly in this system. However, the simulations will be different from the constant mass as the continuous dynamics of the system are changed.

3.6 Summary

In this chapter, we investigated fault diagnosis in a system of Leader/Follower spacecraft in deep space. We described the system components and reviewed the dynamics of the system. We introduced different modes of operations and developed the state-space equations for one of the operating modes. Moreover, we developed a DES model of the system. Parity signals were also formulated from sensor output signal. Subsequently, the fault diagnoser was built based on the system model and parity signals. We considered three fault scenarios for the system and investigated the performance of diagnoser for each mode of operation. We also discussed how to isolate the faults by applying parity signals.

In this work, we studied failure diagnosability using simulation. A formal

test for failure diagnosability for hybrid diagnoser has been presented in [1]. The implementation of this test in the form of a software code would provide a more comprehensive assessment of our approach. This implementation is beyond the scope of this thesis.

In the next chapter, however, we will extend the problem presented in Chapter 3 to the case involving spacecraft rotation.

Chapter 4

Hybrid Fault Diagnosis in Leader-Follower with One-Dimensional Translational Motion and Rotation Around a Fixed Axis

In this chapter, we apply our hybrid fault diagnosis approach to a leader-follower pair spacecraft similar to the one in the previous chapter. In this case, the spacecraft rotate as well around an axis perpendicular to the translation axis. There are three possible faulty components in the follower; *Actuators*, *Accelerometer* and *Gyroscope*. Similar to the previous chapter, we describe the dynamics of the components by DES models. We will then develop a hybrid automaton model representing the behavior of the system and perform a hybrid diagnosis to detect and isolate the faults in the follower.

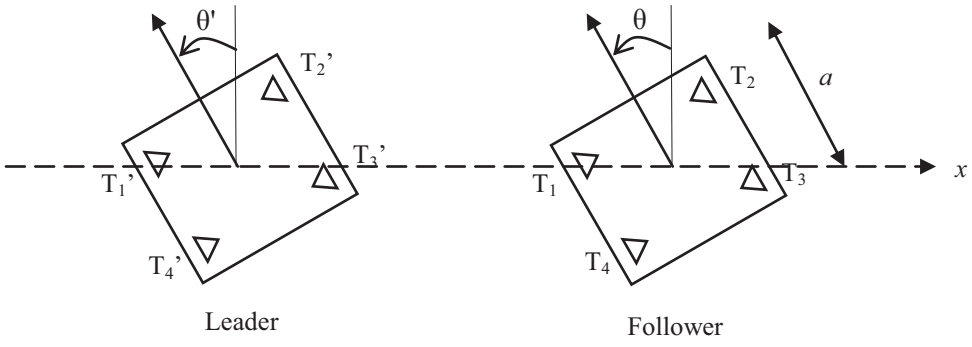


Figure 4.1: Leader-Follower Spacecraft

The rest of this chapter is as follows. In Section 4.1, we explain the mathematical equations of spacecraft and the relationship between the components. In Section 4.2, a hybrid model for our system is developed. Fault diagnosis based on the constructed hybrid model is then explained in Section 4.3. Finally, in Section 4.4, we study different scenarios for the system and use the diagnoser to detect and isolate possible faults.

4.1 Leader-Follower Formation Flying Spacecraft

In this chapter, we consider a group of two spacecraft in a Leader-Follower formation flying in deep space. In this problem, the spacecraft move along a fixed axis while rotating around an axis perpendicular to the translation axis as illustrated in Figure 4.1. In this formation flying, we would like to keep their relative distance at a desired range and their angular orientation similar to each other. Each spacecraft has four **Thrusters,** one **Distance Sensor**, one **Accelerometer** and also one **Gyroscope**. Similar to the previous model, we assume that all the elements in the leader are fault free. In the follower, however, elements may fail according to the list below. Note that the faults are considered to be *permanent*.

- **Thrusters**, thruster 'T_1' may become stuck-on or all thrusters may become stuck-off at the same time.
- **Accelerometer**, measuring the acceleration with a positive or negative bias.
- **Gyroscope**, measures the angular velocity of the spacecraft and it may fail and its data become unavailable.
- **Distance Sensor**, this sensor may fail. However, we assume that the health of this sensor is monitored using hardware redundancy and from the diagnoser's perspective, this sensor is fault-free.

The gyroscope measures the angular velocity. Using this angular velocity, we calculate angular acceleration and also angular position denoted by θ in Figure 4.1. The follower supervisor observes the distance sensor output and the angular position and issues the necessary commands based on these values to maintain the required distance and angular position. A general procedure in the system is as follows.

Assuming that the spacecraft are initially deployed far from each other, the follower receives the command to decrease the distance and moves towards the leader. Before decreasing the distance, the follower should rotate to the vertical position (θ=0). If the gyroscope shows a nonzero value, a '*turn clockwise*' or '*turn counterclockwise*' command will be issued for the follower. It rotates by firing the opposite side thrusters until θ becomes zero. Then, based on the relative distance between the spacecraft, the appropriate commands will be issued as discussed in the previous chapter. Finally, follower rotates so that its angular position becomes similar to the follower's. Note that in the meantime, the leader may change its position.

The dynamics of the system is similar to the previous chapter. The laws of motion, however, should be modified to incorporate the angular rotation as shown

in the following equations.

$$\dot{x} = v \tag{4.1}$$

$$M\dot{v} = F \cdot cos\,(\theta) \tag{4.2}$$

where x is the absolute position of the follower, v is the velocity, M is the mass of the spacecraft and F is the net force applied to it. Without loss of generality, we assume that the spacecraft mass equals 1. The gravity is also negligible as the system operates in deep space. Hence, we have

$$\ddot{x} = F \cdot cos\,(\theta) = (u_1 - u_2 - u_3 + u_4) \cdot cos\,(\theta) \tag{4.3}$$

where 'u_i' refers to the applied force to the spacecraft resulting from thruster 'T_i' as shown in Figure 4.2. For rotation we have

$$\omega = \dot{\theta} \tag{4.4}$$

$$I\dot{\omega} = (-F_1 + F_2 - F_3 + F_4) \cdot \frac{a}{2} \tag{4.5}$$

where I is the moment of inertia and a is the length of spacecraft side. For simplicity, we assume that

$$\frac{a}{2I} = 1. \tag{4.6}$$

4.2 System Model

We begin this section by modeling individual components, controller and interactions among the components. We will then use MATLAB/SIMULINK Software along with DECK Toolbox [1] to build the final model.

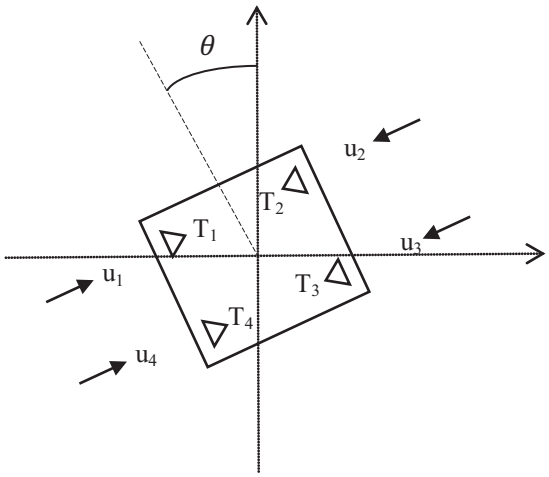


Figure 4.2: The Follower Spacecraft

4.2.1 DES Model of System Components

As discussed before, the considered leader-follower formation has discrete characteristics. We use discrete-event models to show the discrete behavior of the system. The DES model of the system components are described in the following.

Follower Thrusters

The follower has two types of movement, translation in one axis and rotation around its center of gravity. To model the thrusters, we group them based on the movement of the spacecraft as follows.

- **Translation**, (T_1,T_4) for accelerating to the right direction and (T_2,T_3) for accelerating to the left direction.
- **Rotation**, (T_1,T_3) for rotating clockwise (cw) and (T_2,T_4) for rotating counterclockwise (ccw).

Figure 4.3 illustrates the DES model of these thrusters. The configuration of the events and faults are similar to the system in the previous chapter.

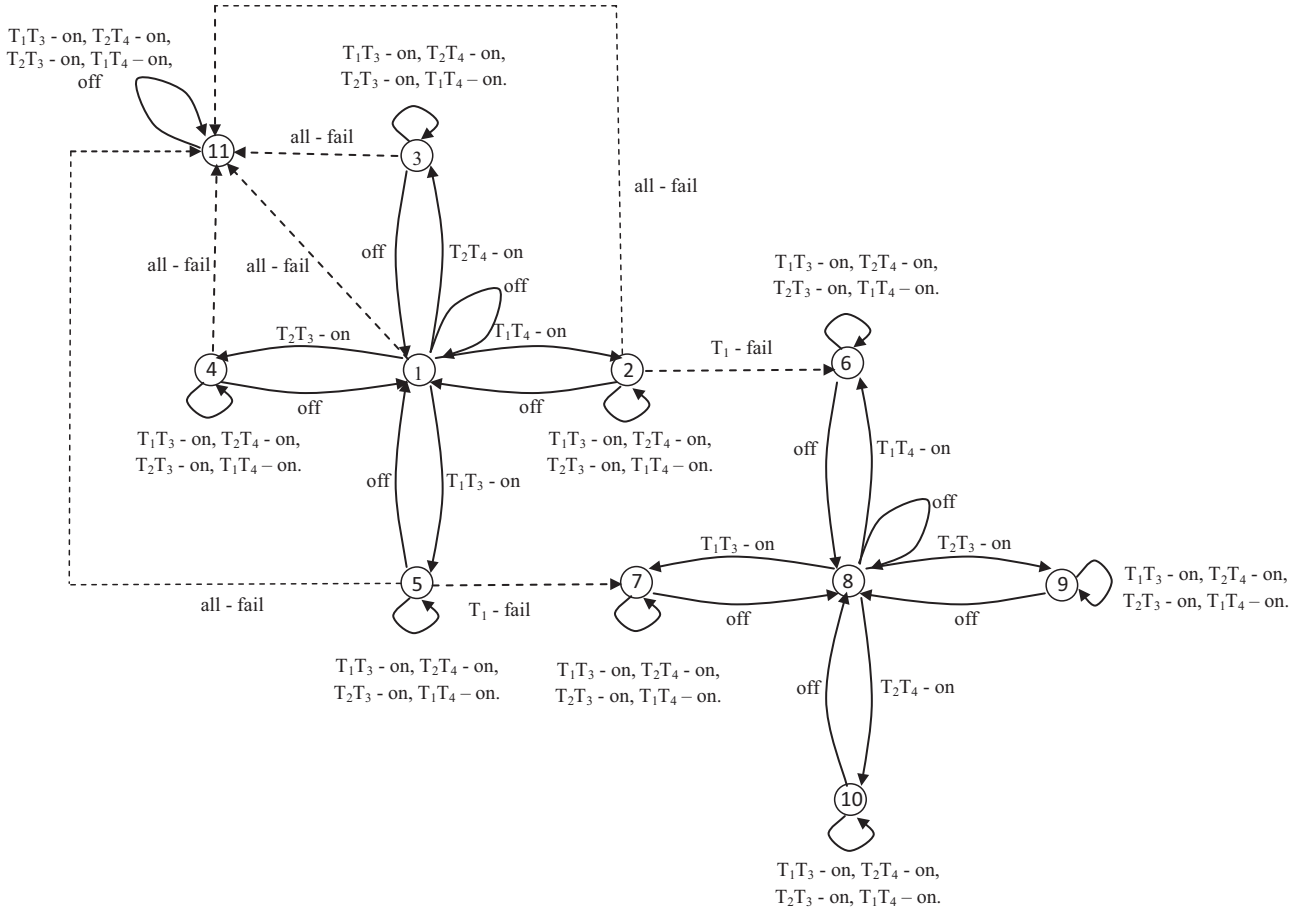


Figure 4.3: FT - Follower Thrusters

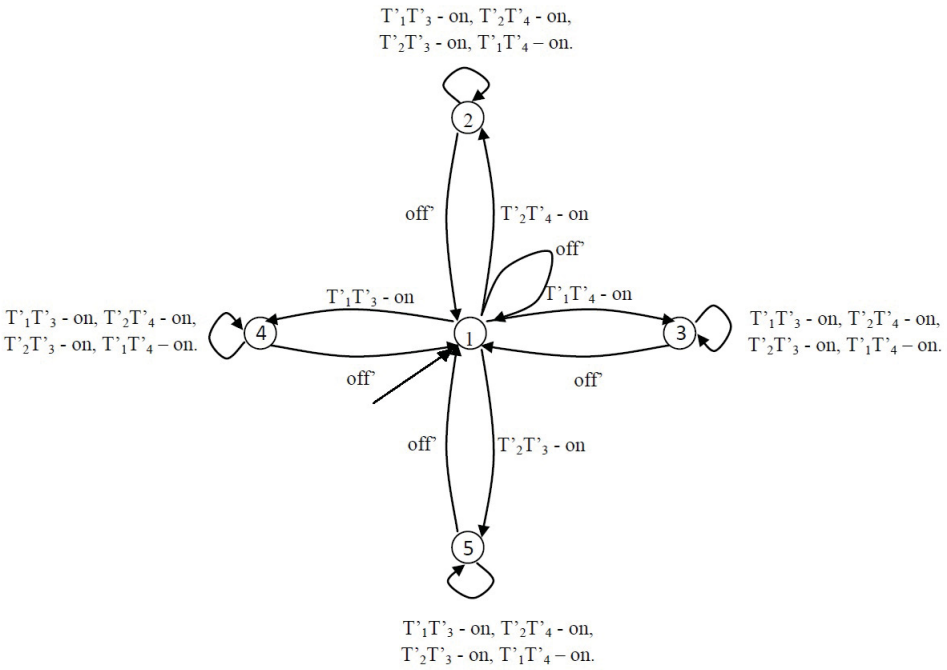


Figure 4.4: LT - Leader Thrusters

The fault-free DES model of leader thrusters is also pictured in Figure 4.4.

Relative Distance Sensor

Similar to the previous chapter, a laser sensor measures the relative distance between the leader and follower and is assumed to be fault-free. The output of this sensor is divided into three discrete levels, **high**, **normal** and **low**. A DES model representing this sensor is shown in Figure 4.5. Taking the second derivative of this sensor's output, we calculate the actual relative acceleration and build a DES model for that as illustrated in Figure 4.6. The relative acceleration can be **zero**, **positive** or **negative**.

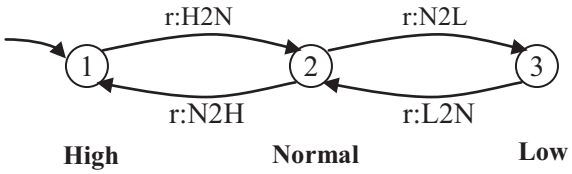


Figure 4.5: *RDS* - Relative Distance Sensor

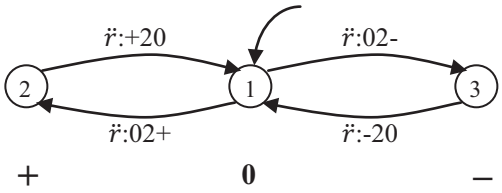


Figure 4.6: *RA* - Relative Acceleration

Accelerometer

The accelerometer measures the acceleration of the follower spacecraft. It may fail and have a bias in its output, e.g. showing a smaller acceleration than the actual value. Similar to the previous chapter, this value is divided into five discrete levels: **negative**, **very negative**, **positive**, **very positive** and **zero**. As an example, it shows *positive* when the spacecraft is moving to the right direction but only thruster T_1 is firing. In the case of rotation, firing of pairs (T_1,T_3) and (T_2,T_4), accelerometer shows a *zero* acceleration. Therefore, the DES model of accelerometer is similar to the model of the previous chapter as shown in Figure 4.7. Note that the fault in the accelerometer means that there is a negative *bias* in the measured value, showing a smaller acceleration than the actual value. For example, it may show *zero* instead of *positive*.

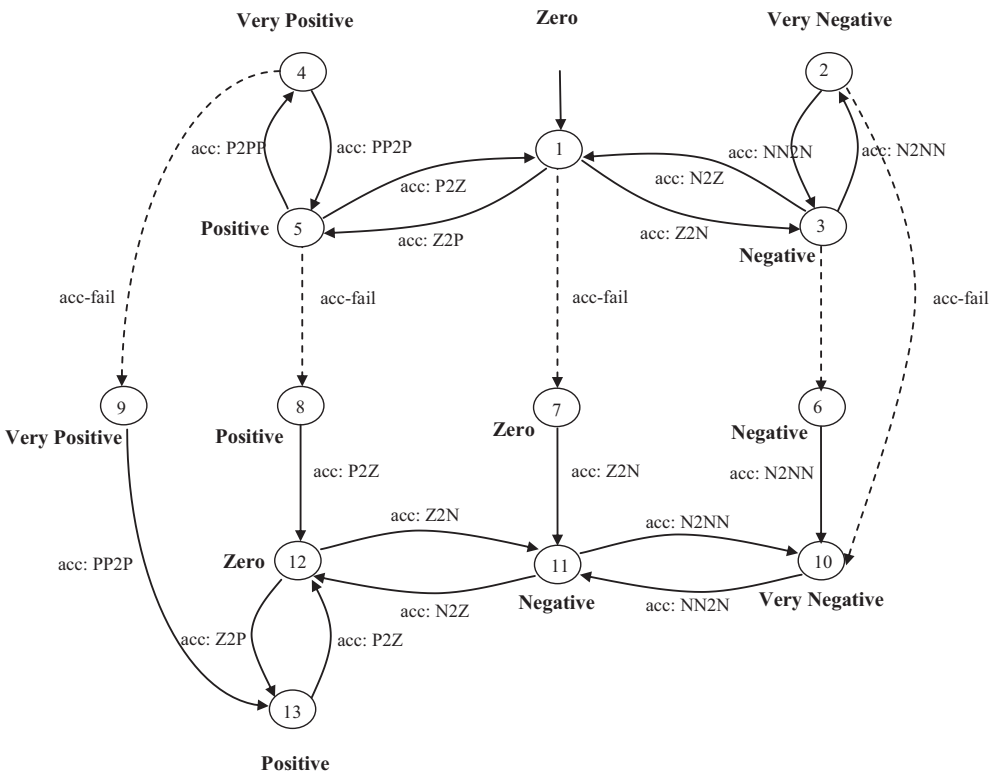


Figure 4.7: *ACC* - Accelerometer

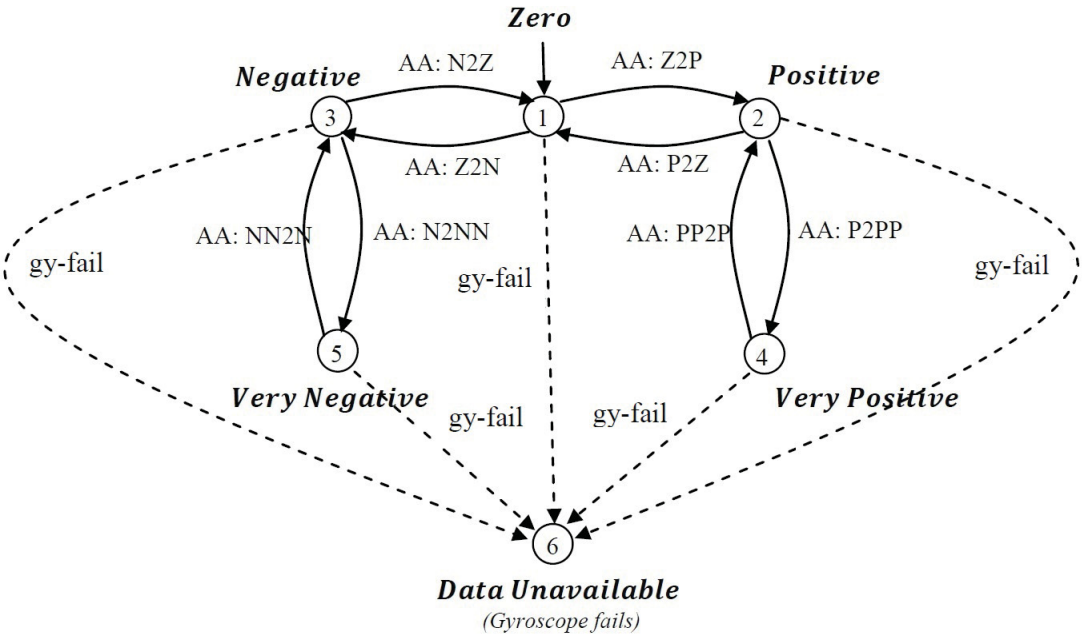


Figure 4.8: *ANGACC* - Discretized Angular Acceleration

Gyroscope

The gyroscope measures the angular velocity in rotation. We use this value and calculate the **angular acceleration** and **angular position** by taking the derivate and integral of angular velocity.

The angular acceleration value is discretized into five discrete levels: **negative**, **very negative**, **positive**, **very positive** and **zero**. For example, it shows *positive* when the spacecraft is rotating counterclockwise and thruster T_1 is stuck-on. If T_1 is normal, the gyroscope shows *very negative* when (T_1,T_3) fires. Figure 4.8 shows the DES model for this discretized value. We assume that the gyroscope may fail and its data becomes unavailable.

The angular position is the angle between y-axis and the body axis of the spacecraft which is perpendicular to the thrusters as shown in Figure 4.2. This value is divided into five discrete levels, *zero*, P_1, P_2, P_3 and P_4 as illustrated in

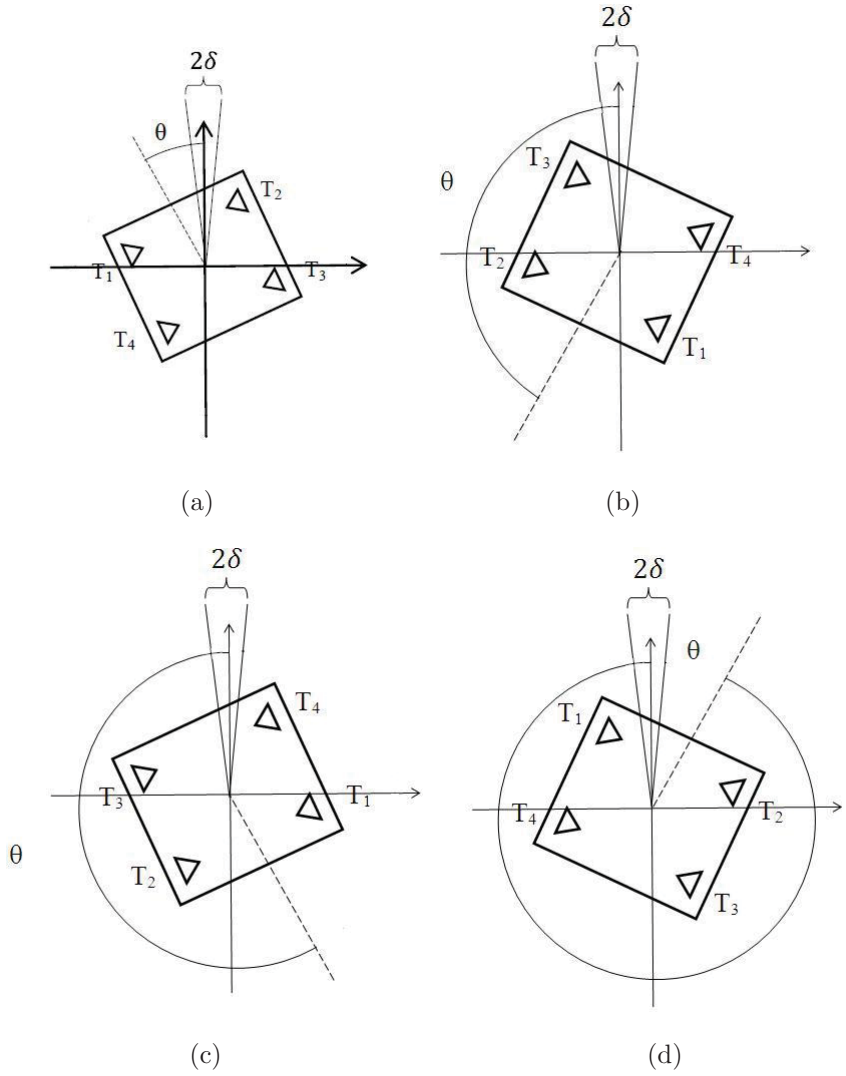


Figure 4.9: Discrete Levels of Angular Position with $0 : -\delta \leq \theta < \delta$ (a) $P_1 : \delta \leq \theta < 90$, (b) $P_2 : 90 \leq \theta < 180$,(c) $P_3 : 180 \leq \theta < 270$, (d) $P_4 : 270 \leq \theta < 360 - \delta$

Figure 4.9 where δ is a small positive number. The spacecraft is in vertical position when the output is *zero*.

The discrete event model for the angular position is depicted in Figure 4.10. We can see that the discrete values become unavailable when the gyroscope fails. There are five events '$\theta : 0$', '$\theta : P_1$', '$\theta : P_2$', '$\theta : P_3$' and '$\theta : P_4$' representing changes in the angular position. The event '$gy - fail$' occurs when the gyroscope fails and becomes unavailable. The outputs $P_j{}'$ are similar to P_j but gyroscope data is unavailable.

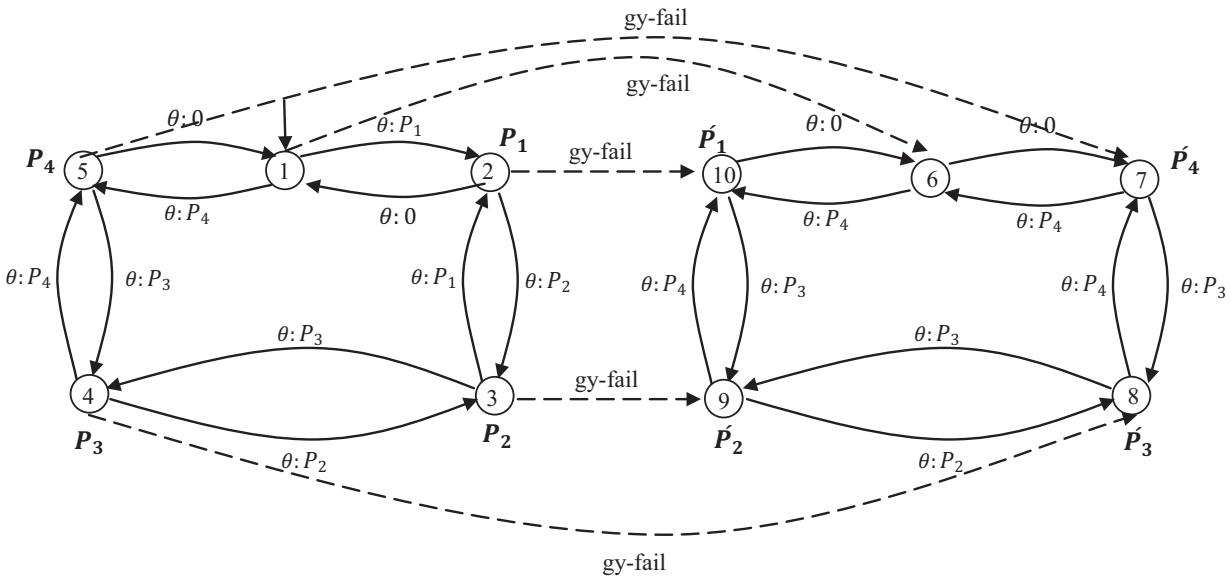


Figure 4.10: $TETA$ - Angular Position

Fault Scenario

Single failure scenario with permanent faults is assumed, i.e. only one fault at a time is present and that the faults are permanent. This assumption is enforced by the DES model in Figure 4.11.

4.2.2 Interactions Among the Components

We expect from our system that the change in the status of one of the components affect the other components' behavior due to the interactions among them. These interactions can be modeled using discrete event models and are:

- Accelerometer reading as a function of follower thruster and angular position.
- Angular acceleration as a function of follower thrusters.
- Relative acceleration as a function of leader thrusters, follower thrusters and their angular positions.

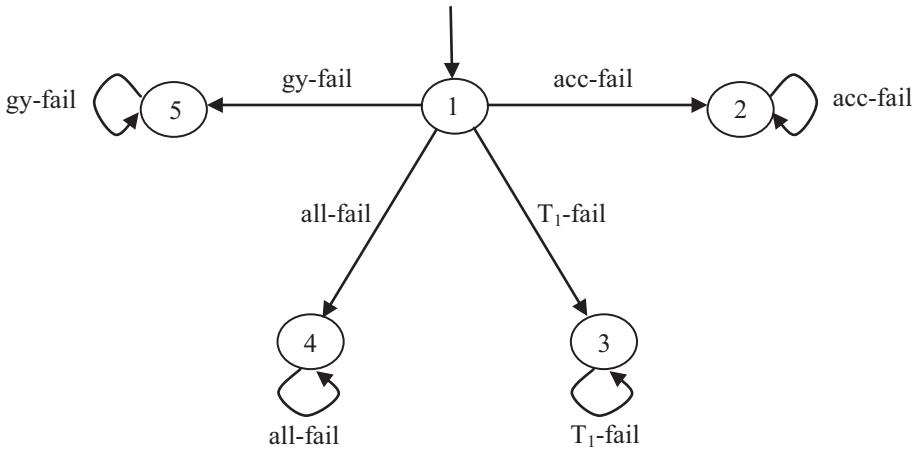


Figure 4.11: $FSCN$ - Single Failure Scenario

Accelerometer reading as a function of follower thruster and angular position

In this part we model the relationship between the follower thrusters and the acceleration. The output of the accelerometer at each state depends upon the angular position of the follower. For example, when the thrusters pair (T_1, T_4) is active and the spacecraft is moving to the right direction, the accelerometer will show '*very positive*' if $\theta \in P_1$ while it will show '*very negative*' if $\theta \in P_2$.

The output of accelerometer in both healthy and faulty conditions is shown in Table 4.1 with respect to the follower thrusters and angular position. Note that thrusters and accelerometer failures are not active at the same time due to the single-failure scenario.

Angular acceleration as a function of follower thrusters

The gyroscope output depends on the number of active thrusters and their health status. We will explain in Section 4.3 that for fault diagnosis the value of angular

Table 4.1: Interactions among Follower Thrusters and Accelerometer

Follower State	**Angular Position**	**Accelerometer Output**	
		Healthy	**Faulty**
1	P_1 or $P_1{}'$	Zero	Negative
	P_2 or $P_2{}'$	Zero	Negative
	P_3 or $P_3{}'$	Zero	Negative
	P_4 or $P_4{}'$	Zero	Negative
2	P_1 or $P_1{}'$	Very Positive	Positive
	P_2 or $P_2{}'$	Very Negative	Very Negative
	P_3 or $P_3{}'$	Very Negative	Very Negative
	P_4 or $P_4{}'$	Very Positive	Positive
3	P_1 or $P_1{}'$	Zero	Negative
	P_2 or $P_2{}'$	Zero	Negative
	P_3 or $P_3{}'$	Zero	Negative
	P_4 or $P_4{}'$	Zero	Negative
4	P_1 or $P_1{}'$	Very Negative	Very Negative
	P_2 or $P_2{}'$	Very Positive	Positive
	P_3 or $P_3{}'$	Very Positive	Positive
	P_4 or $P_4{}'$	Very Negative	Very Negative
5	P_1 or $P_1{}'$	Zero	Negative
	P_2 or $P_2{}'$	Zero	Negative
	P_3 or $P_3{}'$	Zero	Negative
	P_4 or $P_4{}'$	Zero	Negative
6	P_1 or $P_1{}'$	Very Positive	——
	P_2 or $P_2{}'$	Very Negative	——
	P_3 or $P_3{}'$	Very Negative	——
	P_4 or $P_4{}'$	Very Positive	——
7	P_1 or $P_1{}'$	Zero	——
	P_2 or $P_2{}'$	Zero	——
	P_3 or $P_3{}'$	Zero	——
	P_4 or $P_4{}'$	Zero	——
8	P_1 or $P_1{}'$	Positive	——
	P_2 or $P_2{}'$	Negative	——
	P_3 or $P_3{}'$	Negative	——
	P_4 or $P_4{}'$	Positive	——
9	P_1 or $P_1{}'$	Negative	——
	P_2 or $P_2{}'$	Positive	——
	P_3 or $P_3{}'$	Positive	——
	P_4 or $P_4{}'$	Negative	——
10	P_1 or $P_1{}'$	Positive	——
	P_2 or $P_2{}'$	Negative	——
	P_3 or $P_3{}'$	Negative	——
	P_4 or $P_4{}'$	Positive	——
11	P_1 or $P_1{}'$	Zero	——
	P_2 or $P_2{}'$	Zero	——
	P_3 or $P_3{}'$	Zero	——
	P_4 or $P_4{}'$	Zero	——

Table 4.2: Mapping in Normal Condition

(Leader Angular Position, Follower Angular Position)	**Relative Acceleration**
(P_1,P_1)	zero, negative, positive
(P_3,P_1)	negative
(P_1,P_4)	positive

acceleration is required. Therefore, in this part we model the interactions between the follower thrusters and the angular acceleration which is the derivative of the gyroscope output. This interaction model is illustrated in Figure 4.12. Deriving the model is similar to the previous chapter and will not be discussed here.

Relative acceleration as a function of leader thrusters, follower thrusters and their angular positions

Next, we discuss the interactions among leader/follower spacecraft, their relative acceleration calculated using the relative distance sensor output and angular position calculated from the gyroscope output.

Each state of this model has three outputs: angular position of leader, angular position of follower and the relative acceleration. The relative acceleration at each state is a function of angular position at that state as shown in equation 4.7 below.

$$\ddot{r} = F_{Follower} \cdot cos\left(\theta\right) - F_{Leader} \cdot cos\left(\theta^{'}\right) \tag{4.7}$$

For simplicity, we build a mapping from angular positions to the relative acceleration to show this relationship. Table 4.2 list some of the values from this mapping for a normal condition when both spacecraft are firing (T_1, T_4) and ($T_1{'}$, $T_4{'}$) thrusters. Table 4.3 shows these values in the faulty condition when thruster T_1 of follower is stuck-on and the leader is firing (T_2, T_3).

Note that $\ddot{r}$ can have different values at each state in these tables because the tables show relative acceleration as a function of discretized values of θ and $\theta^{'}$. For

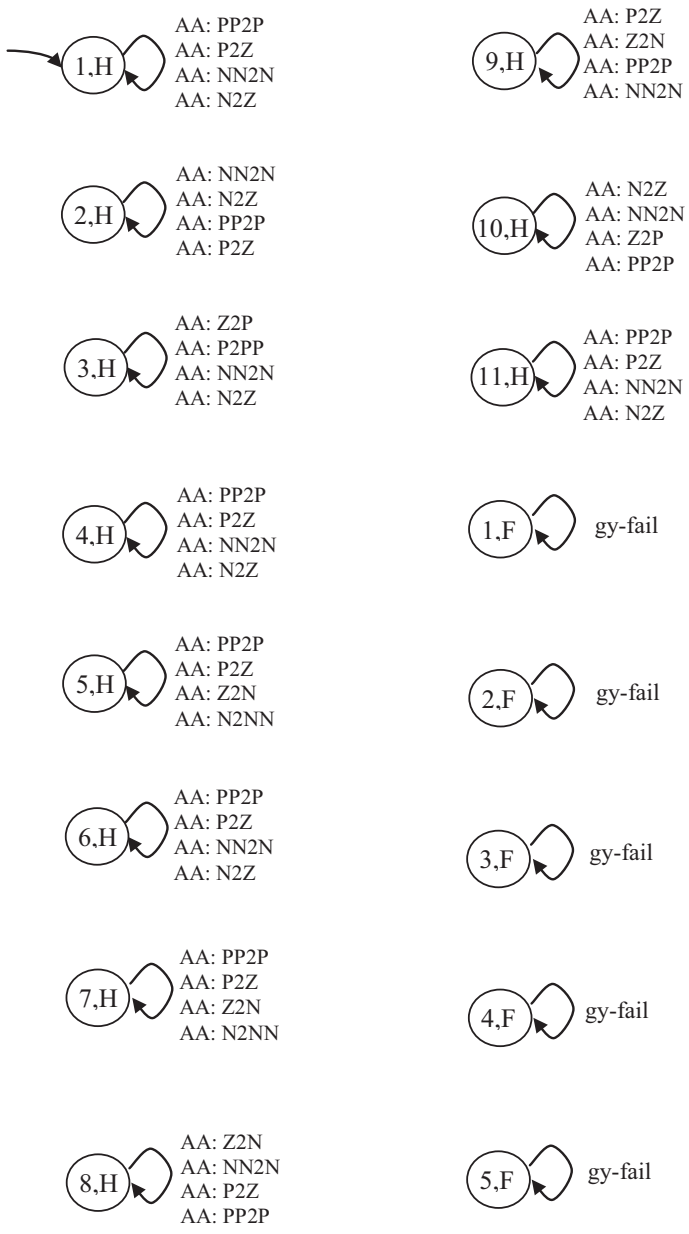


Figure 4.12: $INTFGY$: Interactions among Follower Thrusters and Angular Acceleration

Table 4.3: Mapping in Faulty Condition

(Leader Angular Position, Follower Angular Position)	**Relative Acceleration**
(P_1,P_1)	positive
(P_3,P_1)	negative
(P_1,P_4)	positive

more precision, one can divide the angular position into more discrete levels. However, this leads to a more complex model and requires more information exchange between the components and this is not desirable.

4.2.3 Supervisory Control

The supervisory controller generates sequence of commands for the follower in order to maintain its relative distance and angular position. The sequence involves three main steps:

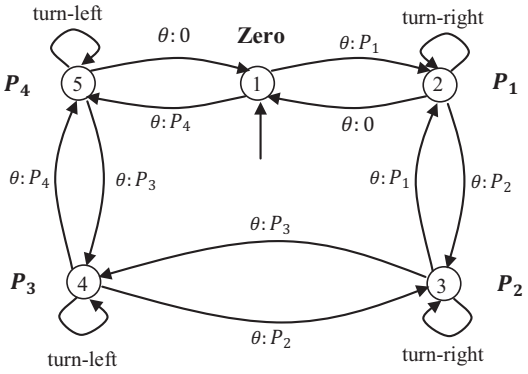
1. The follower should rotate so that it goes to the vertical position, i.e. $\theta \cong 0$.

2. The relative distance is maintained by firing appropriate thrusters.

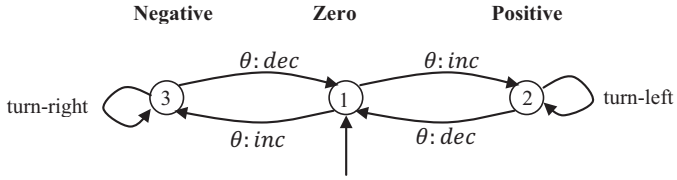
3. Based on the leader's angular position, P_i, the appropriate rotation command is sent to the follower. The follower rotates until its angular position becomes similar to the leader's angular position.

The FSAs for these three steps are shown in Figure 4.13.

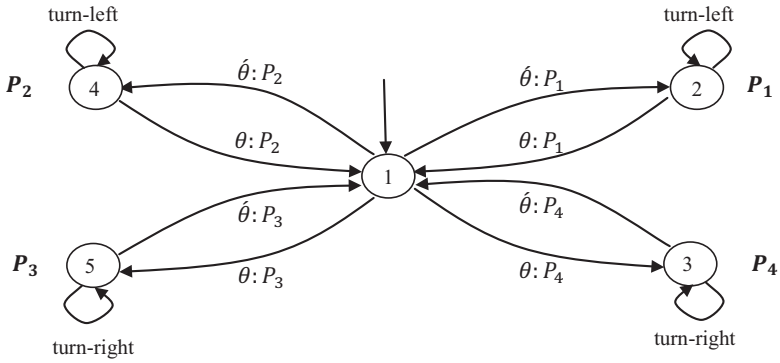
The sequence of commands are similar to the previous chapter and are shown in Figure 4.14. As mentioned in the previous chapter, the event '*pulse*' represents the duration in which the thruster pairs are on. Calculating the synchronous product of automata in Figure 4.13 and Figure 4.14, one can obtain the sequence controller of this system.



(a)

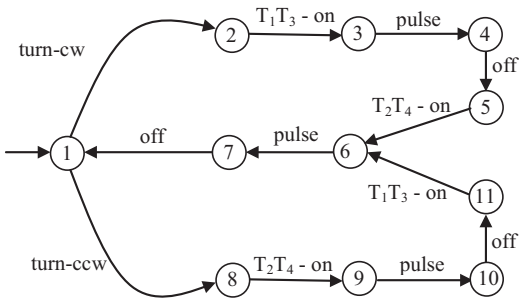


(b)

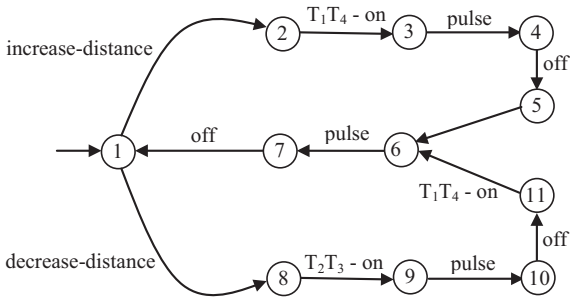


(c)

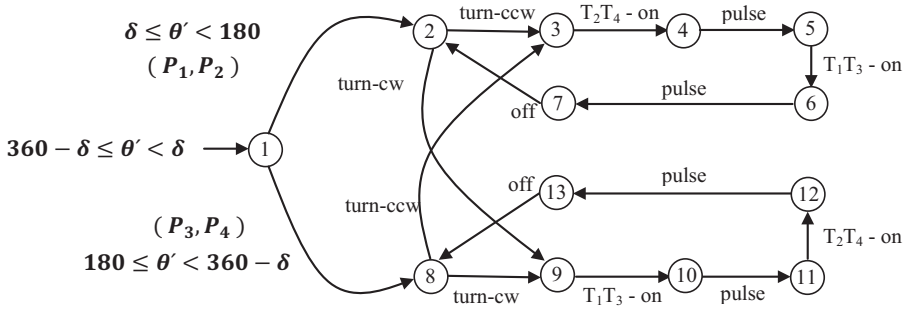
Figure 4.13: *FollowerCommands* - Follower Commands for: (a) Step 1, (b) Step 2,(c) Step 3



(a)



(b)



(c)

Figure 4.14: *FollowerSequences* - Follower Sequences for: (a) Step 1, (b) Step 2,(c) Step 3

4.2.4 Hybrid Model

The DES of this system can be modeled using the synchronous product of the automata modeling the components (Section 4.2.1), their interactions (Section 4.2.2) and the sequence controller (Section 4.2.3).

The hybrid model of the system is similar to the hybrid model in the previous chapter for a formation flying without rotation. We can then write the dynamics of the new system in state q as,

$$S_q : \begin{cases} \dot{X}_1(t) = \begin{bmatrix} 0 & 1 \\ & \\ 0 & 0 \end{bmatrix} X_1(t) + \begin{bmatrix} 0 & 0 & 0 & 0 \\ & & & \\ 1 & -1 & -1 & 1 \end{bmatrix} U(t)\ cos\,(\theta) + \begin{bmatrix} 0 & 0 & 0 & 0 \\ & & & \\ 1 & -1 & -1 & 1 \end{bmatrix} F(t)\ cos\,(\theta) \\ \dot{X}_2(t) = \begin{bmatrix} 0 & 1 \\ & \\ 0 & 0 \end{bmatrix} X_2(t) + \begin{bmatrix} 0 & 0 & 0 & 0 \\ & & & \\ -1 & 1 & -1 & 1 \end{bmatrix} U(t) + \begin{bmatrix} 0 & 0 & 0 & 0 \\ & & & \\ -1 & 1 & -1 & 1 \end{bmatrix} F(t) \\ Y(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1(t) \\ \\ X_2(t) \end{bmatrix} \end{cases} \tag{4.8}$$

where

$$X_1(t) = \begin{bmatrix} x \\ \\ v \end{bmatrix},\ X_2(t) = \begin{bmatrix} \theta \\ \\ \omega \end{bmatrix}. \tag{4.9}$$

4.3 Fault Diagnoser

The residual generators for the system are designed in this section. We will then construct the hybrid fault diagnoser for the system. For the purpose of fault diagnosis, the follower diagnoser receives information about the leader thruster commands and discretized values of leader's angular position.

4.3.1 Residual Generator Design

The method of designing the residual generators are similar to the previous chapter and we will not discuss it here. For ***Fault Detection***, the parity signal P_D is generated as expressed in Equation 4.10. $P_D = 0$ if the system is in normal mode of operation and $P_D \neq 0$ if the system is in faulty mode of operation.

$$P_D = ExpectedAcceleration - MeasuredAcceleration \tag{4.10}$$

For ***Fault Isolation***, P_I is generated based on the following equation.

$$P_I = sgn(Expected\ Relative\ Acceleration) - sgn(Actual\ Relative\ Acceleration) \tag{4.11}$$

where $sgn(.)$ is the Sign Function. The actual relative acceleration is calculated by taking the second derivative of the relative distance sensor output as

$$Actual\ Relative\ Acceleration\ (t) = \frac{d^2r(t)}{dt^2} \tag{4.12}$$

If the parity signal $P_I = 0$, we conclude that the accelerometer is faulty and shows a wrong value for acceleration. On the other hand, if the parity signal is nonzero, the failure is from the thrusters. Finally, probing the accelerometer sensor output will give us the exact fault in the system. If all thrusters are stuck-off, the accelerometer will not pick any acceleration. Note that if the gyroscope fails, its data will become unavailable. It means that we can no longer calculate the angular position of the spacecraft.

In the next section, we will discuss more about the fault detection and isolation procedure in the system.

4.4 Simulation Results

We simulate our system using MATLAB/SIMULINK software and use DECK Toolbox [1] for constructing the DES model of the system. In the following, we assume different maneuvers for the system and analyze the response of the follower in both normal and faulty modes of operation.

4.4.1 Normal Mode of Operation

In this scenario, there is no fault in the system and all the components are functioning well. The follower follows the leader maneuver while keeping its relative distance and angular position in a desired range as shown in Figure 4.15. In this scenario, the leader rotates counterclockwise around 20 degrees at time $t = 0.8\times10^4$ and changes its rotation to clockwise at time $t = 1.2\times10^4$. It rotates till the time $t = 2.6\times10^4$ where it stops rotating and starts moving to the right direction. As we can see, the follower follows the leader maneuver very well. The first signal, shows the positions of both spacecraft with respect to their initial positions. The second graph depicts their angular positions. Parity signals are shown in the third and the last figures and we can see that they are both zero, as we expected.

4.4.2 Faulty Mode of Operation

Now, we examine different failure scenarios for the system and observe its behavior.

Accelerometer Bias

Figure 4.16 shows the parity signals for accelerometer failure. In Figure 4.16(a) the failure happenes while the follower spacecraft is rotating. In Figure 4.16(b) the failure happens after the follower rotation while translating. We can see that the behavior of the system, the maneuver and angle, is not affected by this failure.

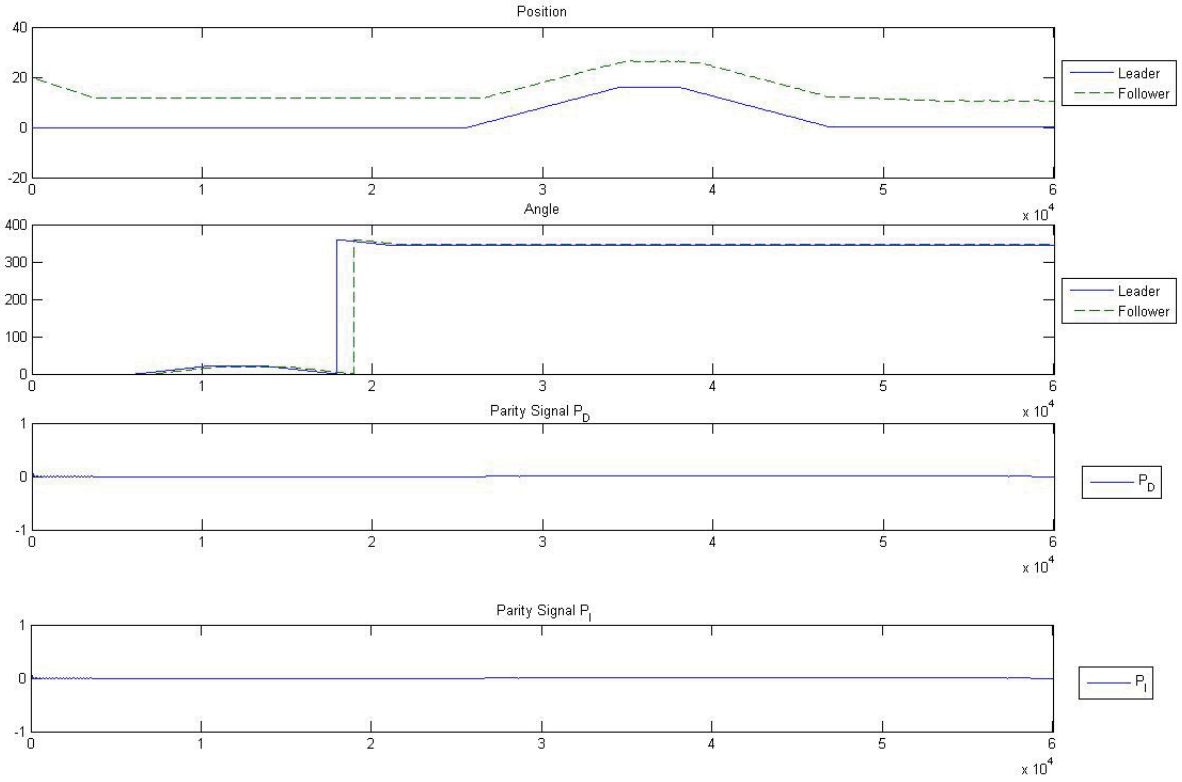


Figure 4.15: Normal Operation Mode

However, the P_D parity signal becomes nonzero and this shows the fault in the accelerometer.

Thruster T_1 Stuck-on

Figure 4.17 shows the parity signals for thruster T_1 failure. In Figure 4.17(a) the failure happenes while the follower spacecraft is rotating. In Figure 4.17(b) the failure happens after the follower rotation, while changing position. We can see that the behavior of the system is affected by this failure. Also, both parity signals becomes nonzero and this shows the fault in the thrusters.

All Thrusters Stuck-off

Similarly, Figure 4.18 shows the parity signals for all thrusters failure. In Figure 4.18(a) the failure happenes while the follower spacecraft is rotating while In Figure 4.18(b) the failure happens after the follower rotation, while changing position. We can see this fault affects the behavior of the system and also made both parity signals nonzero, as we expected.

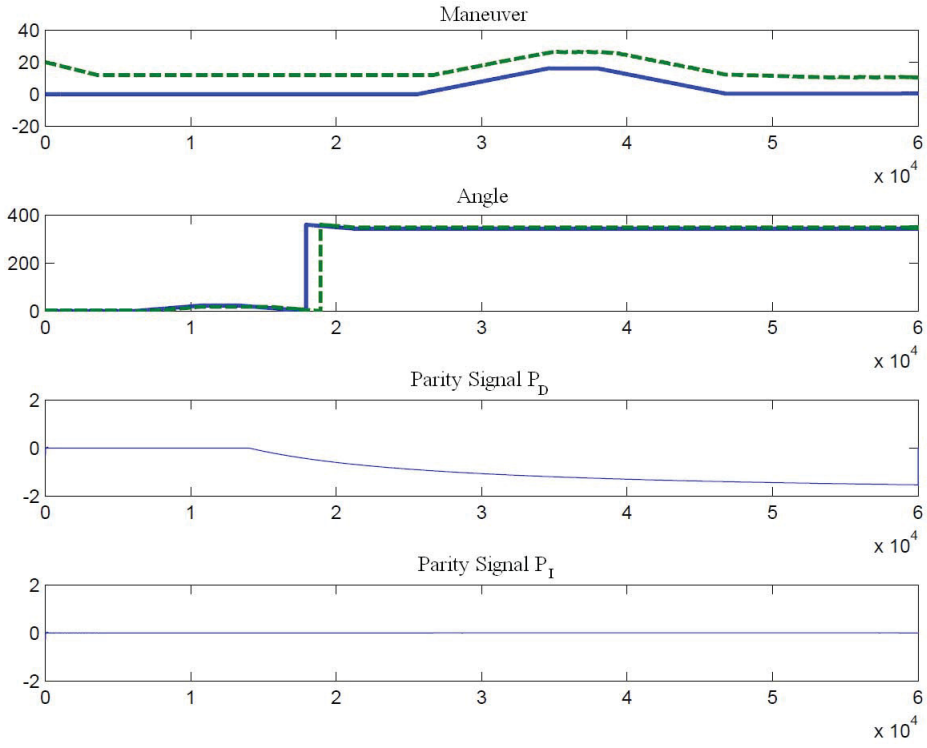
Gyroscope Failure

Figure 4.19 shows the system behavior for gyroscope failure. We can see that although the maneuver and the parity signals did not change, the angle data becomes unavailable.

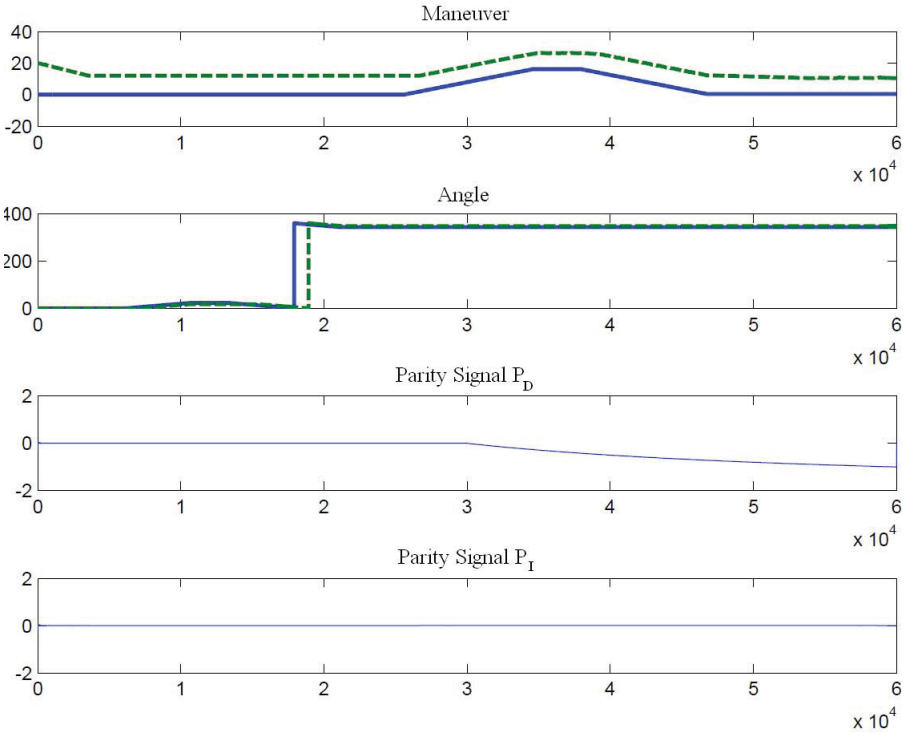
4.4.3 Remarks

Remark 1 - The impact of discretization

In this chapter, we have discretized the output of the relative distance sensor, accelerometer and spacecraft angular position into different discrete levels. Changing the levels of discretization of the angular position will increase the speed of diagnosis

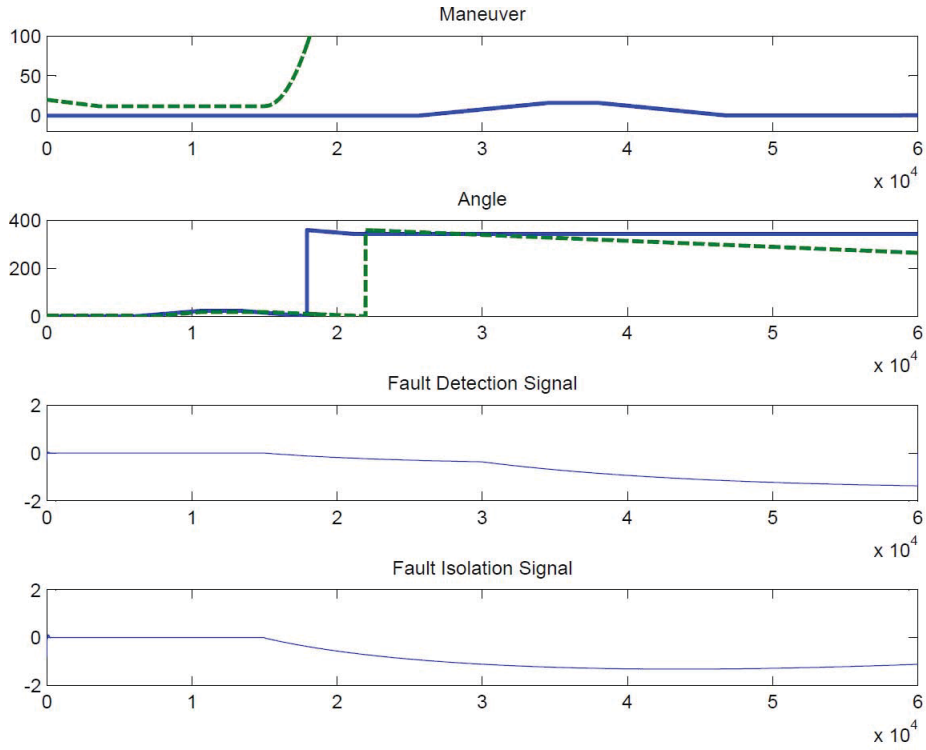


(a)

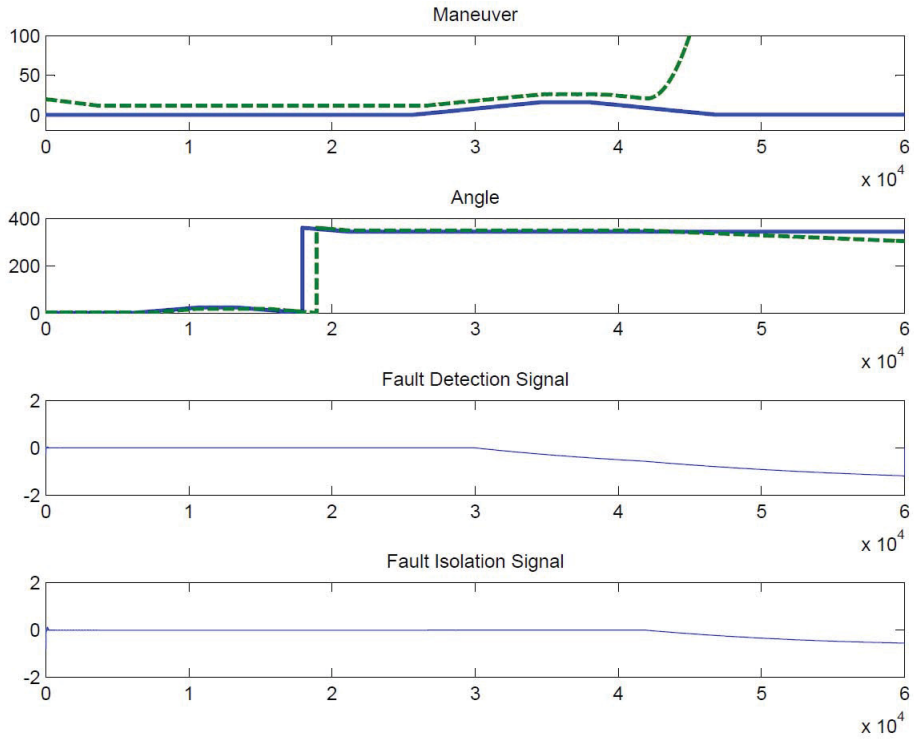


(b)

Figure 4.16: Accelerometer Failure: (a) Failure while Rotating, (b) Failure While Moving

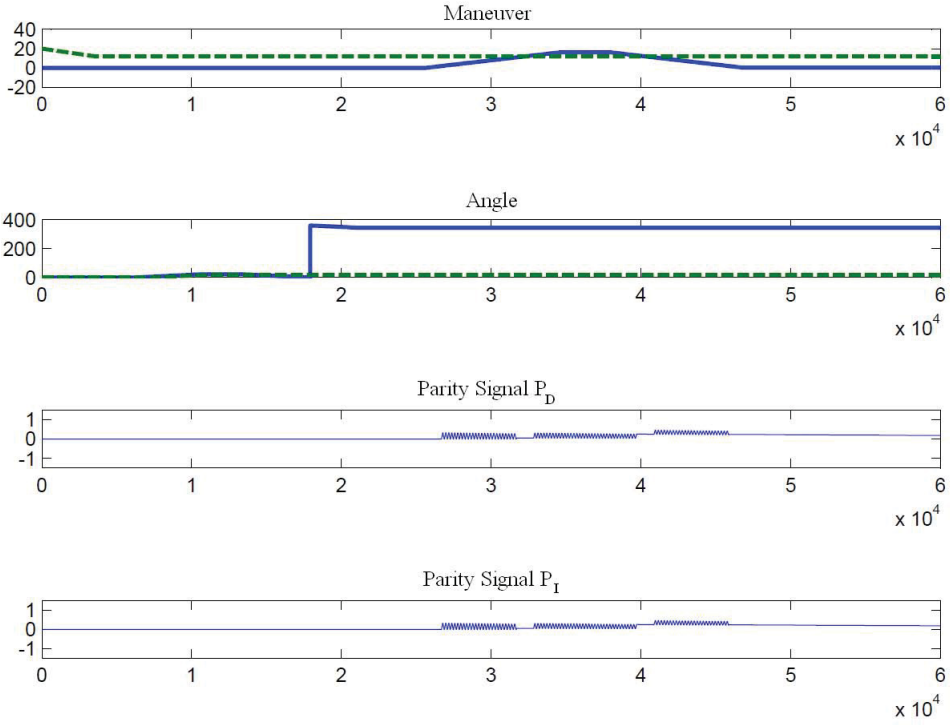


(a)

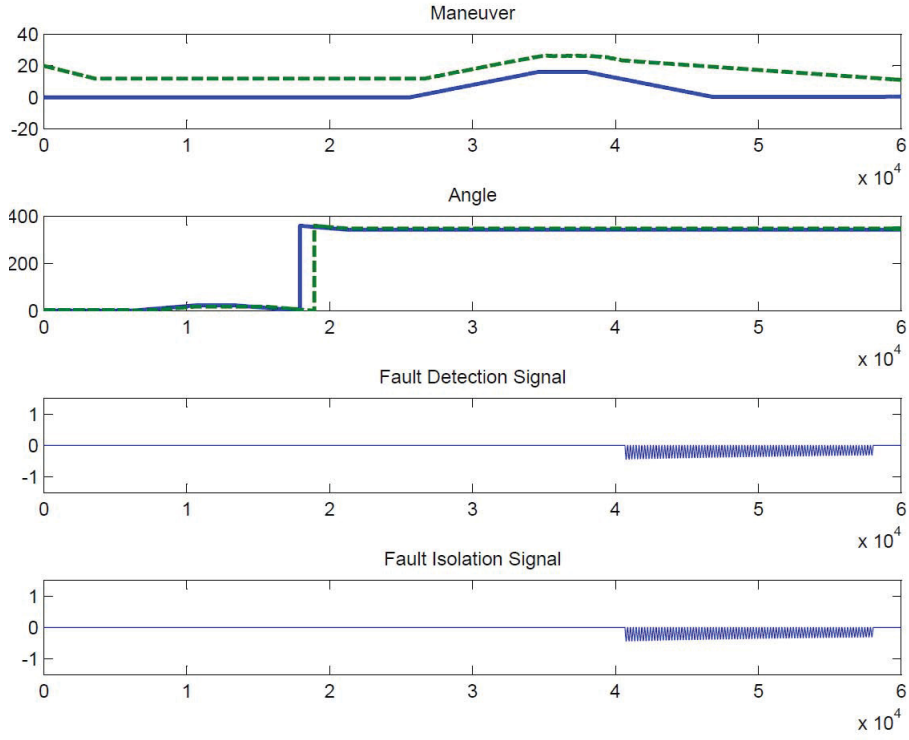


(b)

Figure 4.17: Thruster T_1 Failure: (a) Failure while Rotating, (b) Failure While Moving



(a)



(b)

Figure 4.18: All Thrusters Failure: (a) Failure while Rotating, (b) Failure While Moving

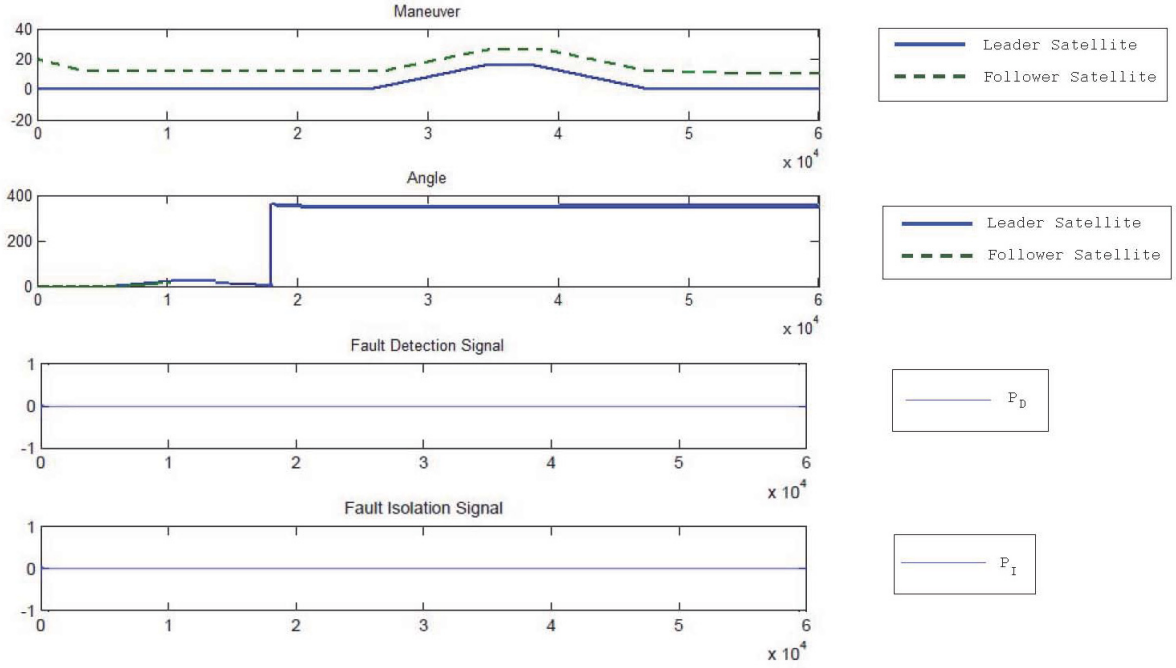


Figure 4.19: Gyroscope Failure

in this system. For example, there is a delay in the change of parity signals in Figure 4.17 after the fault occurs. If we increase the level of discretization of the leader angular position, the parity signal P_I will change sooner after the presence of the fault. This will reduce the delay of diagnosis. In general, finer discretization may also improve diagnosis accuracy. The price for faster or more accurate diagnosis, however, is the higher complexity of the discrete event models.

Remark 2 - Spacecraft mass

In Eq. (4.2) we assumed that the moment of inertial of the spacecraft was constant and we normalized its value to 1. If we considered variable moment of inertial for the spacecraft, the generation of the parity signals and hence the diagnosis system would remain the same.

4.5 Summary

In this chapter, we examined the proposed hybrid fault diagnosis scheme on a system of Leader/Follower spacecraft in deep space with translation motion in one direction and rotation around a axis perpendicular to translational motion using computer simulations. As pointed out in the previous chapter, a formal test of diagnosability is also available but requires extensive software development. This is left for future work.

We described the system components and reviewed the dynamics of the system. We then developed the DES model of the system. Parity signals were also formulated from sensors output signals. Subsequently, the fault diagnoser was built based on the system model and parity signals. We considered one normal scenario and four fault scenarios for the system and investigated the performance of diagnoser for each mode of operation. We also discussed how to isolate the faults by applying parity

signals.

Our study was limited to one-dimensional motion with single-axis rotation for a system whose dynamics is described in terms of simple integration. Therefore the scope of our study as indeed limited. A discussion on possible future extensions is provided in the next chapter.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we proposed an adaption of a framework for *fault diagnosis of hybrid systems* for teams of autonomous systems. The dynamics of hybrid systems are characterized by a *Discrete Event System (DES)* representing transitions among various modes of operation, and a set of continuous models (i.e. differential equations) describing the system's behavior in the discrete modes. In a hybrid system, there are usually two types of sensors: continuous sensors (generating continuous-time readings) and discrete sensors (generating discrete outputs like 'high' and 'low'). Discrete outputs can be used for the diagnosis of drastic failures such as a stuck-off actuator and continuous outputs can be used for the diagnosis of faults that slightly change the system dynamics such as a small loss of effectiveness in a sensor.

First, the DES model of the hybrid system is constructed based on the discrete states of the system. A bank of residuals is then generated based on the continuous models of the system to detect the faults. Then, we model these residuals by DES and integrate them with the DES abstraction of the system to construct an extended DES model. Finally, we use this extended model to build the fault diagnoser.

In the fault diagnosis scheme proposed here for a team of autonomous systems, each member of the team has a local diagnoser and to reduce the amount of communication between the team members, only discrete-event data are exchanged among the local diagnosers.

The proposed fault diagnosis system is applied to a team of two spacecraft in a *Leader-Follower Formation Flying* format. Each spacecraft has a set of four thrusters, one accelerometer, one relative distance sensor and a gyroscope. We assume that the leader and relative distance sensor are fault free while follower thrusters, accelerometer and gyroscope may fail.

Our goal is to maintain their relative position and orientation at a desirable range while detecting faults at the moment they occur. To minimize the communication between these two spacecraft, we keep the continuous information locally and only communicate the discretized data to the other spacecraft. The fault diagnosis for the follower is designed. A number of simulations for different maneuvers were conducted to demonstrate and verify the performance of our proposed hybrid fault diagnoser.

5.2 Future Work

In the following, some of the possible extensions to the results obtained in this thesis are presented.

- *Formation Flying* - In this work we first considered a formation flying of two spacecraft with translation motion in one axis in Chapter 3. Then, we added rotational movements around a fixed axis in Chapter 4. We considered this simple example to show the basic idea of our proposed method and examine the performance of it in different maneuvers. In general, in formation flying, the number of spacecraft is more than two and also they move in 3-dimensional

space.

- *Thrusters* - In our work, we considered on/off thrusters with bang-bang control as these were more suitable for our diagnosis method. If, for example, continuous thrusters are suggested for an application, we should change the system's mathematical model to incorporate the continuous behavior of thrusters. This will lead to a more computationally complex model and is left as a future work.

- *Disturbance and Noise* - We assumed that the system is disturbance and noise-free. Investigating the performance of the proposed fault diagnosis method in the presence of disturbance and noise and uncertainties in the system's model is also an interesting topic for future work.

Bibliography

[1] Shahin Hashtrudi Zad, *Discrete Event Control Kit*. Department of Electrical and Computer Engineering, Concordia University, 2012.

Available at `http://www.ece.concordia.ca/~shz/deck`.

[2] Charles H. Goodrich, James Kurien, "*Continuous measurements and quantitative constraints – challenge problems for discrete modeling mechniques*". i–SAIRAS 2001, Canadian Space Agency, St-Hubert, Quebec, Canada, June 18-22, 2001.

[3] J. E. Hopcroft, R. Motwani and J. D. Ullman, "*Introduction to automata theory, languages, and computation*". Addison-Wesley, 3rd edition, 2006.

[4] K. H. Johansson, J. Lygeros, S.N. Simic, J. Zhang and S. Sastry, "*Dynamical properties of hybrid automata*". IEEE Trans. on Automatic Control, vol. 48, no. 1, pp. 2-17, 2003.

[5] M. A. Massoumnia, "*A geometric approach to failure detection and identification in linear systems*". Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[6] C. De Persis and A. Isidori, "*A geometric approach to nonlinear fault detection and isolation*". IEEE Trans. on Automatic Control, vol. 46, no. 6, pp. 853-865, 2001.

[7] Rasul Mohammadi, "*Fault diagnosis of hybrid systems with applications to gas turbine engines*". Ph.D Dissertation, Concordia University, Montreal, QC, CA, 2009.

[8] Daniel P. Scharf, Fred Y. Hadaegh and Scott R. Ploen, "*A survey of spacecraft formation flying guidance and control (part II): control*". Proceedings of American Control Conference, pp. 29762985, 2004.

[9] Navid Dadkhah, Luis Rodrigues and Amir G. Aghdam, "*Satellite formation flying controller design using an optimal decentralized approach*". Proceedings of American Control Conference, pp. 31623167, 2007.

[10] P.Wang, "*Navigation strategies for multiple autonomous mobile robots moving in formation*". Journal of Robotic Systems, vol. 8, no. 2, pp. 177195, 1991.

[11] J. A. Fax and R. M. Murray, "*Information flow and cooperative control of vehicle formations*". IEEE Transaction Automatic Control, vol. 49, no. 9, pp. 14651479, 2004.

[12] F. Giulietti, L. Pollini, and M. Innocenti, "*Autonomous formation flight*". IEEE Control Systems Magazine, vol. 20, no. 6, pp. 3444, 2000.

[13] D. M. Stipanovic, G. Inalhan, R. Teo, and C. J. Tomlin, "*Decentralized overlapping control of a formation of unmanned aerial vehicles*". Automatica, vol. 40, no. 8, pp. 12851296, 2004.

[14] W. Li, W.H. Gui, Y.F. Xie and S.X. Ding, "*Decentralized fault detection of large-scale systems with limited network communications*". IET Control Theory and Applications, vol. 4, no. 9, pp. 18671876, 2010.

[15] S. Azizi, K. Khorasani, "*A hierarchical architecture for cooperative actuator fault estimation and accommodation of formation flying satellites in deep*

space". IEEE Transactions on Aerospace and Electronic Systems, vol. 48, no. 2, pp. 1428-1450, 2012.

[16] Roy S. Smith and Fred Y. Hadaegh, "*Distributed control topologies for deep space formation flying spacecraft*". International Symposium on Formation Flying: Missions and Technologies Toulouse, France, 2002.

[17] Folta, Newman, and Gardner, "*Foundations of formation flying for mission to planet earth and new millenium*". AIAA-96-3645-CP, NASA Goddard Space Center, 1996.

[18] R.C. Arkin, "*Behavior-based robotics*". Cambridge, MA, MIT Press, 1998.

[19] P. M. Frank and X. Ding, "*Survey of robust residual generation and evaluation methods in observer-based fault detection systems*". Journal of Process Control, vol. 7, no. 6, pp. 403-424, 1997.

[20] R. J. Patton and J. Chen, "*Observer-based fault detection and isolation: robustness and applications*". Control Engineering Practice, vol.5, no. 5, pp. 671-682, 1997.

[21] S. Azizi, K. Khorasani, "*A sub-optimal distributed Kalman filter with fusion feedback for acyclic systems*". IEEE Conf. on Decision and Control, pp. 5151-5157, 2009.

[22] P. M. Frank, "*Analytical and qualitative model-based fault diagnosis-A survey and some new results*". European Journal of Control, pp. 6-28, 1996.

[23] R. Milne, "*Strategies for diagnosis*". IEEE Transactions on Systems, Man and Cybernetics, vol. 17, no. 3, pp. 333-339, 1987.

[24] J. Jr. Sottile and L. E. Holloway, "*An Overview of Fault Monitoring and Diagnosis in Mining Equipment*". IEEE Transactions on Industry Applications, vol. 30, no. 5, pp. 1326-1332, 1994.

[25] A. S. Willsky, "*A survey of design methods for failure detection in dynamic system*". Automatica, vol. 12, pp. 29-32, 1976.

[26] R. Isermann, "*Fault diagnosis systems: an introduction from fault detection to fault tolerance*". Springer, 2006.

[27] W. Li, S. Shah, "*Data-driven kalman filters for on-uniformly sampled multirate systems with application to fault diagnosis*". Proceedings of American Control Conference, vol. 4, pp. 2768-2774, 2005.

[28] J.J. Gertler, "*Fault detection and isolation in engineering systems*". CRC Press, 1998.

[29] R. J. Patton, P. M. Frank and R. N. Clark, "*Issues of fault diagnosis for dynamic systems*". Springer, 1st edition, 2000.

[30] R. Isermann, "*Supervision, fault detection and fault diagnosis methods - an introduction*". Control Engineering Practice, vol. 5, no. 5, pp. 639-652, 1997.

[31] S. Hashtrudi Zad, R.H. Kwong and W.M.Wonham, "*Fault diagnosis in discrete event systems: framework and model reduction*". IEEE Trans. on Automatic Control, vol. 48, no. 7, pp. 1199-1212, 2003.

[32] S. Hashtrudi Zad, R.H. Kwong and W.M.Wonham, "*Fault diagnosis and consistency in hybrid systems*". Proc. 38th Annual Allerton Conf. on Communication, Control, and Computing, University of Illinois at Urbana-Champaign, pp. 1135-1144, 2000.

[33] R. Patton, P. Frank and R. Clark, Eds., "*Fault diagnosis in dynamic systems: theory and applications*". New York: Prentice-Hall, 1989.

[34] R.J. Patton and J. Chen, Eds., "*Fault detection, supervision and safety for technical processes*". Proc. IFAC Symp., Kingston Upon Hull, UK, 1997.

[35] P. J. Antsaklis, X. D. Koutsoukos and J. Zaytoon, "*On hybrid control of complex systems: a survey*". European Journal of Automation, vol. 32, pp. 1023-1045, 1998.

[36] T. Koo, F. Hoffmann, H. Shim, B. Sinopoli and S. Sastry, "*Hybrid control of an autonomous helicopter*". IFAC Workshop on Motion Control, Grenoble, France, pp. 285-290, 1998.

[37] R. Alur, C. Courcoubetis, T.A. Henzinger and P.H. Ho, "*Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems*". Hybrid Systems, vol. 736 of Lecture Notes in Computer Science, pp. 209-229, Springer, 1993.

[38] F. Zhao, X. Koutsoukos, H. Haussecker, J. Reich and P. Cheung, "*Monitoring and fault diagnosis of hybrid systems*". IEEE Trans. on Systems, Man and Cybernetics - Part B, vol. 35, no. 6, pp. 1225-1240, 2005.

[39] S. McIlraith, G. Biswas, D. Clancy and V. Gupta, "*Hybrid systems diagnosis*". Hybrid Systems: Computation and Control, vol. 1790 of Lecture Notes in Computer Science, pp. 282-295, Springer-Verlag, 2000.

[40] F. Lin, "*Diagnosability of discrete event systems and its application*". Discrete Event Dynamic systems, vol. 4, pp. 197-212, 1994.

[41] S. Bavishi and E. K. Chong, "*Automated fault diagnosis using a discrete event systems framework*". Proc. of the 9th IEEE Intl. Symposium on Intelligent Control, Columbus, Ohio, USA, vol.16, no.18, pp. 213-218, 1994.

[42] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis, "*Diagnosability of discrete event systems*". IEEE Trans. Automatic Control, vol. 40, no. 9, pp. 1555-1575, 1995.

[43] P. Baroni, G. Lamperti, P. Pogliano and M. Zanella, "*Diagnosis of large active systems*". Arti

cial Intelligence, vol.110, no.1, pp. 135-183, 1999.

[44] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis, "*Failure diagnosis using discrete event models*". IEEE Trans. on Control System Technology, vol. 4, no. 2, pp. 105-124, 1996.

[45] P. M. Frank, "*Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - A survey and some new results*". Automatica, vol. 26, no. 3, pp. 459-474, 1990.

[46] A. S. Willsky, "*A survey of design methods for failure detection in dynamic systems*". Automatica, vol. 12, no. 6, pp. 601-611, 1976.

[47] J. Gertler, "*Analytical redundancy methods in fault detection and isolation - A survey and synthesis*". IFAC/IMACS SAFEPROCESS, pp. 9-21, 1991.

[48] E. Isermann, "*Process fault detection based on modeling and estimation methods - A survey*". Automatica, vol. 20, no. 4, pp. 387-404, 1984.

[49] J. Chen and R. Patton, "*Robust model-based fault diagnosis for dynamic systems*". Kluwer Academic Publishers, 1999.

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

[51] I. E. Potter and M. C. Sunman, "*Thresholdless redundancy management with arrays of skewed instruments*". Integrity in Electronic Flight Control Systems, AGARDOGRAPH-224, pp. 15-11 to 15-25, 1977.

[52] M. Desai and A. Ray, "*A fault detection and isolation methodology*". Proc. of the 20th IEEE Conf. on Decision and Control, San Diego, CA, USA, pp. 1363-1369, 1981.

[53] A. Misra, G. Provan, G. Karsai, G. Bloor and E. Scarl, "*A generic and symbolic model-based diagnostic reasoner with highly scalable properties*". Proc. of the IEEE Conf. on Systems, Man, and Cybernetics, vol. 4, pp. 3154 - 3160, San Diego, CA, USA, 1998.

[54] P. Mosterman and G. Biswas, "*Diagnosis of continuous valued systems in transient operating regions*". IEEE Trans on Systems, Man, and Cybernetics, vol. 29, pp. 554-565, 1999.

[55] J. Lygeros, C. J. Tomlin and S. Sastry, "*Controllers for reachability specifications for hybrid systems*". Automatica, vol. 35, pp. 349-370, 1999.

[56] M. A. Massoumnia, G.C. Verghese and A.S. Willsky, "*Failure detection and identification*". IEEE Trans. on Automatic Control, vol. AC-34, no. 3, pp. 316-321, 1989.

[57] G.K. Fourlas, K.J. Kyriakopoulos and N.J. Krikelis, "*Model-based fault diagnosis of hybrid systems based on hybrid structure hypothesis testing*". Proc. of the 11th IEEE Mediterranean Conf. on Control and Automation, Rhodes, Greece, 6 pages, 2003.

[58] M.M. Tousi, A.G. Aghdam and K. Khorsani, "*'A Hybrid Fault Diagnosis for a Team of Unmanned Aerial Vehicles*". IEEE International Conference on System of Systems Engineering, 2009.

[59] E.J. Manders, S. Narasimhan, G. Biswas and P.J. Mosterman, "*A combined qualitative/quantitative approach for efficient fault isolation in complex dynamic systems*". Proc. of the 4th symposium on Fault Detection, Supervision, and Safety Processes, Budapest, Hungary, pp. 512-517, 2000.

[60] J. Lunze, "*Diagnosis of quantized systems by means of timed discrete event representations*". IEEE Tran. on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 30, no. 3, pp. 322-335, 2000.

[61] S. Narasimhan and G. Biswas, "*Model-based diagnosis of hybrid systems*". IEEE Tran. on System, Man, and Cybernetics, Part A: Systems and Humans, vol. 37, no. 3, pp. 348-361, 2007.

[62] W.M.Wonham, "*Supervisory control of discrete-event systems*". Department of Electrical and Computer Engineering, University of Toronto, July 2012.

[63] C.G. Cassandras and S. Lafortune, "*Introduction to discrete event systems*". Springer, 2008.