Summer 2002

Analysis of Error Recovery Effects on Digital Flight Control Systems

Arturo Tejada Ruiz
Old Dominion University

ANALYSIS OF ERROR RECOVERY EFFECTS ON DIGITAL FLIGHT CONTROL SYSTEMS

by

Arturo Tejada Ruiz
B.S.E.E. December 1996, Pontificia Universidad Católica del Perú

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
August 2002

Approved by:

Oscar R. González (Director)

W. Steven Gray (Member)

Linda L. Vahala (Member)

ABSTRACT

ANALYSIS OF ERROR RECOVERY EFFECTS ON DIGITAL FLIGHT CONTROL SYSTEMS

Arturo Tejada Ruiz
Old Dominion University, 2002
Director: Dr. Oscar R. González

Life-critical, real-time applications like flight-by-wire aircraft, rely on closed-loop digital control systems and fault-tolerant computer systems to reliably achieve the desired operation. The computer systems, however, may be affected by random hardware/software faults induced mainly by environmental conditions such as high intensity radiated fields (HIRF) and lightning. These harsh electromagnetic environments are known to induce common-mode faults (CMF) in aircraft electronic systems, which disrupt fault-tolerant provisions and possibly affect the operation of the digital control system. Current flight-by-wire aircraft have computer systems that can neither detect CMF nor recover from them. New systems are under investigation that can recover from CMF using error recovery techniques. Nevertheless, little is known about the effect of these recovery algorithms on the stability of the closed-loop flight control system.

The main objectives of this research are to analyze the stability of closed-loop discrete-time digital flight control systems with error recovery capabilities that are triggered by a harsh electromagnetic environment and to compare the strengths and weaknesses of typical fault recovery methods. This research is an extension of previous work in which the environment is assumed to induce abrupt changes on the structure of the closed-loop representation, but the effect of error recovery systems was not considered. Three error recovery algorithms were considered: rollback, reset and cold-restart. For each case new closed-loop models that include their effect were developed. These models were then used with existing

stochastic stability theory to determine the effect on stability. The theoretical analysis was then validated with Monte Carlo simulations, including two aircraft examples. An important consequence of this research is the availability of a new design tool that allow designers of error recovery systems to compare the benefits of the recovery algorithms and the impact of their parameters on the stability of the closed-loop flight control systems.

Quiero dedicar esta tesis a mis padres Urbano y Margarita:
Gracias por 28 años de amor, apoyo, aliento y compresion. Lo que soy y he logrado se los debo a ustedes. Los quiero mucho.

Arturo

Agosto 2002

(Esta vez no tomó 3 años...)

This thesis is dedicated to my parents, Urbano and Margarita:
Thanks for 28 years of love, support and understanding. What I am and have archived is because of you. I love you.

Arturo

August 2002

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Oscar R. González, for giving an inexperienced student (me) the opportunity to participate in this important research project. His exemplary professionalism and commitment to research were my inspiration and motivation to complete this research.

Dr. W. Steven Gray also has my deepest appreciation for his great ideas and support during this research and for introducing me to the delights of coffee. I also sincerely thank Dr. Linda L. Vahala for serving on my thesis advisory committee, with such a short notice.

I am very grateful to my family and friends for their support through the years it took to complete this work. I especially want to thank Milagros, who gave me the strength to carry on during my first difficult year as a foreign student; Ashutosh, who allowed me to enjoy and see life from a new perspective; and MayLing, who holds a special place in my heart.

Finally, I wish to acknowledge the support of the NASA Langley Research Center under grant NCC-1-392 and the support of the Presidential Dominion Scholar Fellowship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation for this Research

The increasing number of life-critical, real-time applications makes the study of fault handling in computer systems a very important and active research area. Modern high-reliability systems are designed to tolerate random hardware faults in their components by the use of hardware and/or software redundancy and reconfiguration. The basic assumption is that the faults can disrupt some of the replicated devices inside the system but not all of them. In this way, the system can continue to operate or in the worst case degrade to a minimum safe level of performance. This assumption has been very successful in handling a large class of random hardware and software faults. The techniques used to introduce fault tolerance and recovery are very mature (in particular, the one called Byzantine Resilience [33]). The majority of these techniques were developed by the microelectronic and computer science communities. Nevertheless, the situation is different for modern commercial fly-by-wire aircraft, which are subject to electromagnetic interference (EMI) from sources such as high intensity radiated fields and lightning [3]. The faults induced by EMI can affect simultaneously several aircraft electronic systems [25] and jeopardize the flight integrity. In [26, 35] the simulation results show how EMI-induced faults could crash a commercial airliner (a Boeing 737) during an automatic landing. Since these harsh electromagnetic environments are able to introduce transient errors in all the aircraft's redundant modules

[1]The journal model used for this work is the *IEEE Transactions on Circuits and Systems Part I: Fundamental Theory and Applications*

at the same time, the faults are known as common-mode faults (CMF). These faults can render the standard fault-tolerant provisions ineffective.

Several research efforts have been developed inside the aerospace community to understand how these CMFs affect the performance of aircraft flight controllers. In [31] the authors presented a model of computer upsets with respect to their arrival and duration, assuming that the affected computer system is composed of N replicated modules and were subject to internal random hardware faults and externally induced CMF. In [14–19] Gray and González presented the first theoretical tools for the analysis of the impact of CMF induced by EMI on the performance of a closed-loop digital aircraft flight controller. Their problem setup is slightly different to the one in [31]. In [21, 22] a digital dual lock-step processor executes the aircraft's flight control algorithm. This processor is then hit by electromagnetic radiation (a source of common-mode faults), modeled by a birth and death process, which produces computational errors in the flight control computer with a certain probability. The errors are assumed to modify the parameters of the closed-loop state space representation in a random fashion. These changes were then modeled with a discrete-time Markovian jump linear model, whose stability is analyzed with the (slightly modified) results from [6, 8].

The research results in [14–19] do not directly address the effect of embedded digital systems with recovery capabilities. In these systems, provisions are made in order to restore their correct behavior when faults are detected. Hence, faults in systems with recovery capabilities modify the closed-loop state space representation with pre-established values.

The research presented in this thesis extends the ideas in [14–19] to include the behavior of three recovery methods: rollback, reset and cold-restart. The new mathematical models are then used to analyze and compare the stability and performance characteristics of these

three methods through several case studies.

1.2 Problem Statement

The main research goals of this thesis are:

1. To analyze the stability of closed-loop discrete-time digital flight control systems with error recovery capabilities when affected by electromagnetically-driven random faults.

2. To compare the strengths and weaknesses of typical fault recovery methods.

The first goal is very close to the motivation for [9, 18]. Hence, that research is the base of the investigation presented here and has been modified and extended to include the fault recovery mechanism explained in later chapters.

The second goal is a logical consequence of the first one. Several new computational tools that use the new theory have been devised. These tools allow designers to test several configurations of one recovery method or to compare the performance of different recovery methods.

1.3 Overview of the Thesis

The results of this research are presented as follows: Chapter 2 summarizes the mathematical foundation and the tools used in Chapter 3 to analyze the stability of the control systems with fault recovery mechanisms. Chapter 3 presents the main contributions of this research. It includes a literature review of fault-tolerant systems, an explanation on how the theory developed in [9, 18] was adapted to model the fault-tolerant problem and the new theorems and corollaries required for stability analysis. In Chapter 4, three simulation

studies are presented in detail. These examples include two aircraft systems: The B737 airliner and the AFTI/F-16 fighter. The last chapter presents the conclusions of this research and future research directions.

CHAPTER II

ON THE STABILITY OF DIGITAL CLOSED-LOOP SYSTEMS SUBJECT TO EM PERTURBATIONS

2.1 Introduction

The objective of this chapter is to introduce the mathematical tools needed to understand the developments in [9, 14–19] and to extend them in the directions required by our first goal in Section 1.2.

As mentioned in the first chapter, [9, 14–19] contains a characterization of the electromagnetic environment and its interference effect on the digital controller. This is summarized in Section 2.2. The environment (exosystem) is represented by a birth and death process in which the births correspond to the random arrival of radiation (Poisson distributed) and the deaths correspond to the removal of the radiation after an exponentially distributed period of time. The exosystem is then sampled (because it can only affect the closed-loop dynamics at the sample instants) and the resulting Markov chain is assumed to drive the parameter changes of the closed-loop space-state representation of the system. Hence, the interference of the exosystem on the digital controller is envisioned as a one-to-one mapping from the state of the chain to the state space representation of the controller. The resulting mathematical description of the closed loop system is known as a discrete-time Markovian jump linear system (DTMJLS).

The last section of the chapter presents the theorems required to analyze the mean square stochastic stability of DTMJLS for several different conditions and some corollaries that simplify the computation of the stability thresholds.

2.2 Characterization of the EM environment

Gray-et al. in [18] show how, under certain simplifications, the effect of an electromagnetic environment in a closed-loop computer control system can be represented by a discrete time Markov chain. This model gives the mapping of the state of the environment $\theta(k)$ to changes on the parameters of the closed-loop control system. This *mapping* constitutes the *interference model* which can be customized to represent the different ways in which the control system changes in response to the environment. For example [18, 19] consider that the EM environment produces random changes in the state equations that describe the dynamics of the system. In that case each state of the environment is mapped to a set of random perturbation state matrices. If the faults produced by the environment trigger a particular fault recovery mechanism, then the interference model will map the environment states to deterministic state matrices. This system, which is switched from a nominal representation to a alternative (recovery) representation, belongs to the class of jump linear systems whose stability can be analyzed with the theory presented in the following sections.

2.2.1 Problem Outline

The basic elements of the problem under consideration are shown in Figure 2.1. An aircraft with a digital flight controller is assumed to be immersed in a high intensity electromagnetic environment induced by surrounding conditions such as lightning and radar signals. If these conditions are strong enough, they may induce upsets in the digital controller. An example of these upsets is the change of the contents of the data stored in devices such as memories or registers. In general, these upsets can be masked by hardware/software protection (like cable shielding, triple modular redundancy or parity correcting codes), but not in the case of CMF. In that case the upsets result in errors in the control signals.

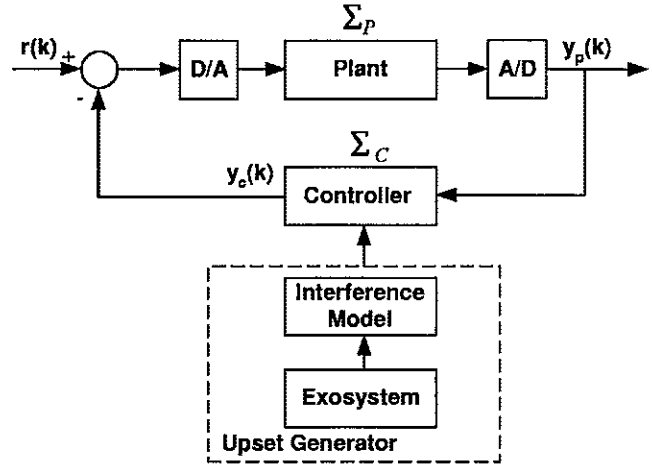

Figure 2.1: Closed-loop flight control system with proposed EM disturbance model.

The main assumptions on the hardware effect of the upsets are formalized in the following statement:

Assumption 1 *All upset conditions are mild enough to prevent the system from going into a permanent failure. Thus, during each sample period a control calculation is performed, but possibly erroneously. It is assumed that the upset introduces errors that can be modeled as a perturbation in the ideal control law, an additive noise disturbance to the ideally computed control signal, or a combination of both.*

For modeling purposes, the upsets can be considered to be the outputs of an upset generator. The generator is excited by external disturbances, the so-called exosystem. The conversion of the disturbances into upsets is characterized by the interference model. The next two subsections will summarize the descriptions of the exosystem disturbance model as presented in [9, 18].

2.2.2 Upset Generator: The Exosystem

The exosystem is the complex electromagnetic environment that affects an aircraft during operation. A useful model for this environment is to quantize the number of significant radiation events that hit the control system. A significant event is one that can cause an upset. At any specific time instant $t \in \mathbb{R}$, let $\boldsymbol{N}(t)$ [1] denote the number of significant EM disturbances. In this model, the i-th disturbance is characterized by its arrival time, $\boldsymbol{t_i}$, and its total duration, $\boldsymbol{d_i}$. It is assumed that the random variables $\boldsymbol{t_i}$ constitute a Poisson process with constant parameter λ, the $\boldsymbol{d_i}$ have an exponential distribution with parameter μ and $\boldsymbol{N}(t)$ is a memoryless Markovian continuous-time random process with transition probability rates

$$P\{\boldsymbol{N}(t+\Delta t) = \kappa + 1 | \boldsymbol{N}(t) = \kappa\} \approx \beta_\kappa \Delta t$$

$$P\{\boldsymbol{N}(t+\Delta t) = \kappa - 1 | \boldsymbol{N}(t) = \kappa\} \approx \delta_\kappa \Delta t.$$

The parameters β_κ and δ_κ are the *birth* and *death rates* respectively. They determine the probability of adding or removing a disturbance in the near future given the current number of disturbances. The remaining transition probability rates are taken to be zero so that the set of all transition rates can be represented by a tridiagonal matrix Λ. It follows directly that $\beta_\kappa = \lambda$ for all $\kappa \geq 0$ and, as discussed in [9,14–19], the death rate is assumed to be proportional to κ ($\delta_\kappa = \kappa\, \delta$). It can be shown that this choice of parameters will give the process $\boldsymbol{N}(t)$ statistics equivalent to those of a $(M|M|\infty)$ queue. The queue's equilibrium state probabilities, p_κ, and the transition probability rates, $\Lambda_{i,j}$, are summarized in Table 2.1, where $\rho = \lambda/\mu$. Due to the digital nature of the controller, any upset produced by the environment can only affect the dynamical response at discrete time instants kT where T

[1] Boldface is used to denote random variables and processes.

Table 2.1: Equilibrium state probabilities and transition probability rates associated with the $(M|M|\infty)$ Markovian exosystem.

Equilibrium State Probabilities	Transition Probability Rates
$p_\kappa = e^{-\rho}\frac{\rho^\kappa}{\kappa!},\ \kappa \geq 0$	$\Lambda_{\kappa,\kappa+1} = \lambda$
	$\Lambda_{\kappa,\kappa-1} = \kappa\,\mu$

represent the controller's clock period and $k \in \boldsymbol{Z}^+$. Let $\boldsymbol{\theta}(k) = \boldsymbol{N}(kT)$ represent the state of the exosystem at each sample instant (that is, each time the controller's clock ticks). Then it can be shown (see references 16 and 17 in [18]) that if T is sufficiently small, $\boldsymbol{\theta}(k)$ becomes a discrete-time Markov chain. What follows is the definition of a discrete-time Markov chain.

Definition 2.2.1 *A discrete-time Markov chain $\boldsymbol{\theta}(k)$ is a stochastic process than can take values, at certain time instants, in the set $\mathcal{S}_1 \subset \boldsymbol{Z}^+ := \{0, 1, 2, \ldots\}$, with transition probability matrix $\Pi = [\pi_{i,j}]$, $i, j \in \mathcal{S}_1$, such that:*

$$\begin{aligned} 0 \leq \pi_{ij} &= P\{\boldsymbol{\theta}(k+1) = j | \boldsymbol{\theta}(k) = i\} \leq 1, \\ \sum_j \pi_{ij} &= 1,\ \forall i \in \mathcal{S}_1. \end{aligned} \tag{2.1}$$

Hence, to consider $\boldsymbol{\theta}(k)$ a Markov chain, the following assumption must hold:

Assumption 2 *The sample period T is sufficiently small such that the process $\boldsymbol{\theta}(k)$ can be approximated as a discrete-time Markov chain with transition probability matrix $\Pi = e^{\Lambda T}$.*

2.2.3 Upset Generator: Interference Model

As mentioned in Section 2.1, the interference caused by the exosystem over the closed-loop space-state model of the system can be represented by a discrete-time Markovian jump linear system whose formal definition is presented next.

Definition 2.2.2 *A Discrete-Time Markovian Jump Linear System (DTMJLS) can be represented by the following dynamical equation:*

$$\boldsymbol{x}(k+1) \;=\; A_{\boldsymbol{\theta}(k)}\boldsymbol{x}(k) + B_{\boldsymbol{\theta}(k)}[\boldsymbol{w}^T(k)\; r^T(k)]^T \tag{2.2}$$

where $k \in \boldsymbol{Z}^+ := \{0, 1, 2, \ldots\}$ *and the initial conditions are independent random variables* $\boldsymbol{x}(0) = \boldsymbol{x}_0$ *and* $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$. $\boldsymbol{x}(k) \in \mathbb{R}^{\mathrm{n}}$ *is the state vector of the system and* $[\boldsymbol{w}^T(k)\; r^T(k)]^T \in \mathbb{R}^{\mathrm{m}}$ *is the input that consists of a random disturbance* $\boldsymbol{w}(k)$ *and deterministic input* $r(k)$.

The matrices $A_{\boldsymbol{\theta}(k)} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$, $B_{\boldsymbol{\theta}(k)} \in \mathbb{R}^{\mathrm{m}\times\mathrm{m}}$ *are a function of* $\boldsymbol{\theta}(k)$, *which is a homogeneous discrete-time Markov chain taking values in the finite set* $\mathcal{S}_1 = \{0, 1, \ldots, N-1\}$.

Now, assume that the plant is modeled at the sampling instants by the sampled-data system given by

$$\begin{aligned} x_p(k+1) &= A_p x_p(k) + B_p u_p(k) \\ y_p(k) &= C_p x_p(k), \end{aligned} \tag{2.3}$$

where $A_p \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$, $B_p \in \mathbb{R}^{\mathrm{n}\times\mathrm{m}}$, $C_p \in \mathbb{R}^{\mathrm{p}\times\mathrm{n}}$ and $x_p(k)$ is the plant's state vector. The control law equation is given by:

$$u_p(k) \;=\; r(k) - y_c(k),$$

where $y_c(k)$ is the output of the controller. The nominal state-space representation of the controller is

$$\begin{aligned} x_c(k+1) &= A_c x_c(k) + B_c y_p(k) \\ y_c(k) &= C_c x_c(k), \end{aligned} \tag{2.4}$$

where x_c is the controllers state vector and A_c, B_c, C_c are the controller's state feedback, input and output gain matrices respectively.

When no radiation is present, the nominal closed-loop system is denoted by (A_0, B_0, C_0) and its closed-loop state space representation is given by

$$\begin{aligned} x_{CL}(k+1) = \begin{bmatrix} x_p(k+1) \\ x_c(k+1) \end{bmatrix} &= A_0 \begin{bmatrix} x_p(k) \\ x_c(k) \end{bmatrix} + B_0[w^T(k)\ r^T(k)]^T \\ y_{CL}(k) &= C_0 \begin{bmatrix} x_p(k) \\ x_c(k) \end{bmatrix}, \end{aligned} \tag{2.5}$$

where

$$\begin{aligned} A_0 &= \begin{bmatrix} A_p & -B_p C_C \\ B_c C_p & A_c \end{bmatrix}, \\ B_0 &= \begin{bmatrix} 0 & B_P \\ 0 & 0 \end{bmatrix}, \text{ and} \\ C_0 &= [C_p\ 0]. \end{aligned} \tag{2.6}$$

The occurrence of radiation events induces with a certain probability an upset of the control system. The effect of the upset is to change the control law and introduce a random disturbance signal $\boldsymbol{w}(k)$. The change from nominal to upset operation is referred to as the

interference mapping which is defined as:

$$\mathcal{I} \ : \ \mathcal{S}_1 \mapsto \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times m} \times [0,1]$$

$$j \mapsto (\Delta A_j, \Delta B_j, p_j^u),$$

where ΔA_j and ΔB_j are perturbation matrices that perturb the nominal triplet (A_0, B_0, C_0) with probability p_j^u when $\boldsymbol{\theta}(k) = j \in \mathcal{S}_1$. Since C_0 depends only on the plant parameter C_p, it is not perturbed but the interference. Note that the interference mapping considers that the Markov chain $\boldsymbol{\theta}(k)$ has a finite state space, that is $\boldsymbol{\theta}(k) \in \mathcal{S}_1 = \{0, 1, 2, \ldots, N-1\}$, because the transition probabilities for very high order events are statistically negligible.

The interference mapping can also be seen as the composition of two mappings. The first one maps the discrete time Markov chain $\boldsymbol{\theta}(k)$ into a second one $\{\tilde{\boldsymbol{\theta}}(k) : k \in Z^+\}$ consisting of two states for each state of $\{\boldsymbol{\theta}(k) : k \in Z^+\}$, one for the upset condition and one for the no upset condition. The second mapping assigns to each state of $\tilde{\boldsymbol{\theta}}(k)$ a particular perturbation. The transition probability matrix of the second Markov chain is given by (see [18]):

$$\tilde{\Pi} = \left(\Pi \otimes \begin{bmatrix} 1 & 1 \\ & \\ 1 & 1 \end{bmatrix} \right) diag(1 - p_0^u, p_0^u, 1 - p_1^u, p_1^u, \ldots, 1 - p_N^u, p_N^u). \tag{2.7}$$

With this setup, the closed-loop system is represented by:

$$\begin{aligned} \boldsymbol{x}_{CL}(k+1) &= A_{\tilde{\boldsymbol{\theta}}(k)} \boldsymbol{x}_{CL}(k) + B_{\tilde{\boldsymbol{\theta}}(k)} [\boldsymbol{w}^T(k) \ r^T(k)]^T \\ \boldsymbol{y}_{CL}(k) &= C_0 \boldsymbol{x}_{CL}(k), \end{aligned} \tag{2.8}$$

where for $\tilde{\theta}(k) = \ell,\ \ell \in \mathcal{S}_2 := \{0, 1, \ldots, 2N-1\}$:

$$A_\ell = \begin{cases} A_0 & : \ \ell \text{ even} \\ A_0 + \Delta A_\ell & : \ \ell \text{ odd} \end{cases}$$

$$B_\ell = \begin{cases} B_0 & : \ \ell \text{ even} \\ B_0 + \Delta B_\ell & : \ \ell \text{ odd.} \end{cases}$$

The closed-loop system representation in (2.8) characterizes both the nominal ($\tilde{\theta}$ even) and upset ($\tilde{\theta}$ odd) operations of the aircraft. In particular, the representation in (2.8) can be used to analyze the stability of the aircraft by setting the external inputs to zero. The next section provides the mathematical foundation for analyzing the stability of DTMJLS.

2.3 Stability of Discrete-Time Markovian Jump Linear System

There are several notions of stability used for DTMJLS (see [34] for a good summary). Among these notions, mean square stability (MSS) is highly regarded because of its natural relationship with the energy of the system and because it is useful in the solution of linear quadratic optimization problems. This notion was first discussed by Kushner (1967) and Kozin (1969) (see the references in [28]). The first sufficient conditions and tests for MSS of DTMJLS were developed by Ji and Chizeck [28] and Ji et al. [29].

In 1993 Costa and Fragoso developed two sets of necessary and sufficient conditions for MSS that are easier to verify by direct calculations. What follows is a summary of Costa and Fragoso results (see [8]).

Definition 2.3.1 (Mean Square Stability) *The DTMJLS in (2.8) with $r(k) = 0$ is mean square stable (MSS) if for any initial condition x_0, initial chain distribution v and*

input disturbance $\boldsymbol{w}(k)$, $k \in \boldsymbol{Z}^+$,, *there exist a* $Q \in \mathbb{R}^{n\times n}$ *independent of* $\boldsymbol{x}_0$ *such that:*

$$\|Q(k) - Q\| \to 0 \qquad as \qquad k \to \infty, \tag{2.9}$$

where $Q(k) \triangleq E\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\}$, $\|.\|$ *is any induced matrix norm.*

The concept of MSS is also important because is related to the concept of almost sure stability (ASS), also called stability with probability 1, which is defined next [10].

Definition 2.3.2 (Almost Sure Stability) *The DTMJLS in (2.8) with* $r(k) = 0$ *is almost surely asymptotically stable (ASS) if for any initial condition* $\boldsymbol{x}_0$ *and initial chain distribution* $\boldsymbol{v}$, *the*

$$Prob\{\lim_{k\to\infty} \|\boldsymbol{x}(k)\| = 0\} = 1 \tag{2.10}$$

where $\|\cdot\|$ *is any induced matrix norm.*

The relationship between MSS and ASS is given by the following theorem [19, 29].

Theorem 2.3.1 *If the DTMJLS in equation (2.8) is mean square stable then, for any initial condition* $\boldsymbol{x}_0$ *and initial chain distribution* $\boldsymbol{v}$, *it is almost sure stable.*

This result states that of mean square stability implies almost sure stability, which is a highly regarded performance metric for design engineers. Hence, Theorem 2.3.1 emphasizes the importance of MSS. The next theorem will provide the basic test for MSS. The theorem requires the following assumptions.

Assumption 3 $\bar{\theta}(k)$ *is an aperiodic Markov chain.*

Assumption 4 $\boldsymbol{x}_0$ *is a second-order random variable.*

Assumption 5 $w(k)$ *is a second-order, independent, wide sense stationary sequence of random variables.*

Assumption 6 *x_0 and $\theta(k)$ are independent of $w(k)$ for each $k \in Z^+$.*

Theorem 2.3.2 (Costa-Fragoso, 1993) *The DTMJLS in equation (2.8) is mean-square stable in the sense of Definition 2.3.1 if and only if the spectral radius of*

$$\mathcal{A}_1 := (\tilde{\Pi}^T \otimes I_{n^2}) \cdot diag(A_0 \otimes A_0, \ldots, A_{2N-1} \otimes A_{2N-1}),$$

is strictly less than one, where $\otimes$ denotes the Kronecker matrix product, I_{n^2} is an $n^2 \times n^2$ identity matrix, and $A_0, A_1, \ldots, A_{2N-1}$ denote the values of $A_{\tilde{\theta}(k)}$ for $\tilde{\theta}(k) = 0, 1, \ldots, 2N-1$.

A detailed proof for this theorem is presented in [8]. A very useful simplification arises when all the external inputs are set to zero, thus $w(k) = 0$ and $r(k) = 0$. The following Corollary is the main result of this section.

Corollary 2.3.1 *The jump linear system defined in equation (2.8) with inputs $w(k) = 0$ and $r(k) = 0$, $\forall k \in Z^+$, is mean-square stable in the sense of Definition 2.3.1 if and only if the spectral radius of*

$$\mathcal{A}_1 := (\tilde{\Pi}^T \otimes I_{n^2}) \cdot diag(A_0 \otimes A_0, \ldots, A_{2N-1} \otimes A_{2N-1})$$

is strictly less than one.

The following proof has been adapted from [19] and is simpler than the one in [8].

Proof: First observe the basic decomposition

$$\begin{aligned} Q(k) &:= E\left\{x(k)x^T(k)\right\} \\ &= \sum_{j=0}^{2N-1} E\left\{x(k)x^T(k)\mathbf{1}_{\{\tilde{\theta}(k)=j\}}\right\} \\ &= \sum_{j=0}^{2N-1} Q_j(k), \end{aligned} \tag{2.11}$$

where $\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=j\right\}} = 1$ when $\tilde{\boldsymbol{\theta}}(k) = j$ and 0 otherwise. Note that $1 = \sum_{i=0}^{2N-1} \mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}$. The equation that describes the evolution of the mean square value of each mode is derived next

where $j = 0, 1, \ldots, 2N-1$

$$
\begin{aligned}
Q_j(k+1) &= E\left\{\boldsymbol{x}(k+1)\boldsymbol{x}^T(k+1)\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k+1)=j\right\}}\right\} \\
&= E\left\{\left(A_{\tilde{\boldsymbol{\theta}}(k)}\boldsymbol{x}(k)\boldsymbol{x}^T(k)A^T_{\tilde{\boldsymbol{\theta}}(k)}\right)1\left(\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k+1)=j\right\}}\right)\right\} \\
&= E\left\{\left(A_{\tilde{\boldsymbol{\theta}}(k)}\boldsymbol{x}(k)\boldsymbol{x}^T(k)A^T_{\tilde{\boldsymbol{\theta}}(k)}\right)\left(\sum_{i=0}^{2N-1}\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\right)\left(\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k+1)=j\right\}}\right)\right\} \\
&= E\left\{\left[\sum_{i=0}^{2N-1} A_i\boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\right]\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k+1)=j\right\}}\right\} \\
&= \sum_{i=0}^{2N-1} E\left\{A_i\boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k+1)=j\right\}}\right\} \\
&= \sum_{i=0}^{2N-1} E\left\{A_i\boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}E\{\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k+1)=j\right\}}|\boldsymbol{x}(k),\tilde{\boldsymbol{\theta}}(k)\}\right\}.
\end{aligned}
$$

The last relation can be simplified since all the statistics of $\tilde{\boldsymbol{\theta}}(k+1)$ are completely determined if $\tilde{\boldsymbol{\theta}}(k)$ is given:

$$
\begin{aligned}
Q_j(k+1) &= \sum_{i=0}^{2N-1} E\left\{A_i\boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}E\{\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k+1)=j\right\}}|\tilde{\boldsymbol{\theta}}(k)\}\right\} \\
&= \sum_{i=0}^{2N-1} E\left\{A_i\boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}P\{\tilde{\boldsymbol{\theta}}(k+1)=j|\tilde{\boldsymbol{\theta}}(k)\}\right\} \\
&= \sum_{i=0}^{2N-1} E\left\{A_i\boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\pi_{\tilde{\boldsymbol{\theta}}(k),j}\right\} \\
&= \sum_{i=0}^{2N-1} E\left\{A_i\boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\right\}\pi_{i,j}\ ,
\end{aligned}
$$

where $\pi_{ij} = P\{\tilde{\boldsymbol{\theta}}(k+1) = j|\tilde{\boldsymbol{\theta}}(k) = 1\}$.

Applying the column stacking operator $vec(\cdot)$ to both sides of the last equation and using

the identity $vec(ABC) = (A \otimes C^T)vec(B)$ (see [5]), it follows that:

$$\begin{aligned}
\tilde{q}_j(k+1) &:= vec(Q_j(k+1)) \\
&= vec\left(\sum_{i=0}^{2N-1} E\left\{A_i \boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T \mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\right\}\right)\pi_{i,j} \\
&= \sum_{i=0}^{2N-1} E\left\{vec\left(A_i \boldsymbol{x}(k)\boldsymbol{x}^T(k)A_i^T\right)\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\right\}\pi_{i,j} \\
&= \sum_{i=0}^{2N-1} (A_i \otimes A_i)\, E\left\{vec\left(\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right)\mathbf{1}_{\left\{\tilde{\boldsymbol{\theta}}(k)=i\right\}}\right\}\pi_{i,j} \\
&= \sum_{i=0}^{2N-1} (A_i \otimes A_i)\,\tilde{q}_i(k+1)\pi_{i,j} \qquad (2.12)
\end{aligned}$$

Next, stacking the vectors $\tilde{q}_j(k+1)$ into a column vector results in the following relation:

$$\begin{bmatrix} \tilde{q}_0(k+1) \\ \tilde{q}_1(k+1) \\ \vdots \\ \tilde{q}_{2N-1}(k+1) \end{bmatrix} = \begin{bmatrix} \pi_{0,0}(A_0 \otimes A_0)\tilde{q}_1(k) + \ldots + \pi_{2N-1,0}(A_{2N-1} \otimes A_{2N-1})\tilde{q}_{2N-1}(k) \\ \pi_{0,1}(A_0 \otimes A_0)\tilde{q}_1(k) + \ldots + \pi_{2N-1,1}(A_{2N-1} \otimes A_{2N-1})\tilde{q}_{2N-1}(k) \\ \vdots \\ \pi_{0,2N-1}(A_0 \otimes A_0)\tilde{q}_1(k) + \ldots + \pi_{2N-1,2N-1}(A_{2N-1} \otimes A_{2N-1})\tilde{q}_{2N-1}(k) \end{bmatrix}$$

Defining $\tilde{q}(k+1) = \left[\tilde{q}_0^T(k+1), \ldots, \tilde{q}_{2N-1}^T(k+1)\right]^T$ the last relation can be restated as:

$$\tilde{q}(k+1) = \begin{bmatrix} \pi_{0,0}(A_0 \otimes A_0) & \pi_{1,0}(A_1 \otimes A_1) & \ldots & \pi_{2N-1,0}(A_{2N-1} \otimes A_{2N-1}) \\ \pi_{0,1}(A_0 \otimes A_0) & \pi_{1,1}(A_1 \otimes A_1) & \ldots & \pi_{2N-1,1}(A_{2N-1} \otimes A_{2N-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{0,2N-1}(A_0 \otimes A_0) & \pi_{1,2N-1}(A_1 \otimes A_1) & \ldots & \pi_{2N-1,2N-1}(A_{2N-1} \otimes A_{2N-1}) \end{bmatrix} \tilde{q}(k)$$

$$\tilde{q}(k+1) = \begin{bmatrix} \pi_{0,0}I_{n^2} & \pi_{1,0}I_{n^2} & \ldots & \pi_{2N-1,0}I_{n^2} \\ \pi_{0,1}I_{n^2} & \pi_{1,1}I_{n^2} & \ldots & \pi_{2N-1,1}I_{n^2} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{0,2N-1}I_{n^2} & \pi_{1,2N-1}I_{n^2} & \ldots & \pi_{2N-1,2N-1}I_{n^2} \end{bmatrix} diag(A_0 \otimes A_0, \ldots, A_{2N-1} \otimes A_{2N-1})\tilde{q}(k)$$

$$\begin{aligned}\tilde{q}(k+1) &:= (\tilde{\Pi}^T \otimes I_{n^2}) \cdot diag(A_0 \otimes A_0, \ldots, A_{2N-1} \otimes A_{2N-1})\tilde{q}(k) \\ &= \mathcal{A}_1\, \tilde{q}(k). \end{aligned} \tag{2.13}$$

Equation 2.13 corresponds to a linear, homogeneous discrete-time system. The system is asymptotically stable if and only if $\rho(\mathcal{A}_1) < 1$. The latter condition is equivalent to

$$\lim_{k\to\infty} \tilde{q}(k) = 0,$$

Note that:

$$\begin{aligned}\lim_{k\to\infty} Q(k) &= \lim_{k\to\infty} \sum_{j=0}^{2N-1} vec^{-1}(\tilde{q}_j(k)) \\ &= vec^{-1}\left(\sum_{j=0}^{2N-1} \lim_{k\to\infty} \tilde{q}_j(k)\right),\end{aligned}$$

and because $\tilde{q}_j(k)$ is the jth column of $vec^{-1}(\tilde{q}(k))$, it follows that

$$\lim_{k\to\infty} Q(k) = \lim_{k\to\infty} \tilde{q}(k) = 0.$$

Hence, the system is mean square stable if and only if $\rho(\mathcal{A}_1) < 1$. ∎

2.3.1 Alternative Statistical Computation

The theoretical stability test proposed in Corollary 2.3.1, $\rho(\mathcal{A}_1) < 1$, can be verified via Monte Carlo simulations. This is done by choosing a suitable distribution for the initial state $\boldsymbol{x}_0$ and then using Monte Carlo simulations to estimate the mean square value of the states, $Q(k)$. If the system is theoretically stable and the estimates vanish as k gets sufficiently large, then the simulations verify the theoretical result. On the other hand, if the system is theoretically unstable and the estimates grow unbounded as k gets large, then the simulation also verifies the theoretical prediction.

If the theoretical predictions are to be verified by direct computation using the conditions given in Definition 2.3.1, the amount of required storage space would be rather massive. The reason is that in order to compute $\|Q(k)\| = \|E\{\boldsymbol{x}(k)\boldsymbol{x}(k)^T\}\|$, the matrices $\boldsymbol{x}(k)\boldsymbol{x}(k)^T$ must be stored for each sample instant of the simulation (see Chapter 4 for simulation details). As an example, consider the case in which the size of $\boldsymbol{x}(k)$ is 12 and the sample period $T = 1/250$ sec. A common theoretical verification might require 500 Monte Carlo runs, each one of length 1000 sec. Hence the amount of data that should be stored, assuming 8 bytes for each double precision number (Matlab's standard), requires $12^2*250*1000*8$ bytes $\approx$ 275Mbytes (considering that the data is accumulated at each sample instant). This problem gets worse for larger and more complex systems.

An easier statistic to compute, which is equivalent to the mean square stability condition, is given by the following theorem.

Theorem 2.3.3 *The jump linear system defined in (2.8) with inputs* $\boldsymbol{w}(k) = 0$ *and* $r(k) = 0$, $\forall k \in \boldsymbol{Z}^+$ *is mean-square stable if and only if*

$$\hat{Q}(k) \to 0 \text{ as } k \to \infty, \tag{2.14}$$

where $\hat{Q}(k) := E\{\|\boldsymbol{x}(k)\boldsymbol{x}^T(k)\|\}$.

Proof: It will be shown that

$$\hat{Q}(k) \to 0 \text{ as } k \to \infty$$

if and only if

$$\|Q(k)\| \to 0 \text{ as } k \to \infty.$$

The last statement is equivalent to mean square stability of the system defined in Definition 2.3.1.

First a simplified relation for $\hat{Q}(k)$ is derived for each $k \in \boldsymbol{Z}^+$:

$$\begin{aligned} \|\boldsymbol{x}(k)\boldsymbol{x}^T(k)\| &= \sqrt{\lambda_{max}([\boldsymbol{x}(k)\boldsymbol{x}^T(k)]^T[\boldsymbol{x}(k)\boldsymbol{x}^T(k)])} \\ &= \sqrt{\lambda_{max}(\boldsymbol{x}(k)[\boldsymbol{x}^T(k)\boldsymbol{x}(k)]\boldsymbol{x}^T(k))} \\ &= \sqrt{\boldsymbol{x}^T(k)\boldsymbol{x}(k)}\sqrt{\lambda_{max}(\boldsymbol{x}(k)\boldsymbol{x}^T(k))} \end{aligned}$$

Now $\boldsymbol{x}(k)\boldsymbol{x}^T(k)$ is a rank 1 matrix and its only nonzero eigenvalue, if $\boldsymbol{x}(k) \neq 0$, is given by $\boldsymbol{x}^T(k)\boldsymbol{x}(k) = \sum_{j=1}^{n} \boldsymbol{x}_i(k)^2 = \|\boldsymbol{x}(k)\|^2 > 0$. Thus

$$\|\boldsymbol{x}(k)\boldsymbol{x}^T(k)\| = \boldsymbol{x}^T(k)\boldsymbol{x}(k) = \|\boldsymbol{x}(k)\|^2,$$

and the desired simplified relation is given by

$$\hat{Q}(k) = E\left\{\|\boldsymbol{x}(k)\|^2\right\} \geq 0.$$

Now, consider the trace of $Q(k)$

$$\begin{aligned} Tr(Q(k)) &= Tr(E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\}) \\ &= \sum_{j=1}^{n} E\{(\boldsymbol{x}_j(k))^2\} \\ &= E\left\{\sum_{j=1}^{n}(\boldsymbol{x}_j(k))^2\right\} \\ &= E\{\|\boldsymbol{x}(k)\|^2\} \\ &= \hat{Q}(k) \end{aligned} \tag{2.15}$$

Since $\hat{Q}(k) = Tr(Q(k))$, then $\hat{Q}(k) \to 0$ as $k \to \infty$ if and only if $Tr(Q(k)) \to 0$ as $k \to \infty$.

Now the main part of the proof.

Suppose that $\|Q(k)\| \rightarrow 0$ as $k \rightarrow \infty$, then

$$\begin{aligned}
\|Q(k)\| &= \|E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\}\| \\
&= \sqrt{\lambda_{max}([E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\}]^T E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\})} \\
&= \sqrt{\lambda_{max}(E\left\{[\boldsymbol{x}(k)\boldsymbol{x}^T(k)]^T\right\} E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\})} \\
&= \sqrt{\lambda_{max}(E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\}^2)} \\
&= \sqrt{\lambda^2_{max}(E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\})} \\
&= \lambda_{max}(E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\}),
\end{aligned}$$

and therefore $\lambda_{max}(E\left\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right\}) \rightarrow 0$ as $k \rightarrow \infty$. But because $Q(k)$ is positive semi-definite (see Theorem 7.1 in [37]), then $Tr(Q(k)) = \hat{Q}(k) \rightarrow 0$ as $k \rightarrow \infty$.

Next, suppose that $Q(k) \rightarrow 0$ as $k \rightarrow \infty$. Because $Q(k)$ is positive semi-definite, then $\lambda_{max}(Q(k)) \rightarrow 0$ as $k \rightarrow \infty$. But $\|Q(k)\| = \lambda_{max}(Q^2(k))$ by the previous discussion and therefore, $\|Q(k)\| \rightarrow 0$ as $k \rightarrow \infty$ when $Q(k) \rightarrow 0$ as $k \rightarrow \infty$. ∎

Theorem 2.3.3 states that in order to analyze the stability of the jump linear system in (2.8) it is enough to analyze the statistics of the norm of the system's state vector. Hence, under the same conditions, the simulation of the example introduced at the beginning of this section using the new statistic will require only 2 Mbytes of computational space.

2.3.1.1 Asymptotic Behavior of $\hat{Q}(k)$

The following lemma gives a useful asymptotic approximation for $\hat{Q}(k)$ as $k \rightarrow \infty$.

Lemma 2.3.1 *Given the jump linear system defined in (2.8) with inputs $\boldsymbol{w}(k)$ and $r(k)$ set to zero, $\forall k \in \boldsymbol{Z}^+$, and if the real or complex conjugate pair of eigenvalues of $\mathcal{A}_1$ with*

maximum magnitude have multiplicity 1, it follows that:

$$\lim_{k\to\infty} \frac{\log \hat{Q}(k)}{k} \simeq \log(\rho(\mathcal{A}_1)) \tag{2.16}$$

Proof: The proof will first derive an expression for $\hat{Q}(k)$ in terms of $(\mathcal{A}_1)^k$. Then, a modal expansion of $(\mathcal{A}_1)^k$ under the lemma's assumptions will be used to derive the desired results. From (2.15) and the definition of $Q(k)$ in (2.11):

$$\begin{aligned} \hat{Q}(k) &= Tr(Q(k)) \\ &= Tr\left(\sum_{j=0}^{2N-1} Q_j(k)\right) \\ &= \sum_{j=0}^{2N-1} Tr(Q_j(k)). \end{aligned}$$

Then using the following property of the $vec(\cdot)$ operator and the trace of a matrix

$$Tr(Q_j(k)) = Tr(I_n Q_j(k)) = vec(I_n)^T vec(Q_j(k)),$$

it follows that

$$\hat{Q}(k) = vec(I_n)^T \sum_{j=0}^{2N-1} vec(Q_j(k)).$$

Recall from equation (2.12) that $vec(Q_j(k)) = \tilde{q}_j(k)$, so the previous expression for $\hat{Q}(k)$ can be formulated as:

$$\begin{aligned} \hat{Q}(k) &= vec(I_n)' \sum_{j=0}^{l-1} \tilde{q}_j(k) \\ &= vec(I_n)' \left[I_{n^2}, \ldots, I_{n^2}\right] \tilde{q}(k) \\ &= \mathcal{N}\tilde{q}(k). \end{aligned} \tag{2.17}$$

To continue the derivation, solve the difference equation in (2.13) which gives

$$\tilde{q}(k) = (\mathcal{A}_1)^k \tilde{q}(0), \tag{2.18}$$

where $\tilde{q}(0)$ is the vector of initial conditions. Substituting (2.18) in (2.17) gives

$$\hat{Q}(k) = \mathcal{N}\,(\mathcal{A}_1)^k\,\tilde{q}(0). \tag{2.19}$$

If a single real pole has the maximum magnitude, it then corresponds to the spectral radius of $\mathcal{A}_1$ and the directed modal expansion of $(\mathcal{A}_1)^k$ can be written as follows

$$\mathcal{A}_1^k = R(\rho(\mathcal{A}_1))^k + \varphi(k) \tag{2.20}$$

where R is a constant residue matrix and $\varphi(k)$ denotes the remaining terms in the directed modal expansion of $(\mathcal{A}_1)^k$.

If the spectral radius of $\mathcal{A}_1$ corresponds to a complex conjugate pair, then the first term in (2.20) is the envelope of the slowest mode of $(\mathcal{A}_1)^k$ [2]. So (2.20) would be written instead as

$$\mathcal{A}_1^k \leq R(\rho(\mathcal{A}_1))^k + \varphi(k). \tag{2.21}$$

In (2.20) and (2.21) the right hand side are both dominated by the first term. Substituting (2.20) in (2.19) gives

$$\begin{aligned}\hat{Q}(k) &= \mathcal{N}\,(R(\rho(\mathcal{A}_1))^k + \varphi(k))\,\tilde{q}(0)\\ &= \mathcal{N}R\tilde{q}(0)(\rho(\mathcal{A}_1))^k + \mathcal{N}\varphi(k)\tilde{q}(0).\end{aligned}$$

Taking logarithms on both sides gives

$$\log\hat{Q}(k) = \log(\mathcal{N}R\tilde{q}(0)(\rho(\mathcal{A}_1))^k + \mathcal{N}\varphi(k)\tilde{q}(0)).$$

Now as $k \to \infty$

$$\begin{aligned}\log\hat{Q}(k) &\to \log(\mathcal{N}R\tilde{q}(0)(\rho(\mathcal{A}_1))^k)\\ &\to \log(\mathcal{N}R\tilde{q}(0)) + k\log((\rho(\mathcal{A}_1))^k)\end{aligned}$$

Dividing by k and taking the limit as $k \rightarrow \infty$ gives the desired result. ■

Thus, logarithmic plots of the estimates of $\hat{Q}(k)$ obtained via Monte Carlo simulations approach a slope of $\log \rho(\mathcal{A}_1)$ for sufficiently large k under the conditions of the lemma. The slope criteria will be used as a verification of the quality of the simulations in Chapter 4.

From the definition of the second Lyapunov exponent [6]

$$\lambda_2(\boldsymbol{x}_0) = \lim_{k\to\infty} \frac{\log(\|\boldsymbol{x}(k, \boldsymbol{x}_0)\|^2)}{k},$$

that is, $\lambda_2(\boldsymbol{x}_0) = \lim_{k\to\infty} \frac{\log \hat{Q}(k)}{k}$. Therefore, it follows from equation (2.16) that $\log \rho(\mathcal{A}_1)$ is, at least, an approximation to the second Lyapunov exponent as $k \rightarrow \infty$. So it might be possible to study MSS of discrete-time jump linear systems with the second Lyapunov exponent stability analysis [6] .

2.3.2 The Rare Event Case

This section analyzes Corollary 2.3.1 when the disturbances occur rarely. Since the arrivals of the disturbance events is characterized by a Poisson process and their duration by an exponential distribution, a rare event means that the average duration of a disturbance, $1/\mu$, is much less than the average interarrival separation of the disturbances, $1/\lambda$. Thus, it follows that $\lambda \ll \mu$ and $p_k \approx 0$ for $k > 1$.

With this assumption, the finite state Markov chain $\tilde{\boldsymbol{\theta}}(k)$ has only 4 states with the corresponding transition probability matrix:

$$\tilde{\Pi} = \left(\Pi \otimes \begin{bmatrix} 1 & 1 \\ & \\ 1 & 1 \end{bmatrix} \right) diag(1 - p_0^u, p_0^u, 1 - p_1^u, p_1^u). \tag{2.22}$$

The following assumptions are also made: An upset never occurs when there is no radiation, because the system is assumed to be free of internal random faults, and an

upset always occurs when radiation is present. Hence $p_0^u = 0$ and $p_1^u = 1$. The following modification of Corollary 2.3.1 will be the main analysis tool used in the rest of this thesis.

Corollary 2.3.2 (Rare Event Assumption) *Under rare event conditions, the jump linear system defined Corollary 2.3.1 with $r(t) \equiv 0$ is mean square stable if and only if the spectral radius of*

$$\mathcal{A}_{r1} = \begin{bmatrix} e_{11}(A_0 \otimes A_0) & e_{21}(A_3 \otimes A_3) \\ e_{12}(A_0 \otimes A_0) & e_{22}(A_3 \otimes A_3) \end{bmatrix} \tag{2.23}$$

is strictly less than 1 with

$$e^{\Lambda T} = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix}.$$

Proof: It is sufficient to prove that $\rho(\mathcal{A}_1) = \rho(\mathcal{A}_{r1})$. From Corollary 2.3.1

$$\mathcal{A}_1 := (\tilde{\Pi}^T \otimes I_{n^2}) \cdot diag(A_0 \otimes A_0, \ldots, A_3 \otimes A_3)$$

Note that under the rare event assumption

$$\begin{aligned} \tilde{\Pi} &= \left(\Pi \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right) diag(1,0,0,1) \\ &= \begin{bmatrix} e_{11} & 0 & 0 & e_{12} \\ e_{11} & 0 & 0 & e_{12} \\ e_{21} & 0 & 0 & e_{22} \\ e_{21} & 0 & 0 & e_{22} \end{bmatrix}. \end{aligned} \tag{2.24}$$

Hence $\mathcal{A}_1$ can be rewritten as

$$\mathcal{A}_1 = \begin{bmatrix} e_{11}(A_0 \otimes A_0) & e_{11}(A_0 \otimes A_0) & e_{21}(A_3 \otimes A_3) & e_{21}(A_3 \otimes A_3) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ e_{12}(A_0 \otimes A_0) & e_{12}(A_0 \otimes A_0) & e_{221}(A_3 \otimes A_3) & e_{22}(A_3 \otimes A_3) \end{bmatrix}.$$

By definition, the eigenvalues of $\mathcal{A}_1$ can be calculated solving the following equation:

$$|zI_{4m} - \mathcal{A}_1| = 0. \tag{2.25}$$

Where I_{4m} is an identity matrix of size $4m \times 4m$, $m = 16n^2$ and n is the size of the plant's state vector. Hence equation (2.25) can be restated as follows:

$$\left\| \begin{bmatrix} zI_m - e_{11}(A_0 \otimes A_0) & -e_{11}(A_0 \otimes A_0) & -e_{21}(A_0 \otimes A_0) & -e_{21}(A_3 \otimes A_3) \\ 0 & zI_m & 0 & 0 \\ 0 & 0 & zI_m & 0 \\ -e_{12}(A_0 \otimes A_0) & -e_{12}(A_0 \otimes A_0) & -e_{22}(A_0 \otimes A_0) & zI_m - e_{22}(A_3 \otimes A_3) \end{bmatrix} \right\| = 0.$$

Using the properties of the determinants over the same equation could be transformed into

$$(-1)^m \left\| \begin{bmatrix} zI_m - e_{11}(A_0 \otimes A_0) & -e_{11}(A_0 \otimes A_0) & -e_{21}(A_0 \otimes A_0) & -e_{21}(A_3 \otimes A_3) \\ -e_{12}(A_0 \otimes A_0) & -e_{12}(A_0 \otimes A_0) & -e_{22}(A_0 \otimes A_0) & zI_m - e_{22}(A_3 \otimes A_3) \\ 0 & 0 & zI_m & 0 \\ 0 & zI_m & 0 & 0 \end{bmatrix} \right\| = 0.$$

Applying the same procedure, equation (2.25) can be rearranged as

$$(-1)^{2m} \left\| \begin{bmatrix} zI_m - e_{11}(A_0 \otimes A_0) & -e_{21}(A_3 \otimes A_3) & -e_{21}(A_0 \otimes A_0) & -e_{11}(A_0 \otimes A_0) \\ -e_{12}(A_0 \otimes A_0) & zI_m - e_{22}(A_3 \otimes A_3) & -e_{22}(A_0 \otimes A_0) & -e_{12}(A_0 \otimes A_0) \\ 0 & 0 & zI_m & 0 \\ 0 & 0 & 0 & zI_m \end{bmatrix} \right\| = 0.$$

Applying the properties of block matrices to equation (2.25) will yield:

$$\left\| \begin{bmatrix} zI_m - e_{11}(A_0 \otimes A_0) & -e_{21}(A_3 \otimes A_3) \\ -e_{12}(A_0 \otimes A_0) & zI_m - e_{22}(A_3 \otimes A_3) \end{bmatrix} \begin{bmatrix} zI_m & 0 \\ 0 & zI_m \end{bmatrix} \right\| = 0.$$

or

$$|zI_{2m} - \mathcal{A}_{r1}|\,|zI_{2m} - 0_{2m}| = 0$$

Because $\rho(\mathcal{A}_1) \geq 0$ then

$$\rho(\mathcal{A}_1) = max(abs(|zI_{2m} - \mathcal{A}_{r1}|))$$

or $\rho(\mathcal{A}_1) = \rho(\mathcal{A}_{r1})$. This completes the proof. ■

CHAPTER III

ON THE STABILITY ANALYSIS OF DIGITAL LINEAR FLIGHT CONTROLLERS WITH EMBEDDED RECOVERY CAPABILITIES

3.1 Introduction

As discussed in Chapter 2, high intensity EM radiation can induce transient non-destructive "soft" faults in the controller of an aircraft. These faults can generate calculation errors that, if allowed to propagate, can produce significant performance degradation of the control system and jeopardize the aircraft's integrity (see [25]). Of course, a single or a few errors in the control signal calculation could be filtered by the dynamics of the aircraft. The problems described here occur when the errors appear at a certain minimum frequency.

There exist several techniques available to detect, mask or correct transient faults that provide different degrees of protection against EM induced faults. Some examples include: improved cable shielding, copper wire substituted with fiber optics, memory/registers checksum or parity detection, modular redundancy (specially triple modular redundancy), reconfiguration of standby spares in some VLSI chips, Byzantine Resilience, watchdog timers, exception traps in memory management units, software checks, etc. [30, 32, 38]. Nevertheless, most of these methods are known to be ineffective against common-mode faults and therefore some faults will produce errors in the system (memory data corruption, bit flips in the processors internal registers, input/output pins temporarily hung up to logic "1" or "0", etc). One method to handle common-mode faults consist of replacing the data in the system by an error free "copy". This "copy", called a checkpoint, is regularly stored in protected memory by a special set of circuits or by a special software routine. The procedure

by which the checkpoint data is re-loaded in the system depends on the recovery algorithm embedded in the system such as rollback (the most standard) or roll-forward recovery.

An example of a commercial fault-tolerant aircraft controller that can overcome common-mode faults is the novel Recoverable Computer System (RCS) developed by Honeywell Inc. A prototype of the RCS is currently being tested in the SAFETI Laboratory at NASA Langley Research Center. This prototype has the option of three recovery methods: rollback, reset and cold-restart [4, 26, 27, 35]. The following sections will develop models and closed-loop stability analysis that include the effect of the RCS on the flight control dynamics. The analysis tools developed here build on those introduced in Chapter 2 (see Corollary 2.3.1). In particular, a new interference mapping for the exosystem model is developed in Section 2.2.2 and the interpretation of some of the exosystem parameters is changed.

This chapter is organized as follows. Section 3.2 presents a detailed explanation of the recovery mechanisms and their corresponding interference mappings. These ideas are linked in Section 3.3 with the stability test introduced in Corollary 2.3.2 to produce the main result of this chapter: the stability analysis tool for digital control systems with fault-tolerant capabilities. Finally, the last section summarizes the main contributions developed in this chapter.

3.2 Interference Models for Recovery Systems

3.2.1 Preliminaries

A hypothesis for this work is that the behavior of a flight control systems with recovery mechanisms subject to EM disturbances can be modeled using the tools introduced in Chapter 2. This hypothesis was developed after observation and analysis of experiments conducted in the SAFETI Laboratory. In these experiments, a digital flight controller

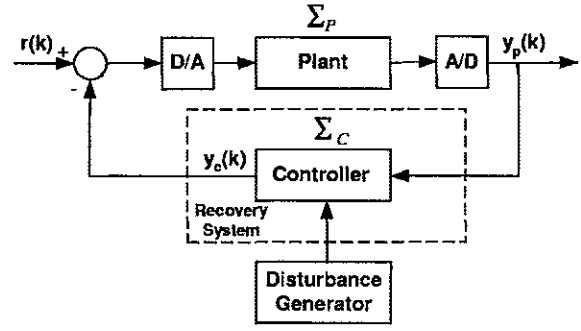

Figure 3.1: Block diagram of sampled-data flight control system with error recovery subject to electromagnetically induced upsets.

programmed with an automatic landing maneuver was inserted into a special chamber where it could be bombarded with high intensity radiated fields (HIRF) similar to those that the aircraft could experience. The controller was appropriately interfaced to a computer with a high fidelity model of a Boeing 737. The latter computer was placed outside of the chamber so that only the digital controller could be affected by HIRF. During some of these experiments the aircraft's trajectory deviated from the nominal one. Some of these deviations could have led to catastrophic consequences if the pilots had not taken over the controls. These are the upsets that are being considered in this thesis. The rest of this section sets up the notation for the nominal aircraft and radiation duration that will allow the development of models for analysis.

Consider the block diagram in Figure 3.1 that shows a closed-loop sampled-data aircraft control system whose controller is implemented on an RCS device. The RCS system is a dual-processor fault-tolerant system similar to the duplex system depicted in Figure 3.2 ([21, 26] explain the known characteristics of the RCS implementation). In this duplex

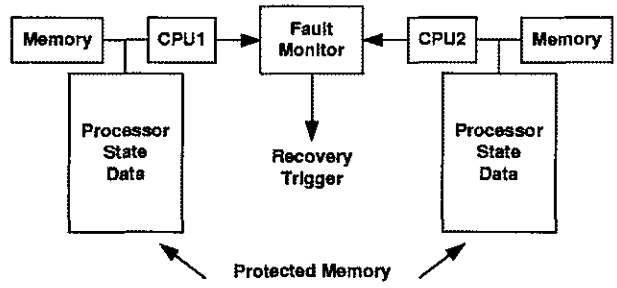

Figure 3.2: Architecture of duplex fault-tolerant system.

system, each processor module consists of a CPU with its own memory as well as protected memory for storing the module's state data, which consists of the processor's registers and selected memory locations. The fault monitor is a fault-tolerant comparator that detects faults by comparing the state of each processor module at predetermined time intervals. Once a fault is detected, the recovery process consists of halting normal operation, reloading each processor module's state data with error-free information and then returning back to normal operation. During the reload operation, the control signal values are kept frozen. The data used to reload the processor modules depend on the recovery method.

Recall from Chapter 2 that the plant's closed-loop representation model is given by

$$x_{CL}(k+1) = \begin{bmatrix} x_p(k+1) \\ x_c(k+1) \end{bmatrix} = A_0 \begin{bmatrix} x_p(k) \\ x_c(k) \end{bmatrix} + B_0[w^T(k)\ r^T(k)]^T \qquad (3.1)$$

$$y_{CL}(k) = C_0 \begin{bmatrix} x_p(k) \\ x_c(k) \end{bmatrix},$$

where $A_0 = \begin{bmatrix} A_p & -B_P C_C \\ B_c C_p & A_c \end{bmatrix}$, $B_0 = \begin{bmatrix} 0 & B_P \\ 0 & 0 \end{bmatrix}$ and $C_0 = [C_p\ 0]$.

Assume that the disturbance generator in Figure 3.1 can be characterized in the manner presented in Section 2.2.2. Then it follows that when there is no recovery mechanism available, the effect of the EM induced upsets is represented by random changes that describe the modifications in the digital controller's output. The interference produced by these changes can be modeled by random perturbations of the nominal closed-loop state feedback matrices A_0, B_0 and C_0.

On the other hand, when there is a recovery mechanism available, it prevents the propagation of the errors, induced by the disturbances, through the system. This is done by introducing structural modifications on the controller's state equation during a period called the recovery window. These modifications depend on the recovery technique implemented (rollback, reset or cold-restart) and affect the closed-loop system behavior in a deterministic way.

In particular, under the rare event conditions of Section 2.3.2, the Markov chain that represents the exosystem has only two states (radiation not present and radiation present), which are easily related to the two controller operating behaviors: nominal behavior and recovery behavior. Then, the interference introduced by the disturbances can be represented by a mapping similar to the one presented in section 2.2.3.

Consider now the exosystem model introduced in Section 2.2.2. Figure 3.3 shows the timeline of the events following the arrival of an EM disturbance. As stated in Section 2.2.2, the i^{th} EM fault arrives at time $\boldsymbol{t}_i$ and lasts $\boldsymbol{d}_i$ seconds. If the recovery mechanism were not present, the system's closed-loop representation would be randomly switched only during the time the fault is present ($\boldsymbol{d}_i$ seconds) and returned to its nominal value (A_0, B_0, C_0), after the fault is removed.

When the recovery mechanism is present, the closed-loop representation is switched dur-

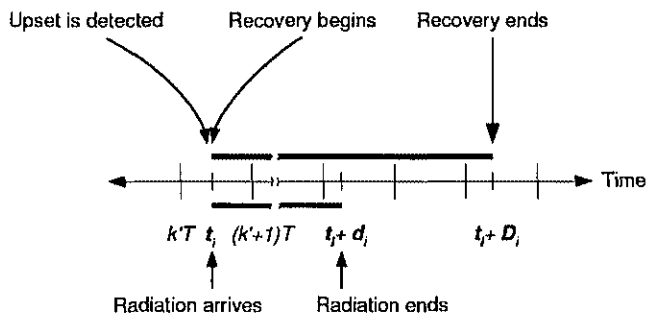

Figure 3.3: Timeline of the events following an EM fault arrival.

ing the time the *recovery window* is active. The recovery window depends on the hardware but it is, in general, a random variable with unknown distribution. For convenience, the following assumption is made.

Assumption 7 *Let $\boldsymbol{D_i}$ denote the recovery duration. $\boldsymbol{D_i}$ is a exponentially distributed random variable with expected value $1/\mu_D$ such that $1/\mu_D \gg 1/\mu$, where and $1/\mu$ is the expected value of the radiation duration, $\boldsymbol{d_i}$.*

This assumption ensures that the Markovian model devised on Section 2.2.2 will correctly model the interaction between the external disturbances and the recovery system.
The characteristics of the three recovery methods will now be presented.

3.2.2 Rollback Recovery Method

This recovery method is highly regarded because it provides an effective mechanism to restore the system from a faulty condition. Typically, this is done only with a small degradation of performance. When successfully applied, this method can restore the system

to (almost) the same operating point before the fault. On the other hand, rollback recovery requires more complex and costly hardware (like radiation hardened memories and other fault-tolerant components) and due to the finite nature of the the recovery time, it cannot handle successive faults if their arrivals are not separated a minimum time (called the minimum interarrival spacing), rendering the system unstable in that case.

The Minimum Interarrival Spacing (MIS) is a function of parameters such as the checkpointing frequency, the amount of data stored in each checkpoint, the fault-tolerant comparison frequency, the recovery duration, etc. and can be estimated using the theory presented shortly (also see Chapter 4 for case study examples).

The rollback recovery method requires the following steps: checkpointing, fault-tolerant comparison, reload and retry. This steps are explained below.

Checkpointing In general, checkpointing is the process by which all the processor's data is latched in special memory and then transferred to longer-term protected memory. The specific implementation characteristics vary greatly with the application. For example, in micro-rollback [39,40] checkpointing is executed by hardware during each of the processor's instruction cycles with the use of replicated registers and memory cells. In other cases, checkpointing is done by a special software routines [7,13] and lasts several (controller) sample periods depending on the amount of data being stored. This work assumes that the data is periodically stored every sampling instant until an upset triggers the rollback recovery process. Note that some recovery systems implement a dynamical checkpointing schedule (e.g. see [7,38]).

A more detailed timeline of the checkpointing process is given in Figure 3.4. In this figure $(L + \triangle)T$ represents the duration of a complete checkpointing process, where L is a non-negative integer and $\triangle$ is a real number such that $\triangle \in (0, 1)$. The

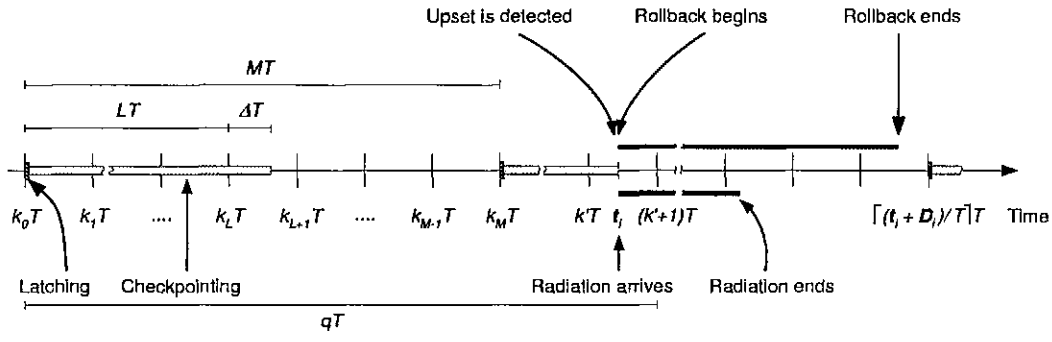

Figure 3.4: Timeline of the checkpointing and rollback recovery events.

checkpointing process period is given by MT where M satisfies $M \geq L+1$. From the recovery mechanism point of view, the rollback recovery process starts immediately after the upset is detected at time t_i and lasts D_i seconds. From the control system's perspective, the recovery process starts at time $(k'+1)T = \lceil t_i/T \rceil T$ and ends when the data is ready to be released for processing at time $k_rT = \lceil (t_i + d_i)/T \rceil T$, where the operator $\lceil x \rceil$ denotes smallest integer greater than x.

Fault-Tolerant Comparisons These comparisons are used to detect faults inside the processor modules that implement the control law. The faults can be detected by hardware [39, 40] (using checksums or parity detectors), by a combination of software and hardware [7] or only by software like in message passing systems [36]. In the RCS these checks are performed frequently throughout the sample period and in parallel with the checkpointing process. It is assumed that the comparisons can detect the i^{th} upset shortly after it happens at time t_i where $t_i \in [k'T, (k'+1)T)$ and $i = 0, 1, 2, ...$

Reload and Retry During a reload, the rollback process performs the following operations.

- Stop the fault-tolerant comparisons and the checkpointing process, if active.
- Freeze the controller's output to the last known-good value

$$y_c(k) = y_c(k'), \tag{3.2}$$

for $k \in [(k'+1), k_r]$, where $k_r = \lceil (t_i + d_i)/T \rceil$.

- Reload the controller's state vector $x_c(k)$ with the data from the most recently completed checkpoint q sample periods ago. The reloaded state vector is released and made available for processing at the end of the recovery process:

$$x_c(k_r) = x_c(k'+1-q), \tag{3.3}$$

where $x_c(k'+1-q)$ is the controller's state vector q sample periods before the upset took place. When the rollback process ends, the controller returns to normal operation.

The effectiveness of the rollback recovery process depends mainly on two parameters. The value of q in (3.3) and the amount of data stored in each checkpoint. The former parameter is of greatest importance for closed-loop system analysis. For example, in Figure 3.4 the value of q corresponding to the upset at time t_i is $q = k' + 1 - k_0$. In practice, computing q is not an issue, since the memory pointer to the last completed checkpoint would be used. Nevertheless for analysis purposes, there is no simple relation for q since its value depends on the duration of the checkpointing process and on the random time when the upset occurs. However, the minimum and maximum values of q for different sets of rollback parameters can be easily computed. The maximum value of q can then be used for worst-case analysis

Table 3.1: Set of possible q values for several checkpointing parameters.

L	M	q
0 [1]	1	{1}
0 [1]	2	{1,2}
0	1	{1,2}
0	2	{1,2,3}
1	2	{2,3,4}
1	3	{2,3,4,5}

of the specific rollback configuration. Let j be congruent to k' modulo M, that is, $k' - j$ is divisible by M. This congruence relation is denoted by $j \equiv k'(mod\ M)$. Assuming that the i^{th} upset occurs at time t_i then:

$$q = \begin{cases} j+1 & : (L+\Delta)T < t_i - (k'-j)T \\ j+M+1 & : otherwise. \end{cases} \tag{3.4}$$

Table 3.1 shows the possible values of q for different combinations of L and M. The first two rows in Table 3.1 assume that checkpointing is instantaneous ($L = 0$, $\Delta \rightarrow 0$), which is the case considered in [21–23]. When the checkpointing is not instantaneous, for any combination of L, Δ and M, the smallest value of q corresponds to the case in which the upset arrives just after a checkpointing has been completed, so $q_{min} = L + 1$. Similarly, it can be shown that $q_{max} = M + L + 1$.

The second parameter, the amount of information stored in each checkpoint, will be addressed after the closed-loop model for the rollback recovery method is presented in the next section.

[1] Assume that checkpointing is instantaneous.

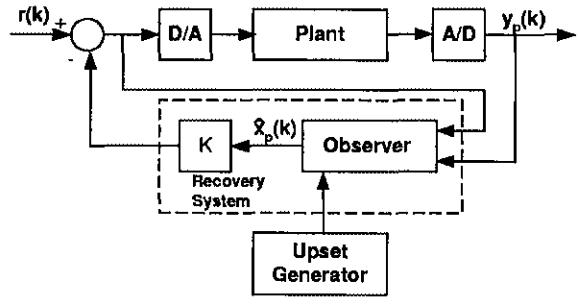

Figure 3.5: Proposed observer-based controller.

3.2.2.1 Rollback Interference Model

The effect of rollback recovery on the closed-loop flight control system can be represented by perturbing the nominal closed-loop system representation in (3.1). If instead of using the general state-space representation of the controller in (2.4), an observer-based representation of the controller is used, the nominal closed-loop matrices in (3.1) are given by:

$$A_0 = \begin{bmatrix} A_p & -B_pK \\ LC_p & A_p - B_pK - LC_p \end{bmatrix}, \; B_0 = \begin{bmatrix} B_p \\ 0 \end{bmatrix}, \; C_0 = \begin{bmatrix} C_p & 0 \end{bmatrix}, \tag{3.5}$$

where K and L are the state feedback and output injection matrices respectively. In the observer-based controller $C_c = I$ and $x_c(k) = \hat{x}_p(k)$ (see (2.4)), where $\hat{x}_p$ are the estimates of the aircraft's state vector. Figure 3.5 shows a conceptual block-diagram of the proposed controller.

Assume the i^{th} disturbance takes place at time t_i and it induces and upset. At the previous sample instant $k'T$, the controller releases (to the analog inputs of the plant) the output $y_c(k')$ and calculates the next state of the controller $\hat{x}_p(k'+1)$ and the next control input $y_c(k'+1)$ using the nominal closed-loop state feedback matrix A_0. At t_i,

the controller's data gets corrupted and the recovery mechanism is triggered. The recovery mechanism immediately freezes the analog control inputs and starts the reload operation. During the reload operation the aircraft's state vector, $x_p(k)$, continues to get updated with the frozen control input. At the end of the reload operations, the controller's state vector is restored to its value from the last completed checkpoint. This is assumed to be q sample instants ago. The value of the controller's state vector during the reload is not important, since it does not affect the aircraft. Because of the discrete nature of the controller, the changes introduced by the recovery mechanism cannot take place before the sample instant $(k'+1)T$.

For analysis purposes, it is necessary to augment the nominal closed-loop dynamics in (3.1) with state vectors that store the value of the controller's state vector from previous sample instants. This makes it possible to roll back the controller to a previous state. Thus, the state vector $\hat{\hat{x}}_{r1}(k) \in \mathbb{R}^n$ is introduced to store the value of the controller's state vector before the upset occurs. The analysis of the rollback operation uses this frozen vector to compute the frozen output of the controller

$$y_c(k) = -K\hat{\hat{x}}_{r1}(k') = -K\hat{x}_p(k'), \tag{3.6}$$

where $k \in [(k'+1), k_r]$. If the recovery were to reload the controller state vector with its value from q sample periods ago, then the following set q additional vectors would be needed: $\hat{\hat{x}}_{r1}(k), \hat{\hat{x}}_{r2}(k), \ldots, \hat{\hat{x}}_{rq}(k) \in \mathbb{R}^n$. During the rollback process, the additional vectors remain frozen allowing the controller's state vector to be reloaded with its value from q sample periods in the past as follows:

$$\hat{x}_p(k+1) = \hat{\hat{x}}_{rq}(k) = \hat{x}_p(k'+1-q), \tag{3.7}$$

for $k \in [(k'+1), k_r]$. When normal operation resumes at k_rT, the controller's state vector

will already be loaded with the desired value as given by (3.3). The following lemma gives the state equation representations of the closed-loop flight control systems during normal operation and under rollback recovery. The model in the lemma is more general because it allows the possibility that not all the controller states are stored during checkpoints. In that case, the recovery scheme is called Partial State Rollback (PSRB). PSRB makes possible to determine the influence of the state variables on the stability of the system during rollback recovery, providing insight on which states do not need to be stored during the checkpointing process. This is important for designers that need to reduce the size and cost of the system while maintaining adequate performance and robust stability.

Lemma 3.2.1 *Consider the closed-loop observer-based flight control system in Figure 3.1 with* $r(k) \equiv 0$. *The augmented closed-loop system with (partial) state rollback to the value* q *sample periods ago is modeled by*

$$x_{CL_{RB}}(k+1) := \begin{bmatrix} x_p(k+1) \\ \hat{x}_p(k+1) \\ \hat{\hat{x}}_{r1}(k+1) \\ \vdots \\ \hat{\hat{x}}_{rq}(k+1) \end{bmatrix} = A_{j_{RB}} \begin{bmatrix} x_p(k) \\ \hat{x}_p(k) \\ \hat{\hat{x}}_{r1}(k) \\ \vdots \\ \hat{\hat{x}}_{rq}(k) \end{bmatrix}, \tag{3.8}$$

where the augmented closed-loop system matrix for nominal and recovery modes are given,

respectively by

$$A_{0_{RB}} = \begin{bmatrix} A_p & -B_pK & 0 & \cdots & 0 & 0 \\ LC_p & A_p - B_pK - LC_p & 0 & \cdots & 0 & 0 \\ 0 & I_n & 0 & \cdots & 0 & 0 \\ 0 & 0 & \bar{I}_n & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & \cdots & \bar{I}_n & 0 \end{bmatrix}$$

and

$$A_{3_{RB}} = \begin{bmatrix} A_p & 0 & -B_pK & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \bar{I}_n \\ 0 & 0 & I_n & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \bar{I}_n & 0 \\ 0 & 0 & \vdots & & 0 & \bar{I}_n \end{bmatrix}. \tag{3.9}$$

Here $\bar{I}_n$ is a modified $n \times n$ identity matrix with diagonal entries set to zero if the corresponding states are not stored during the checkpointing.

Recall from the previous discussion that the effectiveness of the rollback recovery is influenced by the amount of data stored in each checkpoint. This issue can be explored modifying $\bar{I}_n$ in Lemma 3.2.1. Setting $\bar{I}_n = I_n$ allows the model to store all the state variables' data on each checkpoint (FSRB). On the other hand, when PSRB is used, selected diagonal entries of $\bar{I}_n$ are set to zero, corresponding state variables that are not stored.

3.2.3 Reset/Cold-Restart Recovery Methods.

Reset and cold-restart are two recovery techniques that are also implemented in the RCS. They can be regarded as limiting cases of the rollback technique when $q \rightarrow \infty$. For

the reset case, the information reloaded corresponds to the initial state of the controller (the first checkpoint) for the current mode of operation. Cold-restart is a special case of reset where the controller state data are set to zero. Note that for these two techniques the checkpointing process is not required. Also, the average recovery duration can be smaller for cold-restart, since the states only need to be set to zero.

3.2.3.1 Reset/Cold-Restart Interference Model

The reset and cold-restart recovery techniques consist of the following actions (see Figures 3.1 and 3.4):

- Freeze the controller's output to the last known-good value

$$y_c(k) = y_c(k'), \tag{3.10}$$

  for $k \in [(k'+1), k_r]$.

- Reload the controller's state vector $x_c(k)$ with the values of the first checkpoint. This reloaded state vector is released and made available for processing at the end of the recovery process:

$$x_c(k_r) = x_c(0), \tag{3.11}$$

  where $x_c(0) = 0$ for cold-restart.

To analyze the reset and cold-restart operations, a model similar to the one used for rollback in Lemma 3.2.1 is used. In this case, there is a need for only one additional copy of the last value of the controller's state vector. This vector, $\hat{\hat{x}}_{r1}$, is used only to model the frozen control input. The reload of the state vector is done by introducing an input, $\hat{x}_p(0)$, that is multiplied by a nonzero value only during reload operation. Since $\hat{x}_p(0)$ would be a vector of zeros, there is no need to include it for the cold-restart model.

The following lemma gives the state equation representations of the closed-loop flight control systems during normal operation and under (partial) state reset and cold-restart recovery.

Lemma 3.2.2 *Consider the closed-loop observer-based flight control system in Figure 3.1 with* $r(k) \equiv 0$.

Reset *The augmented closed-loop system with reset is modeled by*

$$x_{CL_{RE}}(k+1) := \begin{bmatrix} x_p(k+1) \\ \hat{x}_p(k+1) \\ \hat{\hat{x}}_{r1}(k+1) \end{bmatrix} = A_{j_{RE}} \begin{bmatrix} x_p(k) \\ \hat{x}_p(k) \\ \hat{\hat{x}}_{r1}(k) \end{bmatrix} + B_{j_{RE}} \hat{x}_p(0), \tag{3.12}$$

where the augmented system matrix for nominal and recovery modes are given, respectively by

$$A_{0_{RE}} = \begin{bmatrix} A_p & -B_pK & 0 \\ LC_p & A_p - B_pK - LC_p & 0 \\ 0 & I_n & 0 \end{bmatrix}, \; B_{0_{RE}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and

$$A_{3_{RE}} = \begin{bmatrix} A_p & 0 & -B_pK \\ 0 & 0 & 0 \\ 0 & 0 & I_n \end{bmatrix}, \; B_{3_{RE}} = \begin{bmatrix} 0 \\ \bar{I}_n \\ 0 \end{bmatrix}. \tag{3.13}$$

Cold-Restart *The augmented closed-loop system with cold-restart is modeled by*

$$x_{CL_{CR}}(k+1) := \begin{bmatrix} x_p(k+1) \\ \hat{x}_p(k+1) \\ \hat{\hat{x}}_{r1}(k+1) \end{bmatrix} = A_{j_{CR}} \begin{bmatrix} x_p(k) \\ \hat{x}_p(k) \\ \hat{\hat{x}}_{r1}(k) \end{bmatrix}, \tag{3.14}$$

where the augmented system matrix for nominal and recovery modes are given, respectively by

$$A_{0_{CR}} = \begin{bmatrix} A_p & -B_pK & 0 \\ LC_p & A_p - B_pK - LC_p & 0 \\ 0 & I_n & 0 \end{bmatrix}$$

and

$$A_{3_{CR}} = \begin{bmatrix} A_p & 0 & -B_pK \\ 0 & 0 & 0 \\ 0 & 0 & I_n \end{bmatrix}. \tag{3.15}$$

3.3 Complete Closed-Loop Models

The main results of this section are Lemmas 3.3.1 and 3.3.2 which give the complete closed-loop system representations by combining the state equations in Lemmas 3.2.1 and 3.2.2 with the stochastic model of the exosystem. These representations are jump linear systems where the closed-loop systems are switched between the nominal and the recovery dynamics. They can be analyzed with the tools presented in Corollary 2.3.2.

Lemma 3.3.1 *Consider the closed-loop flight control system in Figure 3.5 with $r(k) \equiv 0$. Under rare event conditions, the jump linear closed-loop system corresponding to rollback error recovery of the controller states to their value q sample periods ago is given by*

$$\begin{aligned} \boldsymbol{x}_{CL_{RB}}(k+1) &= A_{\tilde{\boldsymbol{\theta}}(k)} \boldsymbol{x}_{CL_{RB}}(k) \\ \boldsymbol{y}(k) &= C\boldsymbol{x}_{CL_{RB}}(k), \end{aligned} \tag{3.16}$$

where $\boldsymbol{x}_{CL_{RB}}(k) = \left[x_p^T(k), \hat{x}_p^T(k), \hat{x}_{r1}^T(k), \ldots, \hat{x}_{rq}^T(k)\right]^T$, $\boldsymbol{x}_{CL_{RB}}(k) \in \mathbb{R}^{(2+q)n}$, $C = [C_p, 0, \ldots, 0]$,

and

$$A_{\tilde{\boldsymbol{\theta}}(k)} = \begin{cases} A_{0_{RB}} & : \tilde{\boldsymbol{\theta}}(k) \text{ is } 0, \\ A_{1_{RB}} & : \tilde{\boldsymbol{\theta}}(k) \text{ is } 3. \end{cases}$$

Lemma 3.3.2 *Consider the closed-loop flight control system in Figure 3.5 with* $r(k) \equiv 0$. *Under rare event conditions, the jump linear closed-loop system corresponding to reset/cold-restart error recovery is given by*

$$\begin{aligned} \boldsymbol{x}_{CL_{RE/CR}}(k+1) &= A_{\tilde{\boldsymbol{\theta}}(k)} \boldsymbol{x}_{CL_{RE/CR}}(k) + B_{\tilde{\boldsymbol{\theta}}(k)} \hat{x}_p \\ \boldsymbol{y}(k) &= C \boldsymbol{x}_{CL_{RE/CR}}(k), \end{aligned} \tag{3.17}$$

where $\boldsymbol{x}_{CL_{RE/CR}}(k) = \left[x_p^T(k), \hat{x}_p^T(k), \hat{x}_{r1}^T(k)\right]^T$, $\boldsymbol{x}_{CL_{RE/CR}}(k) \in \mathbb{R}^{3n}$, $C = [C_p, 0, 0]$, *and*

$$A_{\tilde{\boldsymbol{\theta}}(k)} = \begin{cases} A_{0_{RE/CR}} & : \tilde{\boldsymbol{\theta}}(k) \text{ is } 0, \\ A_{3_{RE/CR}} & : \tilde{\boldsymbol{\theta}}(k) \text{ is } 3. \end{cases}$$

$$B_{\tilde{\boldsymbol{\theta}}(k)} = \begin{cases} B_{0_{RE/CR}} & : \tilde{\boldsymbol{\theta}}(k) \text{ is } 0, \\ B_{3_{RE/CR}} & : \tilde{\boldsymbol{\theta}}(k) \text{ is } 3. \end{cases}$$

3.4 Contributions

The author's main contribution was the development of the interference mappings and closed-loop representation for different error recovery techniques. These representations were tested in several case study examples that will be presented in the following chapter. The results on this chapter are summarized in the following tables.

Table 3.2: Interference mappings from normal operation to operation during recovery.

Rollback	Reset/Cold-Restart
$\mathcal{I} : Z^+ \rightarrow \mathbb{R}^{(q+2)n \times (q+2)n} \times [0,1]$ $j \mapsto \left(\Delta A_j, p_j^u\right)$	$\mathcal{I} : Z^+ \rightarrow \mathbb{R}^{3n \times 3n} \times \mathbb{R}^{3n \times n} \times [0,1]$ $j \mapsto \left(\Delta A_j, \Delta B_j, p_j^u\right)$

Table 3.3: Jump linear system closed-loop representations.

Rollback	Reset/Cold-Restart
$\boldsymbol{x}_{CL_{RB}}(k+1) = A_{\tilde{\boldsymbol{\theta}}(k)} \boldsymbol{x}_{CL_{RB}}(k)$ $\boldsymbol{y}(k) = C\boldsymbol{x}_{CL_{RB}}(k),$	$\boldsymbol{x}_{CL_{RE/CR}}(k+1) = A_{\tilde{\boldsymbol{\theta}}(k)} \boldsymbol{x}_{CL_{RE/CR}}(k) + B_{\tilde{\boldsymbol{\theta}}(k)} \hat{x}_p(0)$ $\boldsymbol{y}(k) = C\boldsymbol{x}_{CL_{RE/CR}}(k),$

Table 3.4: $A_{\tilde{\boldsymbol{\theta}}(k)}$ for rollback recovery.

A_0	A_3
$\begin{bmatrix} A_p & -B_pK & 0 & 0 \\ LC_p & A_p - B_pK - LC_p & 0 & 0 \\ 0 & I_n & 0 & 0 \\ 0 & 0 & \bar{I}_{(q-1)n} & 0 \end{bmatrix}$	$\begin{bmatrix} A_p & 0 & -B_pK & 0 & 0 \\ 0 & 0 & 0 & 0 & \bar{I}_n \\ 0 & 0 & I_n & 0 & 0 \\ 0 & 0 & 0 & \bar{I}_{(q-2)n} & 0 \\ 0 & 0 & 0 & 0 & \bar{I}_n \end{bmatrix}$

Table 3.5: $A_{\tilde{\boldsymbol{\theta}}(k)}$ and $B_{\tilde{\boldsymbol{\theta}}(k)}$ for reset and cold-restart recovery.

A_0	A_3
$\begin{bmatrix} A_p & -B_pK & 0 \\ LC_p & A_p - B_pK - LC_p & 0 \\ 0 & I_n & 0 \end{bmatrix}$	$\begin{bmatrix} A_p & 0 & -B_pK \\ 0 & 0 & 0 \\ 0 & 0 & I_n \end{bmatrix}$
B_0	B_3
$\begin{bmatrix} 0_n & 0_n & 0_n \end{bmatrix}^T$	$\begin{bmatrix} 0_n & \bar{I}_n & 0_n \end{bmatrix}^T$

CHAPTER IV

SIMULATION STUDIES

4.1 Introduction

This chapter presents a collection of simulation results in order to illustrate the application of the theory introduced in Chapters 2 and 3. The idea is to show that the theory can be used not only to predict the stability of a control system with recovery capabilities under certain radiation conditions but also to compare the performance of different recovery configurations under the same radiation conditions. Issues like the benefit of rollback recovery over reset/cold-restart recovery, longer checkpoint schemes and partial state feedback configurations will be addressed by the examples presented in the following sections.

The chapter is organized as follows: Section 4.2 presents a discussion on the restrictions of the theory and the simulation programs developed to test the theory. Section 4.3 provides an overview of the simulation process and is followed by three sections that present the simulation study cases. The final section shows general conclusions about the results presented in this chapter.

4.2 Theory and Simulation Restrictions

The theory developed in Chapters 2 and 3 has several requirements (stated as Assumptions 1-7) that must be satisfied to make valid theoretical accurate predictions about the stability of the systems under analysis. It is useful to highlight that the following two restrictions must be carefully observed:

Condition 1 The mean interarrival spacing has to be much greater than the average recovery duration in order to keep the rare event assumption valid.

Condition 2 The control system's sampling frequency frequency must be high enough to ensure that the sampling of the birth and death process (exosystem) renders a discrete-time Markov chain.

The simulation programs also impose some practical conditions that (may) restrict the characteristics of the system under analysis. Otherwise, extremely long simulation runs would be required. The most important one is the number of upsets that hit the system over the time interval corresponding to the slowest time constant of the closed-loop system. This condition can be stated as follows:

Condition 3 The radiation parameters of the system under simulation must produce enough radiation events to ensure that the simulated state vectors do not reach machine zero. This condition can be stated as: $\lambda \gg 1$.

To understand why this condition is important, consider an error-free (with no upsets) stable discrete-time autonomous linear system with a regulator controller, which implies that $x(k) \longrightarrow \infty$ while $k \longrightarrow 0$. If a computer program were to simulate the dynamics of this system, the variables representing its state vector would reach machine zero in finite time given by a multiple of the slowest time constant. Note that this does not mean that the state vector is really zero, but the computer will not be able to distinguish between its value and zero.

Now, consider the same system under radiation conditions that would theoretically render the system unstable. If the radiation parameters are such that the average interarrival spacing is too big, by the time a radiation event hits the system, the computer's variables

will already be in machine zero. Hence, it would not matter whether the nominal or recovery closed loop A matrix is used to calculate the next state because it would always produce a zero next state. To avoid this finite wordlength effect, it is necessary that the radiation parameters of the simulated system are such that the upsets happen early enough during the simulation time and with a high enough frequency.

These three conditions can be summarized by the following rule of thumb:

$$1 < D_{\mu_D} \ll D_\lambda \ll 1/T \tag{4.1}$$

where T is the sample period, $D_{\mu_D} = \frac{1}{\mu_D T}$ represents the average recovery duration in sample periods and $D_\lambda = \frac{1}{\lambda T}$ represents the mean interarrival spacing in sample periods.

Some remarks about this equation:

- The rightmost inequality in equation (4.1) is a consequence of the Condition 3. Nevertheless, this "much smaller than" relation can be relaxed to a "less than" condition depending on the system under consideration.

- The center inequality in equation (4.1) corresponds to Condition 1.

- The three conditions listed above do not establish a clear relationship among the parameters under consideration. It is a standard engineering practice to consider that if $a \ll b$ then b must be at least an order of magnitude greater than a. This assumption yields $D_\lambda \geq 10 D_{\mu_D}$ and $\frac{1}{T} \geq 10 D_\lambda$.

  In that case, for a standard average recovery duration of 10 sample periods the sampling frequency should be in the order of 1000 Hz, which is uncommon in practical systems. It is clear then that not all the relationships in (4.1) will be satisfied at the same time in most real problems, with the consequent (possible) reduction of accuracy on the theoretical predictions.

While equation (4.1) is not a direct restriction imposed by the theory developed in previous chapters, experience has shown that systems that follow it closely present better agreement between the theory and the simulations.

4.3 Simulation Strategy

The goal is to validate the theoretical stability predictions using Monte Carlo type simulations. Several parameters can affect the stability of the systems under analysis. Among them are the radiation parameter λ, the sample period T, the recovery method used, the plants's continuous-time closed-loop poles, etc. Once these parameters are selected, the validation process is done in two steps. First, the theoretical stability threshold is computed, then several simulations are conducted to verify this threshold.

The theoretical stability threshold is found using the following procedure: For a given plant and set of conditions, a recovery window length is selected (D_{μ_D} is fixed). Then the radiation's mean interarrival spacing (D_λ) is swept through a suitable range. For every value of D_λ, the spectral radius of $\mathcal{A}_1$ (Corollary 2.3.2) is calculated and plotted. The theoretical threshold is given by the minimum value of D_λ that makes $\rho(\mathcal{A}_1) = 1$.

The verification procedure is done as follows: Two values of D_λ are chosen in the vicinity of the threshold value (one smaller and the other one greater than the critical value). For each value a complete Monte Carlo simulation is conducted. The threshold is considered validated if the smaller value of D_λ renders an unstable system and the greater value renders a stable one. An obvious conclusion is that the closer the test values of D_λ are to the threshold values, the better the validation of the theory.

Each simulation is performed as follows: Using one of the test values for D_λ and the designed recovery duration D_{μ_D}, a continuous-time birth and death process is simulated and

sampled. At each sample instant kT, the state of the birth and death process determines which closed-loop model (A_0 or A_3, see Chapter 3) is used to calculate the next state of the system $x(k)$ and to compute and store the value of $\|x(k)\|^2$. This process is repeated until the end of the simulation time (fixed at the beginning of the simulation process).

The procedure described above constitutes one "run" of the simulation. At the end of the last run, the average of $\|x(k)\|^2$ is computed and plotted in logarithmic scale. The average obtained constitutes the estimate of the second order moment of the system and its convergence (or divergence) to zero determines the mean square stability (or instability) of the system.

The following sections will present the simulations results for three examples:

Scalar plant affected by lightning conditions. In this example, the typical lightning environment that an aircraft could encounter during flight is considered the exosystem of a scalar plant with recovery capabilities and its stability is tested under those conditions. This is an example on how the theoretical tools developed in previous chapters can be used as design tools for real applications.

B737 longitudinal landing model. This example examines the stability characteristics of a B737 regulator controller with recovery capabilities. The B737 model used for this study corresponds to the longitudinal dynamics of the aircraft during landing approach. For this example several configurations of the rollback recovery scheme were tested.

F16 longitudinal model. This study is similar to the B737 example but uses the F16 longitudinal dynamical model. Several possible rollback configurations were tested.

4.4 First Simulation Study: Scalar Plant Affected by Lightning Conditions

This study will focus on the indirect effects of lightning in an aircraft (see [1, 11] for a detailed description of the lightning environment). One of the waveforms present during the lightning strike is the so-called Multiple Burst Waveform (MBW). This waveform occurs at the initiation of lightning strikes and randomly throughout the lightning flash duration of $1.5 - 2$ sec and, although unlikely to cause physical damage to the aircraft, it can induce upsets in several of its systems. For this reason, a MBW test set has been devised to evaluate the susceptibility of computer systems to functional upsets during lightning. The set lasts between $61 - 620$ msec. and is composed by three bursts of twenty H waveforms as shown in Figure 4.1. Hence, the indirect effects induced by the MBWs will be considered to trigger upsets in the aircraft controller. In accordance to the model presented in Section 2.2.2 and Sections 3.1 and 3.2, the random process that characterizes the exosystem is assumed to be a birth and death process, where the births are due to MBW arrivals and the deaths correspond to the termination of the recovery process triggered by an arrival if it caused an upset. It is assumed that if a MBW arrival causes an upset, it only happens at the arrival time, so that the duration of the radiation process is not relevant in the analysis. Alternatively, assume that the average duration of a MBW is much less than the duration of the average recovery process. A timing diagram that illustrates a birth and death process for two MBWs is given in Figure 4.2 where the average interarrival spacing is $1/\lambda$ and the average recovery duration is $1/\mu_D$. A useful measure of the robustness of the closed-loop system, including the recovery method dynamics, is the minimum value of the average interarrival spacing of the MBWs that preserves stability. It is important

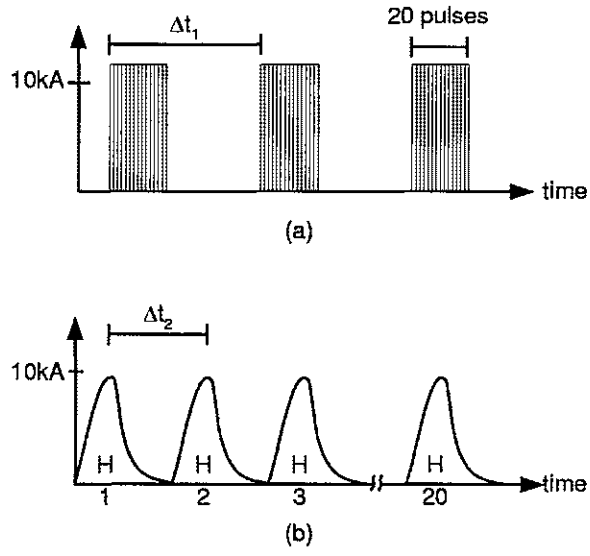

Figure 4.1: (a) Multiple Burst Waveform set. $\Delta t_1 \in [30, 300]$ msec. (b) Burst detail. $\Delta t_2 \in [50, 1000]$ μsec.

that this value be small enough to allow frequent soft faults induced by the MBWs so that the resulting recovery processes do not destabilize the closed-loop system. This minimum value serves as a stability threshold. The closed-loop system will be mean square stable only if the average interarrival spacing is greater than the threshold. This measure can be computed by finding the minimum value of $1/\lambda$ that satisfies $\rho(\mathcal{A}_1) < 1$, the spectral radius of $\mathcal{A}_1$. For this study, consider a plant with a zero-order-hold equivalent representation given by $A_p = 1.009$, $B_p = 1$, $C_p = 1$ and $D_p = 0$ when the sampling period is $T = 0.01$ sec. Assume that the observer-based digital controller is characterized by $K = 0.0097$ and $L = 0.01082$. For rollback recovery assume that $q = 1$, that is, the controller state is reloaded with its value prior to the upset, and consider the shortest possible MBW duration. Corollary 2.3.2 is used to develop the relation between the average length of the

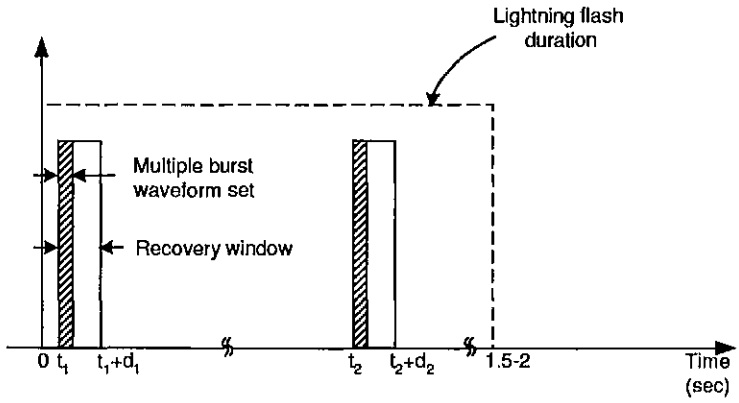

Figure 4.2: Timing diagram for a lightning flash that starts at $t = 0$ sec.

recovery processes and the stability of the closed-loop system. Figure 4.3 shows for each average recovery duration, the corresponding stability thresholds (the minimum interarrival spacing between MBWs on average) for rollback and reset error recovery. It is seen that the closed-loop system with rollback can be less sensitive than the closed-loop systems with reset recovery if the average recovery duration is small enough. For rollback recovery, longer recovery durations require longer average interarrival spacing to preserve stability. Furthermore, recovery durations longer than about 0.55 sec. will always produce an unstable closed-loop system for any (practical) interarrival spacing. The stability results for reset or cold-restart recovery are identical. To preserve stability the average interarrival spacing must be greater than 150 sec. and the recovery duration can be no longer than about 0.55 sec. These conclusions were derived after analyzing Figure 4.3 which clearly shows that for any average recovery duration longer that 0.55 sec., the system will become unstable regardless of the recovery technique being used. Note that until now, the only tool available

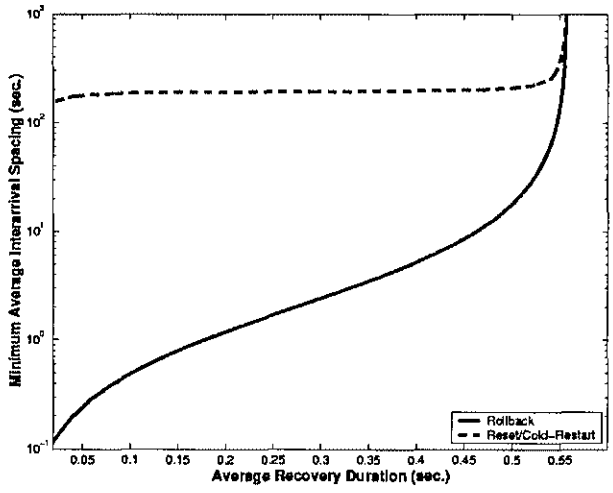

Figure 4.3: Minimum average interarrival spacing as a function of the average recovery duration for the closed-loop scalar systems with rollback and reset error recovery techniques.

to estimate the maximum time delay that preserves the asymptotic stability of the nominal plant with the observer-based compensator was the phase margin computation. Using this technique, the maximum time delay that the system could withstand before becoming unstable was found to be 1.0562 sec. which is clearly an overestimation. Also note that for any average interarrival spacing, Corollary 2.3.2 can provide an estimation of the critical value of the average recovery duration that will preserve the stability. This customization is not possible with phase margin analysis. Now, suppose that the average duration of the recovery is $D_{\mu_D} = 1/(\mu_D T) = 10$ samples, i.e., the recovery lasts on average $1/\mu_D = 100$ msec. For this example, assume that the average duration of the MBWs is much less than 100 msec.

Using Corollary 2.3.2 or Figure 4.3, the stability thresholds for this particular average recovery durations are 0.49 sec., 186.5 sec. and 186.5 sec. for the closed-loop systems

Table 4.1: Minimum Average Interarrival Spacing (MAIS) as a function of q. The average recovery duration is 100 msec.

	$q=1$	$q=1$	$q=3$	$q=4$	$q=5$	$q=6$	$q=7$	$q=8$	$q=9$	$q=10$
MAIS	0.49	0.54	0.58	0.63	0.67	0.72	0.76	0.80	0.85	0.89

with rollback, reset and cold-restart recovery, respectively. These numbers mean that the closed-loop systems with reset or cold-restart error recovery techniques can only tolerate a single upset due to a MBW during a lightning flash, whereas the closed-loop with rollback could tolerate more as long as the upsets are spaced, on average, every 0.49 sec. The results of four Monte Carlo simulations of 500 runs each are shown in Figure 4.4. These plots of the estimates of the mean square values of the states, $\hat{Q}(k)$, verify the stability thresholds. In the top plot, soft faults are injected to the closed-loop system on average every 0.44 sec. Since this rate of injection is less than the stability thresholds, the closed-loop systems with rollback and reset error recovery techniques are unstable. In the bottom figure, the soft faults are injected on average every 0.54 sec. In this case the closed-loop system with rollback is mean square stable, whereas the closed-loop system with reset is not. The simulation results with cold-restart are not shown since they are essentially the same as the closed-loop system with reset recovery. Finally, a study was conducted to analyze the effect of longer rollback schemes, that is, rollback configurations in which the checkpoint data being rolled back corresponds to higher values of q. The stability thresholds were calculated for several values of q for the same average recovery duration of 100 msec. As expected, higher values of q require higher average interarrival spacings, which is desirable. The results are summarized in Table 4.1

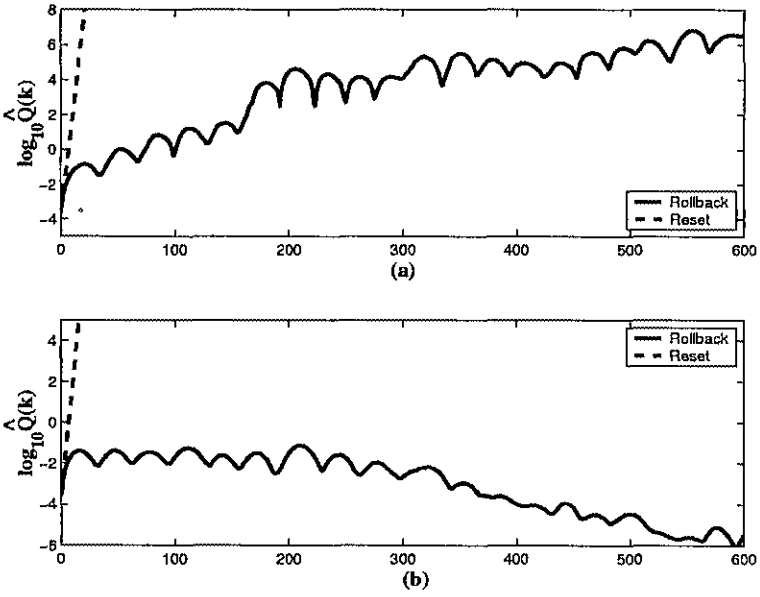

Figure 4.4: Plots of the mean-square value of the states for 500 Monte Carlo simulations of the closed-loop scalar systems with rollback and reset error recovery techniques. A soft fault was injected on average every 0.44 sec. in (a) and every 0.54 sec. in (b).

4.5 Second Simulation Study: B737 Longitudinal Landing Model

This example is of particular interest because it inspired the modeling developed in the previous chapters. In general, in any aircraft flight, the take-off and landing periods are the most critical because of the short time available to correct any potential errors or failures in the aircraft. In particular when the landing is done automatically, like in modern fly-by-wire aircraft, computer upsets induced by radar radiation and other common electromagnetic sources (see [25]), can produce errors in the aircraft control system and jeopardize the

aircraft integrity.

In this context, consider the following B737 longitudinal linear model with zero-order-hold representation:

$$A_p = \begin{bmatrix} 0.999867302536 & 0.000502116422 & -0.056123843400 & -0.128378126451 \\ -0.001117613630 & 0.997454815227 & 0.806019157135 & -0.008837214625 \\ -0.000001289944 & -0.000023008491 & 0.998040510498 & 0.000000185807 \\ -0.000000002398 & -0.000000046038 & 0.003996085917 & 1.000000000236 \end{bmatrix}$$

$$B_p = \begin{bmatrix} 3.026819725 & 41.693713934 \\ 0.014479426 & -601.427775376 \\ 0.050067045 & -75.010364623 \\ 0.000100168 & -0.150074144 \end{bmatrix}, \; C_p = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}, \; D_p = \begin{bmatrix} 0 & 0 \end{bmatrix},$$

where the sampling period is $T = 0.004$. The state vector x_p is given by:

$$x_p(k) = \begin{bmatrix} U_b \\ W_b \\ Q_b \\ \theta \end{bmatrix} = \begin{bmatrix} forward\ velocity \\ down\ velocity \\ pitch\ rate \\ pitch \end{bmatrix}.$$

The nominal closed-loop representation uses a predictor observer-based representation of a stabilizing controller as given in (3.5) where:

$$K = \begin{bmatrix} 0.000765145994 & 0.011253256140 & -3.282498621390 & -0.275546073493 \\ 0.000006744104 & 0.000002866568 & -0.006813539520 & -0.000634632390 \end{bmatrix} \times 10^4$$

$$L = \begin{bmatrix} 788.3897697222 \\ -15.5351533275 \\ 0.4956894746 \\ 0.0604783208 \end{bmatrix}.$$

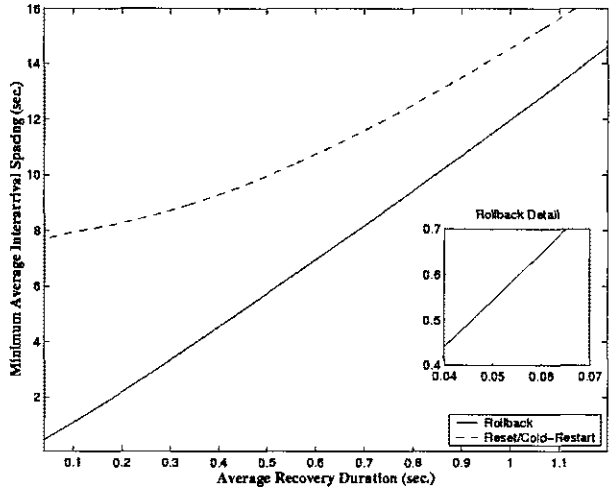

Figure 4.5: Minimum average interarrival spacing as a function of the average recovery duration for B737 example with rollback and reset error recovery techniques when $q = 1$.

Recall from Figure 3.4 the characteristics of rollback recovery. Assume that the checkpointing process lasts much less than one sample period, that is, $L = 0$ and $\Delta \in [0, 1]$, and the checkpoint period is $M = 1$. In this situation, the controller's state vector is reloaded with its value prior to the upset, and therefore $q = 1$. As in the previous simulation study, Corollary 2.3.2 is used to develop the theoretical relation between the average length of the recovery processes and the stability of the closed-loop system. Figure 4.5 shows, for each average recovery duration, the corresponding stability thresholds (the minimum interarrival spacing between upsets on average) for rollback and reset error recovery.

For standard mean recovery durations of 10 to 15 sample periods (0.04 sec. to 0.06 sec.), the rollback recovery scheme will maintain stability for a mean interarrival spacing in the order of 0.44 sec. to 0.65 sec. as seen on the rollback detail inset. Clearly, for this range of recovery durations, the reset technique performance is significantly below than rollback'a

performance because the latter's minimum average interarrival spacing needed for this example is an order of magnitude smaller. Now, suppose that the average duration of the recovery is $D_{\mu_D} = 1/(\mu_D T) = 10$ samples, i.e., the recovery lasts on average $1/\mu_D = 40$ msec. Using Corollary 2.3.2 or Figure 4.5 one can predict that electromagnetic radiation with average interarrival spacing smaller than 0.44 sec. will make the system unstable for all three recovery techniques (reset and cold-restart have similar characteristics). This was confirmed by four Monte Carlo simulations of 500 runs each that are shown in Figures 4.6 and 4.7. These plots of the estimates of $\log \hat{Q}(k)$ verify the stability thresholds. In Figures 4.6a and 4.7a, upsets are injected in the closed-loop system on average every 0.4 sec. Since this rate of injection is less than the stability threshold, the closed-loop system is unstable. In Figures 4.6b and 4.7b, the upsets are injected on average every 0.48 sec. In this case the closed-loop system with rollback is mean square stable whereas the closed-loop system with reset is not. The simulation results with cold-restart are not shown since they are essentially the same as the closed-loop system with reset recovery. Figures 4.6 and 4.7 also include a line with the slope predicted by equation (2.16). These lines were placed arbitrarily because the "y-intercept" is unknown since it depends on the initial conditions of the distributions of x_0 and v. Therefore, it was sufficient to compare the slope of the line asymptote to the slope of $\log \hat{Q}(k)$ as $k \to \infty$. It is seen that there is good agreement with some of the simulations but not in all of them. One source for the difference in slope is whether the eigenvalues corresponding to the spectral radius have multiplicity one or not. Another source of inaccuracy is related to equation (4.1). For all the simulations, $D_{\mu_D} = 10$ samples and D_λ was either $D_\lambda = 100$ or $D_\lambda = 120$ samples. Considering that $1/T = 250$, it is clear that D_λ is not much smaller than the sample frequency. Hence equation (4.1) is

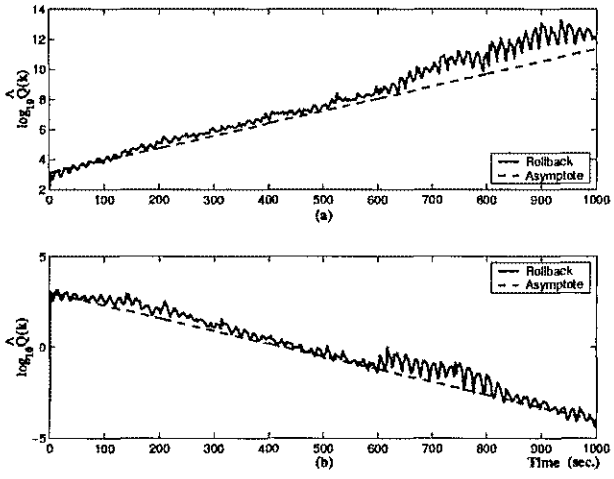

Figure 4.6: Plots of the $\log_{10}\hat{Q}(k)$ vs. time and theoretical plots of $k \times \log_{10}(\rho A_1)$ for B737 example with rollback recovery scheme. In (a), the average interarrival spacing is 0.4 sec. and in (b) it is 0.48 sec.

being approximated by:

$$1 < D_{\mu_D} = 10 \ll D_\lambda = 100 \text{ or } D_\lambda = 120 < 1/T = 250$$

In Chapter 3 it was stated that, in order to save storage space, some designers prefer not to roll back all the available states. This possibility was also explored for this system. Figure 4.8 shows a comparative plot of the stability characteristics for different partial rollback schemes. In this Figure, FSRB and PSRB i denotes full and partial state rollback respectively. In the latter case, the i denotes the state that is *not* rolled back. This plot suggests that not rolling back the forward or down velocities improves the system behavior because for the same mean recovery duration, the required minimum interarrival spacing is significantly smaller. On the other hand, not rolling back the pitch rate or the pitch angle sensible deteriorates the system performance as expected.

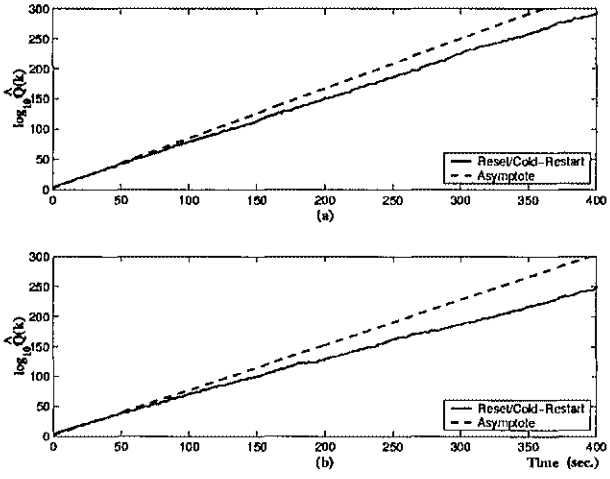

Figure 4.7: Plots of the $\log_{10} \hat{Q}(k)$ vs. time and theoretical plots of $k \times \log_{10} \rho(\mathcal{A}_1)$ for B737 example with reset recovery scheme. In (a), the average interarrival spacing is 0.4 sec. and in (b) it is 0.48 sec.

Simulations were used to validate these theoretical results. In particular, when the down velocity is not rolled back (PSRB 2), the stability threshold corresponds to a minimum interarrival spacing of 0.260 sec.; that is almost 40% smaller than the minimum interarrival spacing for full state rollback. This threshold was confirmed by two simulations in which the interarrival spacing was set to 0.18 sec. and 0.34 sec. respectively. The results are presented in Figure 4.9. Note that in this case equation (4.1) is approximated by:

$$1 < D_{\mu_D} = 10 < D_\lambda = 45 \text{ or } D_\lambda = 85 < 1/T = 250$$

Finally, an investigation of the effect of the duration of the checkpointing processes was also conducted for this system. Table 4.2 shows the minimum interarrival spacings computed for several checkpointing durations when the average recovery duration is 10 samples. For each

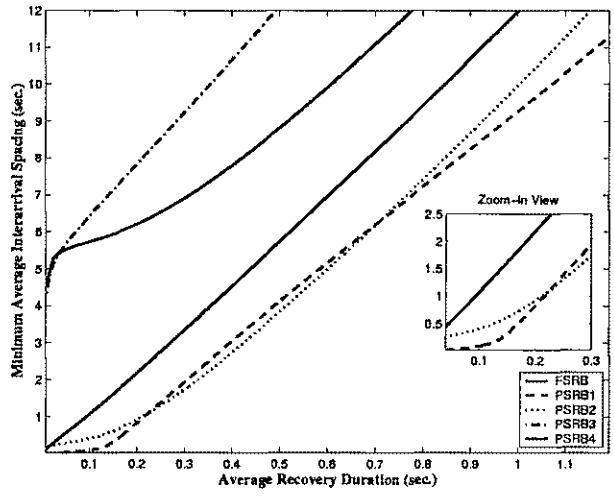

Figure 4.8: Minimum average interarrival spacing as a function of the average recovery duration for B737 example with full state and partial state rollback when $q = 1$.

integer number of checkpointing sample periods, L, the stability threshold was computed for the corresponding worst possible q. As expected, as L and hence q are increased, the stability thresholds deteriorate. For this particular example, PSRB 1 presents the bigger rate of deterioration while PSRB 4 is almost insensible to bigger values of q. Nevertheless, in all the cases the rate of deterioration is small.

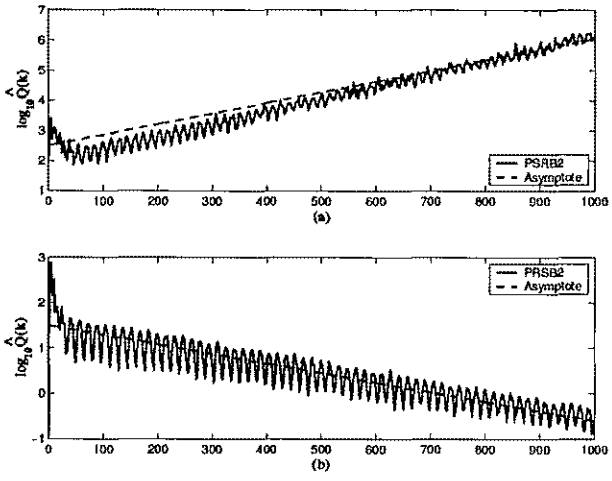

Figure 4.9: Plots of the $\log_{10} \hat{Q}(k)$ vs. time and theoretical plots of $k \times \log_{10} \rho(\mathcal{A}_1)$ for B737 example with PSRB 2 recovery scheme. In (a), the average interarrival spacing is 0.18 sec. and in (b) it is 0.34 sec.

Table 4.2: Comparison of the stability thresholds (minimum interarrival spacing between upsets) assuming $M = L + 1$ for B737 example.

Method	$L = 0$, $q = 2$	$L = 1$, $q = 4$	$L = 2$, $q = 6$
FSRB	0.484 sec.	0.564 sec.	0.644 sec.
PSRB 1	0.028 sec.	0.036 sec.	0.044 sec.
PSRB 2	0.272 sec.	0.288 sec.	0.304 sec.
PSRB 3	5.548 sec.	5.632 sec.	5.724 sec.
PSRB 4	5.456 sec.	5.460 sec.	5.460 sec.

4.6 Third Simulation Study: AFTI/F-16 Longitudinal Model

Consider the linearized model of the AFTI/F-16 aircraft longitudinal dynamics given in [12] where the relevant states are the change in speed, the angle of attack, the pitch rate and the pitch angle. The sampling period was selected to be $T = 0.004$ sec., the nominal continuous-time closed-loop poles were placed at $\{-0.2 \pm j0.9798, -0.01 \pm j0.0995\}$ and the discrete-time state observer has poles that are five times faster.

As in the previous example, Corollary 2.3.2 is used to analyze the mean-square stability of the closed-loop flight control system. From the previous example, it is known that the rollback recovery scheme shows better performance than the reset or cold-restart recovery methods. Therefore, only rollback will be considered in this study.

Figure 4.10 shows for each average recovery duration the corresponding stability thresholds in terms of the minimum average interarrival spacing between upsets for full and partial state rollback assuming $L = 1$, $M = 2$ and the corresponding worst-case value for q, that is, $q = 4$. As in the previous example FSRB and PSRB i denote full and partial state rollback, respectively, with i denoting the state that is not rolled back. Note that, as expected, the minimum average interarrival spacing increases as the average recovery duration does. Also note that this relation is different for full state and each partial state rollback. The worst-case corresponds to not rolling back the angle of attack (PSRB 2). In this case, the minimum average interarrival spacing that the system can withstand without becoming unstable is larger than 10 sec. for any recovery duration. This is the reason it was not included in the figure. This behavior means that the angle of attack must be included in any rollback scheme. Similarly, not rolling back the pitch rate (PSRB 3) leads to a significant deterioration of the stability thresholds. On the other hand, not rolling back the change in speed state (PSRB 1) has little effect on the thresholds. Surprisingly, not rolling back the

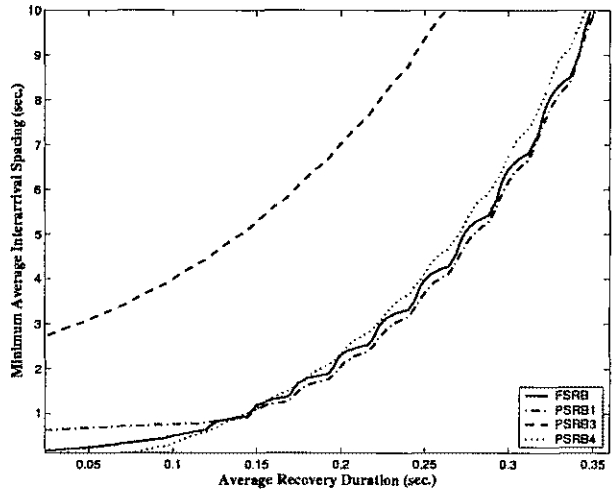

Figure 4.10: Plot of the minimum average interarrival spacing as a function of the average recovery duration for closed-loop F-16 with full state and partial state rollback when $L = 1$ and $q = 4$.

pitch angle (PSRB 4) leads to slightly better thresholds if the average recovery duration is small enough.

The analytically derived stability thresholds in Figure 4.10 were confirmed by Monte Carlo simulations (500 runs each) for an average recovery duration of 10 samples (0.04 sec.) as shown in Figure 4.11. These plots confirm the stability thresholds for full state rollback and partial state rollback when the change in speed was not rolled back. The minimum average interarrival spacing for these cases are 0.2120 sec. and 0.668 sec. respectively. For simulation, average interarrival spacings on both sides of the thresholds were chosen for each case (see Figure 4.11 for details).

The improved behavior for PSRB 4 was also tested. The stability threshold for PSRB 4 corresponds to an interarrival spacing of 0.068 sec. Hence, simulations were run with

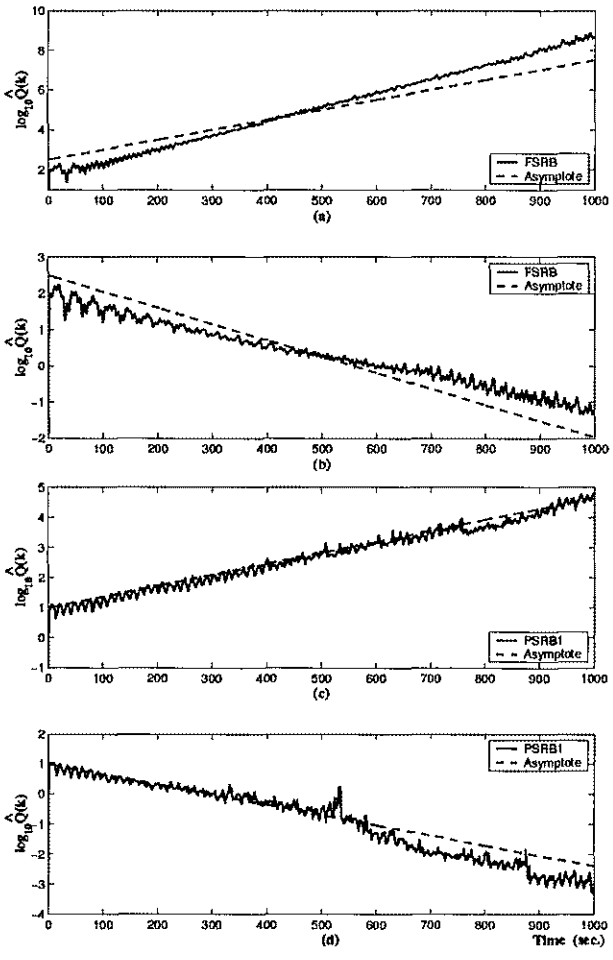

Figure 4.11: Plots of $\log_{10} \hat{Q}(k)$ vs. time computed by Monte Carlo simulation for closed-loop F-16 when $L = 1$ and $q = 4$. For the FSRB simulations, the average interarrival spacings simulated were (a) 0.172 sec. and (b) 0.252 sec. Similarly, for PSRB 1 simulations, the average interarrival spacings simulated were (c) 0.588 sec. and (d) 0.733 sec.

interarrival spacings of 0.048 sec. and 0.084 sec. respectively. The results are shown in Figure 4.12. Note that in this case, the simulation plots does not approximate correctly the theoretical asymptotes given by 2.16. The explanations is that the rare event assumption is barely being satisfied because equation 4.1 is being approximated by:

$$1 < D_{\mu_D} = 10 < D_\lambda = 12 \text{ or } D_\lambda = 21 < 1/T = 250$$

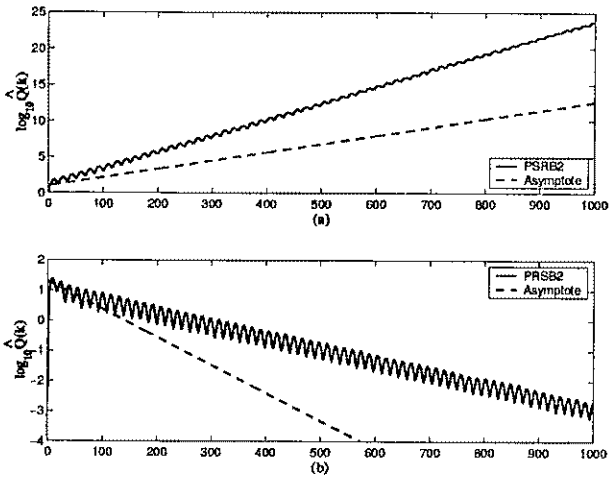

Figure 4.12: Plots of $\log_{10} \hat{Q}(k)$ vs. time computed by Monte Carlo simulation for closed-loop F-16 when $L = 1$ and $q = 4$. For the PSRB 4 simulations, the average interarrival spacings simulated were (a) 0.048 sec. and (b) 0.084 sec.

Because the second inequality is far from expressing a "much less than" relationship, the simulations do not follow the theory as close as possible. Nevertheless, the simulations still confirm the boundary as expected.

An investigation of the effect of the duration of the checkpointing processes was also conducted for this system. The minimum interarrival spacing was computed for several checkpointing durations in Figure 4.13 when the average recovery duration is 10 samples. For each integer number of checkpointing sample periods, L, the stability threshold was computed for the corresponding worst possible q. For this example and the range of checkpoint durations considered, there is an affine relation between the checkpoint duration and the stability thresholds: as L and hence q are increased, the stability thresholds deteriorate. Figures 4.10 and 4.13 can be used to trade-off between mean-square stability and complexity of error recovery rollback systems.

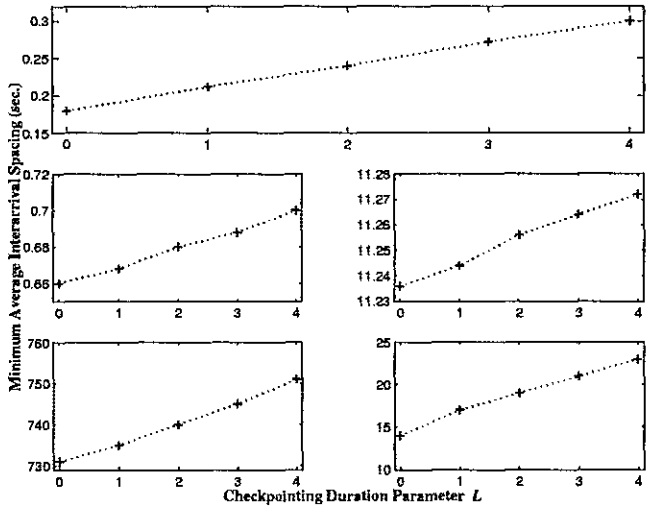

Figure 4.13: Comparison of the stability thresholds (minimum average interarrival spacing between upsets) for different values of L assuming $M = L + 1$ and the corresponding maximum value for q.

4.7 Conclusions

In general, the simulations show that the theory predicts with very good accuracy the stability thresholds for the systems under consideration. The simulations also confirmed the intuitive idea that the rollback recovery will present better behavior than reset or cold-restart recoveries. The reason is simple: only rollback can restore the controller with data very close to the one the controller had before failing. Finally, it is also clear from the simulations, that the partial state feedback is not only important for cost-effective designs, but could also lead to a surprising improvement in stability robustness. This was clearly show in the two aircraft examples presented above.

CHAPTER V

CONCLUSIONS AND FUTURE RESEARCH

This research had two main goals. The first was to develop theoretical tools to analyze the mean square stability of closed-loop systems with error recovery capabilities. This goal was fully accomplished by extending the theory developed in [9, 14–19].

- The exosystem model in [18] introduced a birth and death process to model the arrival and duration of radiation events. For systems with error recovery capabilities it has been shown that the birth and death process could be used to model the start and duration of a recovery process. Hence, this "new exosystem" is still modeled as a Markov chain that switches the closed-loop representation of a discrete-time linear system.

- A new interference mapping was defined for this application. The resulting closed-loop models can simulate the behavior of three different error recovery techniques: rollback, reset and cold-restart.

- For the rollback recovery technique a very useful model of the checkpointing process was devised. This model permits the analysis the rollback technique for different checkpointing setups.

The second goal, to compare the characteristics of different fault recovery methods, was also accomplished. The following steps were taken to address this goal:

- A Simulink model of Boeing's 737 autoland controller implemented in Honeywell's Recoverable Computer System was studied in order to analyze and understand the

behavior of the recovery mechanisms implemented in a state of the art controller (see [25,26]). This knowledge was used to create the new interference models presented in Chapter 3.

- New simulation programs were developed and extensive simulations, using several aircraft examples were performed to validate the new developed models. These simulations showed very good agreement with the theory and demonstrated that the technique with better performance is rollback.

- It was also possible to quantify how better rollback recovery was in comparison to reset or cold-restart recovery using the critical boundary plots presented in Chapter 4.

- Several rollback configurations were explored, including different checkpointing situations and partial state feedback, which proved to be a great improvement not only because its cost-effectiveness but also because its potential for better stability characteristics.

There are several issues that deserve to be addressed in future research:

- Fixed recovery window duration, multiple radiation arrivals during the recovery process or radiation events present when the recovery is completed are situations not addressed by the current models. In order to include these cases, a more general framework than the current Markov process theory needs to be used. Hybrid systems, semi-Markov models or automata theory are possible alternatives that might be explored.

- Better and more accurate (and hence more complex) models of the recovery mechanisms need to be developed. This also requires a more general framework.

- An in-depth study of the numerical issues of digital simulations needs to be conducted to improve the reliability and accuracy of the current simulation tools.

REFERENCES

[1] 'Aircraft Lightning Environment and Related Test Waveforms Standard,' Report of EUROCAE WG-31 and SAE Committee AE4L, 1997.

[2] P. J. Antsaklis and A. N. Michel *Linear Systems*, McGraw-Hill, New York, 1997.

[3] C. M. Belcastro, 'Closed-loop HIRF experiments performed on a fault tolerant flight control computer,' *Proc. 16th Digital Avionics Systems Conference*, Philadelphia, PA, pp. 4.1 – 40–54, 1997.

[4] C. M. Belcastro, 'Overview of the Systems and Airframe Failure Emulation Testing & Integration (SAFETI) Laboratory at NASA Langley Research Center,' *Proc. of the 2001 International Conference on Lightning and Static Electricity*, Philadelphia, PA. 2001.

[5] J. W.Brewer, 'Kronecker Products and Matrix Calculus in System Theory,' *IEEE Transactions on Circuits & Systems*, Vol.25, no.9, pp.772-81, September 1978.

[6] K. Benjelloun, E. K. Boukas and P. Shi, 'Robust Stochastic Stability of Discrete-Time Linear Systems with Markovian Jumping Parameters,' *Proc. 36th IEEE Conference on Decision and Control*, San Diego, California, pp. 559-564, 1997.

[7] D. W. Caldwell and D. A. Rennels, 'Minimalist Recovery Techniques for Single Event Effects in Spaceborne Microcontrollers,' *IEEE Dependable Computing for Critical Applications 7.*, Piscataway, NJ, pp. 47-65, 1999.

[8] O. L. V. Costa and M. D. Fragoso, 'Stability Results for Discrete-Time Linear Systems with Markovian Jumping Parameters,' *J. Mathematical Analysis and Applications*, Vol. 179, pp. 154-178, 1993.

[9] M. Doğan, 'Stability Analysis and Augmentation of a Closed-Loop Computer Control System Subject to Electromagnetic Disturbances,' Master of Science Thesis, Electrical Engineering Department, Old Dominion University, 1999.

[10] Y. Fang, 'A New General Sufficient Condition for Almost Sure Stability of Jump Linear Systems,' *IEEE Trans. on Automatic Control*, Vol. 42, No. 3, pp. 378-382, March 1997.

[11] F. A. Fisher et al., *Lightning Protection of Aircraft*, Pittsfield, MA: Lightning Technologies, Inc., 1990.

[12] B. Friedland, *Control System Design, An Introduction to State-Space Methods*, McGraw-Hill, New York, 1986.

[13] E. Gelenbe and M. Hernández, 'Optimum Checkpoints with Age Dependant Failures,' *Acta Informatica*, West Germany, Springer-Verlag, Vol 27, pp. 519–31, 1990.

[14] W. S. Gray and O. R. González, 'Modeling Electromagnetic Disturbances in Closed-Loop Computer Controlled Flight Systems,' *Proc. 1998 American Control Conference*, Philadelphia, PA, pp. 359-364, 1998.

[15] W. S. Gray, O. R. González, M. Doğan, 'Digital Linear State Feedback Control Subject to Electromagnetic Disturbances,' *Proc. 1999 American Control Conference*, San Diego, CA, pp. 3500-3504, 1999.

[16] —,'Stochastic Perturbation Analysis of Computer Control Systems Subject to Electromagnetic Disturbances,' *Proc. 1999 Conf. Control Applications*, Honolulu, HI, pp. 1797-1802, 1999.

[17] —,'Stochastic Perturbation Models of Electromagnetic Disturbances in Closed-Loop Computer Control Flight Systems,' *Proc. 18th DASC Digital Avionics Systems Conference*, St. Louis, Missouri, pp.10.C.4-1-8, 1999.

[18] —, 'Stability Analysis of Digital Linear Flight Controllers Subject to Electromagnetic Disturbances,' *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-36, No. 4, pp. 1204-1218, October 2000.

[19] W. S. Gray, O. R. González, S. Patilkulkarni, 'Stability of Digital Control Systems Subject to Jump Linear Random Perturbations,' *Proc. 39th IEEE Conference on Decision and Control*, Sidney, Australia, pp. 1154-1159, 2000.

[20] O. R. González, W. S. Gray, and S. Patilkulkarni, 'Analysis of Memory Bit Errors Induced by Electromagnetic Interference in Closed-Loop Digital Flight Control System,' *Proc. 19th DASC Digital Avionics Systems Conference*, Philadelphia, Pennsylvania, pp. 3.C.5-1-9, 2000.

[21] O. R. Gonzalez, W. S. Gray and A. Tejada, 'Analytical Tools for the Design and Verification of Safety Critical Control Systems,' *Proc. of the 2001 International Conference on Lightning and Static Electricity*, Seattle, Washington, 2001.

[22] O. R. González, W. S. Gray, A. Tejada and S. Patilkulkarni, 'Stability Analysis of Upset Recovery Methods for Electromagnetic Interference,' *Proc. 20th DASC Digital Avionics Systems Conference*, Daytona Beach, Florida, 2001.

[23] O. R. González, W. S. Gray, A. Tejada and S. Patilkulkarni, 'Stability Analysis of Electromagnetic Interference Upset Recovery Methods,' *Proc. 40th IEEE Conference on Decision and Control*, Orlando, Florida, pp. 4134-4139, 2001.

[24] O. R. González, A. Tejada and W. S. Gray 'Analysis of Design Trade-Offs in the Rollback Recovery Method for Fault Tolerant Digital Control Systems,' *Proc. 2002 American Control Conference*, Anchorage, Alaska, pp. 4801-4806, 2002.

[25] R. Hess, 'Computing Platform Architectures for Robust Operation in the Presence of Lightning and Other Electromagnetic Threats,' *Proc. 16th Digital Avionics Systems Conference*, Philadelphia, PA, pp. 4.3 – 9–16, 1997.

[26] R. Hess and M. Malekpour, 'Rapid Soft Fault Recovery,' *Proc. of the 2001 International Conference on Lightning and Static Electricity*, Philadelphia, PA. 2001.

[27] R. Hess and C. M. Belcastro, 'Design and Verification of Robust Architectures for Electronic Systems,' *Proc. of the 2001 International Conference on Lightning and Static Electricity*, Philadelphia, PA. 2001.

[28] Y. Ji and H. J. Chizeck 'Jump Linear Quadratic Gaussian Control: Steady-State Solution and Testable Conditions,' *Control Theory and Advanced Technology*, Vol 6, No. 3, pp. 289-319, 1990.

[29] Y. Ji, H. J. Chizeck, X. Feng and K. A. Loparo ' Stability and Control of Discrete-Time Jump Linear Systems,' *Control Theory and Advanced Technology*, Vol 7, No. 2, pp. 247-270, 1991.

[30] B. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison Wesley Publishing Co., 1989.

[31] H. Kim and K. G. Shin, 'Modeling of Externally-Induced/Common-Cause Faults in Fault-Tolerant Systems' *Proc. 19th DASC Digital Avionics Systems Conference* New York, NY, pp. 402–407, 1994.

[32] J. H. Lala and R. E. Harper, 'Architectural Principles for Safety-Critical Real-Time Applications' *Proceedings of the IEEE*, Vol. 82, No. 1, pp. 25–40, January 1994.

[33] J. H. Lala and R. E. Harper, 'Fault Tolerance in Embedded Real-Time Systems: Importance and Treatment of Common-Mode Failures' *Lecture Notes on Computer Science*. Berlin Heidelberg: Springer-Verlag, No 774, pp. 263–282, 1994.

[34] K. A. Loparo, X. Feng, 'Stability of Stochastic Systems,' In W. S. Levine (ED.), *The Control Handbook.*, Boca Raton, Florida, The CRC Press, 1996.

[35] M. Malekpour and W. Torres, 'Characterization of a Flight Control Computer with Rollback Recovery,' *Proc. 19th DASC Digital Avionics Systems Conference*, Philadelphia, PA, pp. 3.C.4 - 1-8, 2000.

[36] M. Elnozahy, L. Alvisi, Y. Wang and D. B. Johnson,'A Survey of Rollback-Recovery Protocols in Message-Passing Systems.'*Technical Report CMU-CS-99-148, School of Computer Science, Carnegie Mellon University*, June 1999.

[37] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes, Fourth Edition*, McGraw-Hill, New York, 2002.

[38] D. K. Pradham, *Fault-Tolerant Computer System Design* , Prentice Hall PTR, 1996.

[39] M. Pflanz and H. T. Vierhaus, 'Online Check and Recovery Techniques for Dependable Embedded Processors' *IEEE Micro.* Vol. 21, No. 5, pp. 24–40, September-October 2001.

[40] Y. Tamir and M.Vierhaus, 'High-Performance Fault-Tolerant VLSI Systems Using Micro Rollback' *IEEE Transaction on Computers.* Vol. 39, No. 4, pp. 548–54, April 1990.

APPENDIX

SIMULATION ENVIRONMENT.

A.1 Overview of the Simulation Environment

This appendix summarizes the procedure followed to develop the examples presented in Chapter 4. This procedure comprised several steps:

Identification of suitable examples : This step included the selection of different plants along with the proper set of test parameters such as the desired recovery method, radiation parameters, etc.

Creation of data script files : Each of these files contain the examples' parameters in the appropriate format and each is used as an input file by the simulation programs.

Identification of the stability thresholds : Using the same procedure as in chapter 4.

Selection of the final simulation parameters : After identifying the stability thresholds, the final radiation parameters had to be include in the data script files to run the Monte Carlo simulations.

The first three steps are normally iteratively repeated until the selected parameters make the examples satisfy equation (4.1) or until the they show the desired stability characteristics. After the examples were identified and their simulation parameters located, the simulations were executed and the data collected and plotted.

The mathematical analysis tools and the simulation programs were developed under Matlab

release 12.1 running in a Windows Xp station with a Pentium 4 processor. These programs can be divided in two categories:

Analysis Tools This category includes a set of Matlab functions and script files that calculate and display the theoretical stability thresholds and the plots of the minimum average interarrival spacing vs. the average recovery duration (also known as the critical boundary). There are three main files in this category: the data script files, the stability threshold calculator and the critical boundary calculator.

Simulation Tools These programs were originally developed by Dr. W. S. Gray, Dr. O. R. González, M. Doğan and S. Patilkulkarni. These simulation programs take any data script file and generate a continuous-time birth and death process using the radiation parameters given by the data file. This process is sampled and the resulting Markov chain is stored. The chain is then used to switch the parameters of the plant during the dynamical simulation. Finally, these programs collect several statistical measures and generate the required plots and reports.

There are also other support programs that are used for very specific tasks (like creating the figures for these documents).

A.1.1 The Data Script Files

These are the user interface files. In them, the user defines the following parameters:

- The plant's continuous-time state space model.
- The desired continuous-time closed loop poles.
- The sample period T and the radiation parameters μ and λ.

- The desired recovery method model and its parameters

These script files are used by both the analysis and the simulation programs to generate the experimental data. The next section will show the actual script files used to generate these document's examples.

A.1.2 The Stability Threshold Calculator

This Matlab function takes as input a data script file and an array containing a range of values for D_λ and calculates for all values of D_λ on the given range and for a fixed value of D_μ (included inside the script file) the corresponding spectral radius of $\rho(\mathcal{A}_1)$ using the definition provided in equation 2.23. The data is then presented graphically as a plot of $\rho(\mathcal{A}_1)$ vs. D_λ. This function also returns two arrays containing the data used to make the plot.

The syntax of this function is:

$$[sDl, sRad] = TB\_genV3(examplefname, [Dlmin, Dlstep, Dlmax])$$

where:

- $examplefname$ is the name of the data script file being analyzed.
- $[Dlmin, Dlstep, Dlmax]$ is the input array used to specify the value range for D_λ.
- sDL is an output array containing the values of D_λ considered for the calculations.
- $sRad$ is an output array containing the values of $\rho(\mathcal{A}_1)$ for each value of D_λ on sDL.

Figure A.1 shows an example of the output for this function for the F-16 example, rollback configuration. $(tb\_genv3('ACC02\_F16\_NM', [1, 1, 250]))$.

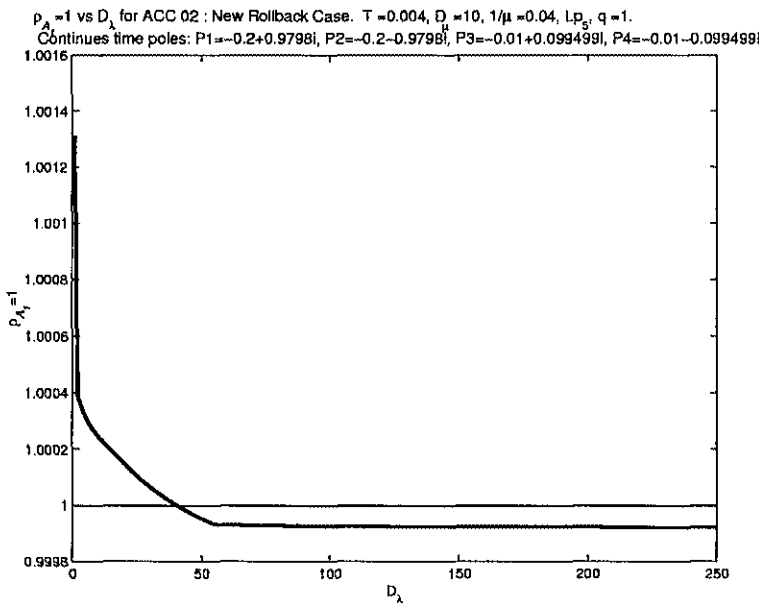

Figure A.1: Example of the output of TB_genV3. The example considered is the FSRB case of the F-16 example with $q = 1$.

A.1.3 The Critical Boundary Calculator

This function is very similar to the previous one with the difference that the value of D_μ is also varied within a range provided by the user. Hence, the output information is presented as a three-dimensional plot. After the data is gathered, the resulting plot is intersected with the plane $\rho(\mathcal{A}_1) = 1$ to produce the plot of the minimum average interarrival spacing vs.the average recovery duration.

The syntax of this function is:

$$[sDl, sDu, sRad] = TB\_gen3DV1(examplefname, [Dlmin, Dlstep, Dlmax], [Dumin, Dustep, Dumax])$$

where:

- *examplefname* is the name of the data script file being analyzed.
- $[Dlmin, Dlstep, Dlmax]$ is the input array used to specify the value range for D_λ.
- $[Dumin, Dustep, Dumax]$ is the input array used to specify the value range for D_μ.
- sDL and sDu are output arrays containing the values of D_λ and D_μ considered for the calculations.
- $sRad$ is an output array containing the values of $\rho(\mathcal{A}_1)$ for each pair of value of D_λ and D_μ on sDL and sDu.

Figure A.2 shows an example of the output for this function for the F-16 example, rollback configuration. $(TB\_gen3DV1('ACC02\_F16\_NM\_3of4', [30, 30, 2490], [6, 6, 102]))$.

A.1.4 The Simulation Programs

The simulation programs used for this research are the same developed by the author of [9] for his Master of Science thesis. Only small modifications were introduce to both make them faster (using equation 4.1) and to let them accept the data script files as inputs. The main simulations programs are:

run_sim_allc This is program the user executes to run a simulation. This program takes a data script file and a set of parameters and calls the appropriate functions to perform the simulations.

Dynsimpu_allc This function is the core simulation program. It takes the data provided by "run_sim_allc" and generates the dynamical simulation of the system. This program also calculates the statistics of the simulated system.

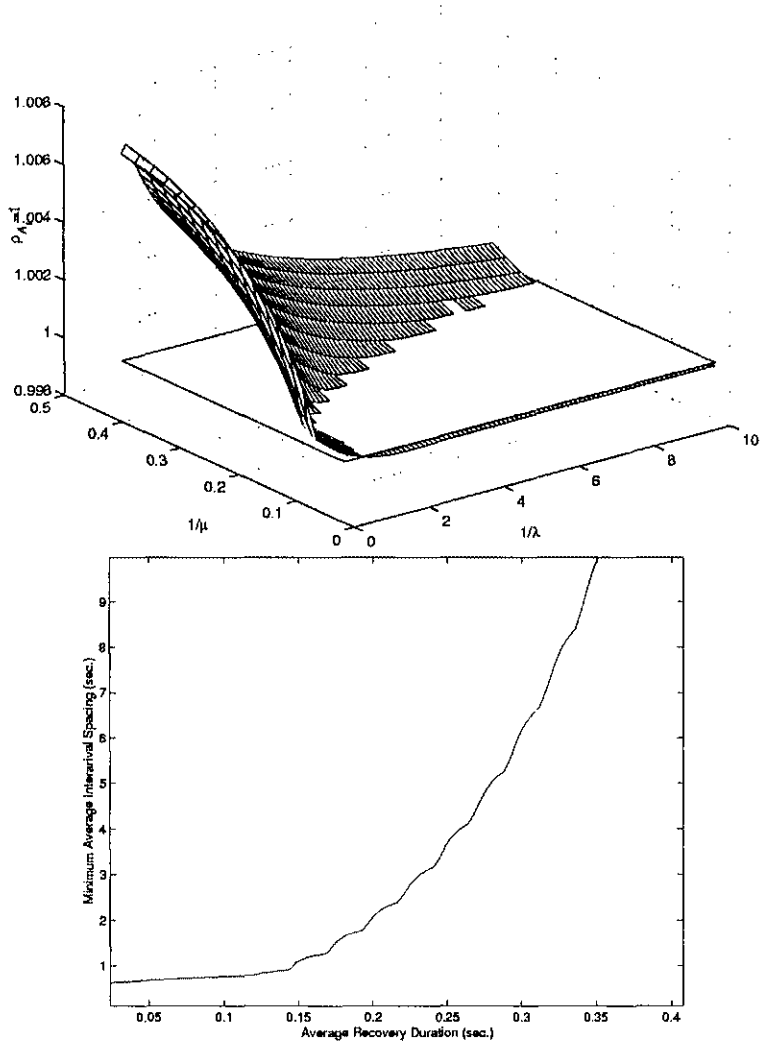

Figure A.2: Example of the output of TB_gen3DV1. Top: three-dimensional plot of $\rho(\mathcal{A}_1)$ as a function of D_μ and D_λ (alternative $1/\mu$ and $1/\lambda$). Bottom: plot of the minimum average interarrival spacing vs.the average recovery duration. Both plots are for the F-16 example with PSRB1 and $q = 1$.

Plot_data_allc This function formats the simulation data and outputs it in graphical and Latex formats. A large portion of this file was developed by S. Patilkulkarni.

For a detailed explanation on this programs please refer to the thesis cited above [9].

A.2 The Examples' Data Script Files

A.2.1 Scalar Plant under Lightning Conditions: Rollback

```
% This script produces the ICOLSE 2001 scalar example data using

% the new Reset and Rollback models.

% Created by Arturo Tejada on April 5, 2001

% Plant

    A=1.009

    B=1

    C=1

    D=0

    F=0.0097

    Lp=0.01082

    T=0.01

% Radiation Parameters

    D_lambda=100;

D_mu=10;

   p0star=0;

p1star=1;

% Fixed Parts of A0

   n=size(A,1);

       qRB=1;

       A011= [A      , -B*F         ;

 Lp*C    , A-B*F-Lp*C ];
```

```
%Variable parts of A0

  A012= [zeros(2*n,qRB*n)];

  A021= [zeros(qRB*n,n)];

  A022= [eye(qRB*n) zeros(qRB*n,n)];

  A0=[A011 A012; A021, A022];

%Variable parts of A3

      A311= [A        ,   0*eye(n)   ,   -B*F , zeros(n,(qRB-1)*n)];

  A312= [zeros(n,(qRB+1)*n),eye(n) ];

  A321= [zeros(qRB*n,2*n)];

    A322= [eye(qRB*n)];

      A3worst=[A311 ;A312; A321, A322];

test_name=strcat(' ICOLSE01, NM Rollback (',date,'): T = ',num2str(T),...

  ', D_{\mu} = ',num2str(D_mu),', 1/\mu = ',num2str(D_mu*T),...

  ', Lp = ',num2str(Lp));

x0=[ pi/200 zeros(1,n*(qRB+1))]';
```

A.2.2 Scalar Plant under Lightning Conditions: Reset/Cold-Restart

```
% This script produces the ICOLSE 2001 scalar example data
% using the new Reset and Rollback models.
% Created by Arturo Tejada on April 5, 2001
% Plant
    A=1.009
    B=1
    C=1
    D=0
    F=0.0097
    Lp=0.01082
    T=0.01
% Radiation Parameters
    D_lambda=100;
D_mu=10;
   p0star=0;
p1star=1;
% Fixed Parts of A0
   n=size(A,1);
       A0=   [A       , -B*F        , 0*eye(n) ;
 Lp*C     , A-B*F-Lp*C , 0*eye(n) ;
            0*eye(n), eye(n)      , 0*eye(n)];
 %Variable parts of A3
       A3worst= [A            ,   0*eye(n)   ,   -B*F;
```

```
                zeros(n,3*n);

                zeros(n,2*n),   eye(n)];

test_name=strcat(' ICOLSE01, NM Reset (',date,'): T = ',num2str(T),...

   ', D_{\mu} = ',num2str(D_mu),', 1/\mu = ',num2str(D_mu*T),...

   ', Lp = ',num2str(Lp));

x0=[0 pi/200 0 0 zeros(1,n*2)]';
```

A.2.3 B737 Longitudinal Landing Model: FS Rollback

```
% B737_NM  Data file
% Created by Arturo Tejada
% original July 11, 2001.
% This new version of the file is for Theis purposes
% on March 29, 2002
% This is the ZOH equivalent of reduced linear model of the 737 Aircraft.
% The original model has 10 state variables
% and seven inputs. The reduced one has only 4 states (the first 4),
% two inpunt (first and third) and one output (pitch angle).
% Complete Linear model of the plant
Act_=[-3.311550e-02,  1.255350e-01, -1.403210e+01, -3.209610e+01,
       2.842170e-14,       0      ,     0        ,  5.684340e-14,
      -2.842170e-14,   7.390670e-05;
      -2.796460e-01, -6.347080e-01,  2.019550e+02, -2.230080e+00,
          0        ,      0       ,     0        , -4.547470e-13,
          0        ,  9.832660e-04;
      -3.260460e-04, -5.764990e-03, -4.880340e-01,
      -7.112450e-12, -1.896050e-15,      0       ,      0       ,
          0        ,  2.040460e-15,  2.317750e-06;
       4.775510e-08,  3.318080e-09,      1       ,  1.147350e-14,
           0       ,      0       ,      0       ,       0      ,
       5.421010e-20, -4.641240e-13;
       2.896600e-15,       0      ,       0      ,      0       ,
```

```
      -1.342080e-01, -2.011500e+02,  1.581050e+01,  3.077590e+01,
      -6.102970e-14,       0      ;
      -6.422490e-17,       0      ,       0      ,
      -9.005580e-18,  2.998050e-03, -1.374070e-01, -1.358770e-01,
      -3.836690e-03,  1.363330e-15,       0      ;
       3.654310e-16,       0      ,       0      ,
       7.949950e-19, -1.678690e-02,  8.075590e-01, -1.448390e+00,
       3.386950e-04, -7.633660e-15,       0      ;
       7.623300e-22,  8.470330e-22,       0      ,
      -7.623300e-19, -4.798570e-08,  6.948120e-02,     1        ,
       6.459310e-07,  1.694070e-21, -1.185850e-22 ;
       3.325460e-08,  2.310570e-09,       0      ,
       7.988400e-15,  1.059770e-08,  1.002410e+00,     0        ,
      -1.426540e-07,  2.704020e-06,   -3.231960e-13 ;
       6.931410e-02, -9.975950e-01,       0      ,
       2.025330e+02,       0      ,       0      ,
           0        ,       0      ,       0      ,     0 ];
Bct_=[ 7.571070e-04 -2.062720e-02 9.933170e-03
      -1.784430e-02 0 0 -1.784430e-02;
      -1.011500e-06 2.968750e-01 -1.429620e-01
       1.532630e-01 0 0 1.532630e-01;
       1.252950e-05 3.894960e-02 -1.877260e-02
       1.917180e-03 0 0 1.917180e-03;
       0 0 0 0 0 0 0;
```

```
        0 0 0 -2.242710e-02 4.041170e-04 1.227080e-01 2.242710e-02;
        0 0 0 2.421490e-03 1.352600e-03 -9.631680e-03 -2.421490e-03;
        0 0 0 1.753290e-02 1.456350e-02 8.190990e-03 -1.753290e-02;
        0 0 0 0 0 0 0 ;
        0 0 0 0 0 0 0 ;
        0 0 0 0 0 0 0 ];
     Cct_=[eye(4),zeros(4,6)];
     Dct_=[zeros(4,7)];
% Choped part
    [Act,Bct,Cct,Dct]=ssselect(Act_,Bct_,Cct_,Dct_,[1,3],[1,2,3,4],[1,2,3,4]);
sysCTPLANT=ss(Act,Bct,Cct,Dct);
% Nominal Close loop poles on CT
    CTpoles=[-0.014-0.18i;-0.014+0.18i;-1-1.1i;-1+1.1i]
%Nominal Dicrete Time Open Loop Plant's Space State Equations
   T=1/250;
   sysDTPLANT=c2d(sysCTPLANT,T,'ZOH');
   A=sysDTPLANT.A;
   B=sysDTPLANT.B;
   C=sysDTPLANT.C;
   D=sysDTPLANT.D;
% Nominal Pole placement gain in DT
F=place(A,B,exp(CTpoles*T));
% Augmented closed loop system including observer (A0)
    Lpfactor=8;
```

```
    Lp=acker(A',C',exp(CTpoles*T*Lpfactor))';
% Radiation Parameters
    D_lambda=100;
D_mu=10;
   p0star=0;
p1star=1;
 % Fixed Parts of A0
   n=size(A,1);
       qRB=1;
       A011= [A      , -B*F         ;
 Lp*C    , A-B*F-Lp*C ];
 %Variable parts of A0
   A012= [zeros(2*n,qRB*n)];
   A021= [zeros(qRB*n,n)];
   A022= [eye(qRB*n) zeros(qRB*n,n)];
   A0=[A011 A012; A021, A022];
 %Variable parts of A3
       A311= [A         ,   0*eye(n)   ,   -B*F , zeros(n,(qRB-1)*n)];
   A312= [zeros(n,(qRB+1)*n),eye(n) ];
   A321= [zeros(qRB*n,2*n)];
     A322= [eye(qRB*n)];
       A3worst=[A311 ;A312; A321, A322];
test_name=strcat(' B737, NM Rollback (',date,'): T = ',num2str(T),...
   ', D_{\mu} = ',num2str(D_mu),', 1/\mu = ',num2str(D_mu*T),...
```

```
    ', Lp_{',num2str(Lpfactor),'}. \newline Continues time poles: P1= '...
    ,num2str(CTpoles(1)),', P2= ',num2str(CTpoles(2)),', P3= '...
    ,num2str(CTpoles(3)),', P4= ',num2str(CTpoles(4)));
x0=[0 pi/200 0 0 0 0 0 0  zeros(1,n*qRB)]';
```

A.2.4 B737 Longitudinal Landing Model: PS Rollback

```
% B737_NM_3of4  Data file
% Created by Arturo Tejada
% original July 11, 2001. This new version of the file is for
% Theis purposes on March 29, 2002
% This is the ZOH equivalent of reduced linear model of the 737 Aircraft.
% The original model has 10 state variables
% and seven inputs. The reduced one has only 4 states (the first 4),
% two inpunt (first and third) and one output
% (pitch angle).
% Complete Linear model of the plant
Act_=[-3.311550e-02,  1.255350e-01, -1.403210e+01, -3.209610e+01,
       2.842170e-14,       0      ,     0        ,  5.684340e-14,
      -2.842170e-14,   7.390670e-05;
      -2.796460e-01, -6.347080e-01,  2.019550e+02, -2.230080e+00,
          0        ,      0       ,     0        , -4.547470e-13,
          0        ,  9.832660e-04;
      -3.260460e-04, -5.764990e-03, -4.880340e-01,
      -7.112450e-12, -1.896050e-15,      0       ,      0       ,
          0        ,  2.040460e-15,  2.317750e-06;
       4.775510e-08,  3.318080e-09,      1       ,  1.147350e-14,
           0       ,      0       ,      0       ,       0      ,
       5.421010e-20, -4.641240e-13;
       2.896600e-15,       0      ,       0      ,      0       ,
```

```
      -1.342080e-01, -2.011500e+02,  1.581050e+01,  3.077590e+01,
      -6.102970e-14,       0      ;
      -6.422490e-17,       0      ,       0      ,
      -9.005580e-18,  2.998050e-03, -1.374070e-01, -1.358770e-01,
      -3.836690e-03,  1.363330e-15,       0      ;
       3.654310e-16,       0      ,       0      ,
       7.949950e-19, -1.678690e-02,  8.075590e-01, -1.448390e+00,
       3.386950e-04, -7.633660e-15,       0       ;
       7.623300e-22,  8.470330e-22,       0      ,
      -7.623300e-19, -4.798570e-08,  6.948120e-02,     1        ,
       6.459310e-07,  1.694070e-21, -1.185850e-22 ;
       3.325460e-08,  2.310570e-09,       0      ,
       7.988400e-15,  1.059770e-08,  1.002410e+00,     0        ,
      -1.426540e-07,  2.704020e-06,   -3.231960e-13 ;
       6.931410e-02, -9.975950e-01,       0      ,
       2.025330e+02,       0      ,       0      ,
           0        ,       0      ,       0      ,     0 ];
Bct_=[ 7.571070e-04 -2.062720e-02 9.933170e-03
      -1.784430e-02 0 0 -1.784430e-02;
      -1.011500e-06 2.968750e-01 -1.429620e-01
       1.532630e-01 0 0 1.532630e-01;
       1.252950e-05 3.894960e-02 -1.877260e-02
       1.917180e-03 0 0 1.917180e-03;
       0 0 0 0 0 0 0;
```

```
        0 0 0 -2.242710e-02 4.041170e-04 1.227080e-01 2.242710e-02;
        0 0 0 2.421490e-03 1.352600e-03 -9.631680e-03 -2.421490e-03;
        0 0 0 1.753290e-02 1.456350e-02 8.190990e-03 -1.753290e-02;
        0 0 0 0 0 0 0 ;
        0 0 0 0 0 0 0 ;
        0 0 0 0 0 0 0 ];
      Cct_=[eye(4),zeros(4,6)];
      Dct_=[zeros(4,7)];
% Choped part
    [Act,Bct,Cct,Dct]=ssselect(Act_,Bct_,Cct_,Dct_,[1,3],[1,2,3,4],[1,2,3,4]);
sysCTPLANT=ss(Act,Bct,Cct,Dct);
% Nominal Close loop poles on CT
    CTpoles=[-0.014-0.18i;-0.014+0.18i;-1-1.1i;-1+1.1i]
%Nominal Dicrete Time Open Loop Plant's Space State Equations
   T=1/250;
   sysDTPLANT=c2d(sysCTPLANT,T,'ZOH');
   A=sysDTPLANT.A;
   B=sysDTPLANT.B;
   C=sysDTPLANT.C;
   D=sysDTPLANT.D;
% Nominal Pole placement gain in DT
F=place(A,B,exp(CTpoles*T));
% Augmented closed loop system including observer (A0)
    Lpfactor=8;
```

```
    Lp=acker(A',C',exp(CTpoles*T*Lpfactor))';
% Radiation Parameters
    D_lambda=50
D_mu=10;
   p0star=0;
p1star=1;
 % Fixed Parts of A0
   n=size(A,1);
       qRB=1;
       A011= [A      , -B*F         ;
 Lp*C    , A-B*F-Lp*C ];
 %Variable parts of A0
        Vstate=1;
        A022aux=[];
        A022aux3=ones(1,n);
        A022aux3(1,Vstate)=0;
        for A022aux2=2:qRB
            A022aux=[A022aux A022aux3];
        end
        A022aux=diag(A022aux);
      A012= [zeros(2*n,qRB*n)];
    A021= [zeros(qRB*n,n)];
    A022= [blkdiag(diag(A022aux3),A022aux) zeros(qRB*n,n)];
    A0=[A011 A012; A021, A022];
```

```
%Variable parts of A3

        A311= [A        ,   0*eye(n)   ,   -B*F , zeros(n,(qRB-1)*n)];

    A312= [zeros(n,(qRB+1)*n),diag(A022aux3)];

    A321= [zeros(qRB*n,2*n)];

      A322= [blkdiag(diag(A022aux3),A022aux)];

        A3worst=[A311 ;A312; A321, A322];

test_name=strcat(' B737, NM Rollback (',date,'): T = ',num2str(T),...

   ', D_{\mu} = ',num2str(D_mu),', 1/\mu = ',num2str(D_mu*T),...

   ', Lp_{',num2str(Lpfactor),'}. \newline Continues time poles: P1= '...

   ,num2str(CTpoles(1)),', P2= ',num2str(CTpoles(2)),', P3= '...

   ,num2str(CTpoles(3)),', P4= ',num2str(CTpoles(4)));

x0=[0 pi/200 0 0 0 0 0 0  zeros(1,n*qRB)]';
```

A.2.5 B737 Longitudinal Landing Model: Reset/Cold-Restart

```
% B737_NM_RE  Data file
% Created by Arturo Tejada
% original July 11, 2001. This new version of the file is for
% Theis purposes on March 29, 2002
% This is the ZOH equivalent of reduced linear model of the
% 737 Aircraft. The original model has 10 state variables
% and seven inputs. The reduced one has only 4 states
% (the first 4), two inpunt (first and third) and one output
% (pitch angle).
% Complete Linear model of the plant
Act_=[-3.311550e-02,  1.255350e-01, -1.403210e+01, -3.209610e+01,
       2.842170e-14,       0      ,     0        ,  5.684340e-14,
      -2.842170e-14,   7.390670e-05;
      -2.796460e-01, -6.347080e-01,  2.019550e+02, -2.230080e+00,
          0        ,      0       ,     0        , -4.547470e-13,
          0        ,  9.832660e-04;
      -3.260460e-04, -5.764990e-03, -4.880340e-01,
      -7.112450e-12, -1.896050e-15,      0       ,      0       ,
          0        ,  2.040460e-15,  2.317750e-06;
       4.775510e-08,  3.318080e-09,      1       ,  1.147350e-14,
           0       ,      0       ,      0       ,       0      ,
       5.421010e-20, -4.641240e-13;
       2.896600e-15,       0      ,       0      ,      0       ,
```

```
      -1.342080e-01, -2.011500e+02,  1.581050e+01,  3.077590e+01,
      -6.102970e-14,       0      ;
      -6.422490e-17,       0      ,       0      ,
      -9.005580e-18,  2.998050e-03, -1.374070e-01, -1.358770e-01,
      -3.836690e-03,  1.363330e-15,       0      ;
       3.654310e-16,       0      ,       0      ,
       7.949950e-19, -1.678690e-02,  8.075590e-01, -1.448390e+00,
       3.386950e-04, -7.633660e-15,       0       ;
       7.623300e-22,  8.470330e-22,       0      ,
      -7.623300e-19, -4.798570e-08,  6.948120e-02,     1        ,
       6.459310e-07,  1.694070e-21, -1.185850e-22 ;
       3.325460e-08,  2.310570e-09,       0      ,
       7.988400e-15,  1.059770e-08,  1.002410e+00,     0        ,
      -1.426540e-07,  2.704020e-06,   -3.231960e-13 ;
       6.931410e-02, -9.975950e-01,       0      ,
       2.025330e+02,        0     ,       0      ,
           0       ,        0     ,       0      ,     0 ];
Bct_=[ 7.571070e-04 -2.062720e-02 9.933170e-03
      -1.784430e-02 0 0 -1.784430e-02;
      -1.011500e-06 2.968750e-01 -1.429620e-01
       1.532630e-01 0 0 1.532630e-01;
       1.252950e-05 3.894960e-02 -1.877260e-02
       1.917180e-03 0 0 1.917180e-03;
       0 0 0 0 0 0 0;
```

```
        0 0 0 -2.242710e-02 4.041170e-04 1.227080e-01 2.242710e-02;
        0 0 0 2.421490e-03 1.352600e-03 -9.631680e-03 -2.421490e-03;
        0 0 0 1.753290e-02 1.456350e-02 8.190990e-03 -1.753290e-02;
        0 0 0 0 0 0 0 ;
        0 0 0 0 0 0 0 ;
        0 0 0 0 0 0 0 ];
      Cct_=[eye(4),zeros(4,6)];
      Dct_=[zeros(4,7)];
 sysCTPLANT=ss(Act,Bct,Cct,Dct);
% Choped part
 [Act,Bct,Cct,Dct]=ssselect(Act_,Bct_,Cct_,Dct_,[1,3],[1,2,3,4],[1,2,3,4]);
 sysCTPLANT=ss(Act,Bct,Cct,Dct);
% Nominal Close loop poles on CT
    CTpoles=[-0.014-0.18i;-0.014+0.18i;-1-1.1i;-1+1.1i]
% Nominal Dicrete Time Open Loop Plant's Space State Equations
   T=1/250;
   sysDTPLANT=c2d(sysCTPLANT,T,'ZOH');
   A=sysDTPLANT.A;
   B=sysDTPLANT.B;
   C=sysDTPLANT.C;
   D=sysDTPLANT.D;
% Nominal Pole placement gain in DT
F=place(A,B,exp(CTpoles*T));
% Augmented closed loop system including observer (AO)
```

```
    Lpfactor=8;
    Lp=acker(A',C',exp(CTpoles*T*Lpfactor))';
% Radiation Parameters
    D_lambda=200;
D_mu=10;
   p0star=0;
p1star=1;
 % Fixed Parts of A0
   n=size(A,1);
       qRB=1;
       A0=   [A      , -B*F       , 0*eye(n) ;
 Lp*C    , A-B*F-Lp*C , 0*eye(n) ;
            0*eye(n), eye(n)     , 0*eye(n)];
 %Variable parts of A3
       A3worst= [A           ,   0*eye(n)  ,   -B*F;
                 zeros(n,3*n);
                 zeros(n,2*n),   eye(n)];
test_name=strcat(' B737, NM Reset (',date,'): T = ',num2str(T),...
   ', D_{\mu} = ',num2str(D_mu),', 1/\mu = ',num2str(D_mu*T),...
   ', Lp_{',num2str(Lpfactor),'}. \newline Continues time poles: P1= '...
   ,num2str(CTpoles(1)),', P2= ',num2str(CTpoles(2)),', P3= '...
   ,num2str(CTpoles(3)),', P4= ',num2str(CTpoles(4)));
x0=[0 pi/200 0 0 zeros(1,n*2)]';
```

A.2.6 AFTI/F-16 Longitudinal Model: FS Rollback

```
% ACC02_F16_NM2  ACC02 example but with a new RB A0 state equation
% This is the ZOH equivalent of the ACC 1999 F-16 example. T=1/250
% Created by  Arturo Tejada on the 02/12/02
% Nominal Continues time Plant Space State Model
Act=[-0.0507 -3.861      0 -32.17;
 -0.00117 -0.5164  1   0;
         -0.000129      1.4168 -0.4932       0;
          0  0  1   0];
Bct=[  0;
-0.0717;
            -1.645;
             0];
Cct=[ 0 0 1 0];
Dct=0;
sysCTPLANT=ss(Act,Bct,Cct,Dct);
% Nominal Close loop poles on CT
    CTpoles=roots(conv([1 2*1*0.2 1^2],[1 2*0.1*0.1 0.1^2]));
% Nominal Dicrete Time Open Loop Plant's Space State Equations
   T=1/250;
   sysDTPLANT=c2d(sysCTPLANT,T,'ZOH');
   A=sysDTPLANT.A;
   B=sysDTPLANT.B;
   C=sysDTPLANT.C;
```

```
   D=sysDTPLANT.D;
% Nominal Pole placement gain in DT
F=place(A,B,exp(CTpoles*T));
% Augmented closed loop system including observer (A0)
    Lpfactor=5;
    Lp=place(A',C',exp(CTpoles*T*Lpfactor))';
% Radiation Parameters
D_lambda=63;
D_mu=10;
   p0star=0;
p1star=1;
% Fixed Parts of A0
   n=size(A,1);
       qRB=8;
       A011= [A      , -B*F         ;
 Lp*C    , A-B*F-Lp*C ];
%Variable parts of A0
   A012= [zeros(2*n,qRB*n)];
   A021= [zeros(qRB*n,n)];
   A022= [eye(qRB*n) zeros(qRB*n,n)];
   A0=[A011 A012; A021, A022];
%Variable parts of A3
       A311= [A         ,   0*eye(n)   ,   -B*F , zeros(n,(qRB-1)*n)];
   A312= [zeros(n,(qRB+1)*n),eye(n) ];
```

```
   A321= [zeros(qRB*n,2*n)];

     A322= [eye(qRB*n)];

   A3worst=[A311 ;A312; A321, A322];

test_name=strcat(' ACC 02 : New Rollback Case.  T = '...

,num2str(T),', D_{\mu} = ',num2str(D_mu),', 1/\mu = ',num2str(D_mu*T),...

', Lp_{',num2str(Lpfactor),'}, q = ',num2str(qRB),...

'. \newline Continues time poles: P1= ',num2str(CTpoles(1)),...

', P2= ',num2str(CTpoles(2)),', P3= ',num2str(CTpoles(3)),', P4= '...

,num2str(CTpoles(4)));

x0=[0 pi/200 0 0 0 0 0 0  zeros(1,n*qRB)]';
```

A.2.7 AFTI/F-16 Longitudinal Model: PS Rollback

```
% ACC02_F16_NM2_3of4  ACC02 example but with a new RB A0 state equation
% This is the ZOH equivalent of the ACC 1999 F-16 example. T=1/250
% Created by  Arturo Tejada on the 02/12/02
% Nominal Continues time Plant Space State Model
Act=[-0.0507 -3.861      0 -32.17;
 -0.00117 -0.5164  1    0;
         -0.000129      1.4168 -0.4932       0;
          0  0  1   0];
Bct=[  0;
-0.0717;
            -1.645;
             0];
Cct=[ 0 0 1 0];
Dct=0;
sysCTPLANT=ss(Act,Bct,Cct,Dct);
% Nominal Close loop poles on CT
    CTpoles=roots(conv([1 2*1*0.2 1^2],[1 2*0.1*0.1 0.1^2]));
% Nominal Dicrete Time Open Loop Plant's Space State Equations
   T=1/250;
   sysDTPLANT=c2d(sysCTPLANT,T,'ZOH');
   A=sysDTPLANT.A;
   B=sysDTPLANT.B;
   C=sysDTPLANT.C;
```

```
   D=sysDTPLANT.D;
% Nominal Pole placement gain in DT
F=place(A,B,exp(CTpoles*T));
% Augmented closed loop system including observer (A0)
    Lpfactor=5;
    Lp=place(A',C',exp(CTpoles*T*Lpfactor))';
% Radiation Parameters
D_lambda=147;
D_mu=10;
   p0star=0;
p1star=1;
% Fixed Parts of A0
   n=size(A,1);
       qRB=10;
       A011= [A      , -B*F         ;
 Lp*C    , A-B*F-Lp*C ];
% Special Identity matrix
        vstate=2;         %%% This is the voided state
        mdiag=ones(1,size(A,1));
        mdiag(vstate)=0;
        In=diag(repmat(mdiag,1,qRB-1));  %% This creates the main diagonal
%Variable parts of A0
   A012= [zeros(2*n,qRB*n)];
   A021= [zeros(qRB*n,n)];
```

```
   A022= [blkdiag(eye(n),In)  zeros(qRB*n,n)];

   A0=[A011 A012; A021, A022];

%Variable parts of A3

       A311= [A        ,   0*eye(n)  ,   -B*F , zeros(n,(qRB-1)*n)];

   A312= [zeros(n,(qRB+1)*n),diag(mdiag) ];

   A321= [zeros(qRB*n,2*n)];

     A322= [blkdiag(eye(n),In)];

   A3worst=[A311 ;A312; A321, A322];

test_name=strcat(' ACC 02 : New PSRB',num2str(vstate),' V2 Case.  T = '...

,num2str(T),', D_{\mu} = ',num2str(D_mu),', 1/\mu = ',num2str(D_mu*T),...

', Lp_{',num2str(Lpfactor),'}, q = ',num2str(qRB),...

'. \newline Continues time poles: P1= ',num2str(CTpoles(1)),...

', P2= ',num2str(CTpoles(2)),', P3= ',num2str(CTpoles(3)),...

', P4= ',num2str(CTpoles(4)));

x0=[0 pi/200 0 0 0 0 0 0  zeros(1,n*qRB)]';
```

A.3 The Stability Threshold Calculator Code

```
% TB_genV3  Main program to run a complete simulation
% SYNTAX: TB_genV2(ExampleFileName,[Dlmin Dlstep Dlmax])
%  EXAMPLES:
% TB_genV3('cdc2Kex1b')  %Assumes Dlmin=10. Dlstep=10, Dlmax=800
% TB_genV3('example1',[10 1 100])
% where:
%     ExampleFileName = name of .m script that defines ALL the variables
%      Dlmin: Min Dlambda value
%     Dlstep: Step increments between Dlmin and Dlmax
% Dlmax: Max Dlambda value
% Notice that the program doesn't chek for inconsistencies in Dlmin, Dlmax,
% and Dlstep
% Code By Arturo Tejada , Vesion 3
% Date: June 3, 2001
% The main change is that the code will be modified to use the same
% example file that the rest of the simulation code uses normally
% Version 3 change is that the A1 eigenvalues are now calculated in a
% faster way : eig([e11*AK0,e21*AK3;e12*AK0,e22*Ak3]
function [sDl,sRad]=TB_genV3(examplefname,DLarg)
eval(examplefname);
% Check to be safe
if nargin>=3
   fprintf('Too many arguments');
```

```
end
if nargin==0
   fprintf('You need to include the Example File Name');
end
if nargin==2
Dll=DLarg(1):DLarg(2):DLarg(3);
   onespan=DLarg(1)*T:DLarg(2)*T*0.1:DLarg(3)*T;
   onesplot=ones(1,length(onespan));
else
Dll=10:10:80;
   onespan=10:1:80;
   onespan=onespan*T;
   onesplot=ones(1,length(onespan));
end
if (rank(obsv(A,C))~=size(A,1))
   'Plant Not Observable'
elseif (max(eig(A-B*F))>1)
   'Unstable nominal Closed-Loop  System'
else
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %Now, the real Sweep Begins
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % SWEEP PARAMETERS
   mu=1/(D_mu*T);
```

```
la=1./(Dll*T);
% Script A1
  dlvec=[];
savec=[];
      KA0=kron(A0,A0);
      KA3=kron(A3worst,A3worst);
      t_sim=cputime;
   for j=1:length(Dll)
         fprintf('Calculation for Dl = %d',Dll(j));
         tic
         e11=(mu+la(j)*exp(-T*la(j)-T*mu))/(mu+la(j));
e12=-la(j)*(exp(-T*la(j)-T*mu)-1)/(mu+la(j));
e21=-mu*(exp(-T*la(j)-T*mu)-1)/(mu+la(j));
         e22=(la(j)+mu*exp(-T*la(j)-T*mu))/(mu+la(j));
         sA=[e11*KA0,e21*KA3;
             e12*KA0,e22*KA3];
         savec(j)=max(abs(eig(sA)));
         fprintf(', took %3.2f secs\n\d',toc );
   end
   fprintf('Total Calculation time: %3.2f secs \n \d' ...
       ... ,cputime-t_sim);
   figure
   hold on;
plot(Dll,savec,'linewidth',2);
```

```
        plot(onespan/T,onesplot,'linewidth',1);

    xlabel('D_{\lambda}');

        ylabel ('\rho_{\sl{A_1}}=1');

        title(strcat('\rho_{\sl{A_1}}=1 vs D_{\lambda} for'...

        ... ,test_name),'fontsize',11);

        hold off;

        sRad=savec;

        sDl=Dll;

end
```

A.4 The Critical Boundary Calculator Code

```
% TB_gen3DV1  Main program to run a complete simulation
% SYNTAX:
%  TB_gen3DV1(ExampleFileName,[Dlmin Dlstep Dlmax],[Dumin,Dlstep,Dlmax])
%
%  EXAMPLES:
% TB_gen3DV1('cdc2Kex1b')
%                          %Runs the simulation and assumes Dlmin=10.
%                          %Dlstep=10, Dlmax=800
% where:
% ExampleFileName = name of .m script that defines ALL the variables
% Dlmin: Min Dlambda value
% Dlstep: Step increments between Dlmin and Dlmax
% Dlmax: Max Dlambda value
% Notice that the program doesn't chek for inconsistencies in Dlmin, Dlmax,
% and Dlstep
% Code By Arturo Tejada , Vesion 1
% Date: July 23, 2001
% Version 1  A1's eigenvalues are now calculated in a faster way
% eig([e11*AK0,e21*AK3;e12*AK0,e22*Ak3]
function [sDl,sDu,sRad]=TB_genV3(examplefname,DLarg,DUarg)
eval(examplefname);
% Check to be safe
if nargin>=4
```

```
   fprintf('Too many arguments');
end
if nargin==0
   fprintf('You need to include the Example File Name');
end
if nargin>=2
   [Duu,Dll]=meshgrid(DUarg(1):DUarg(2):DUarg(3),DLarg(1):DLarg(2):DLarg(3));
elseif nargin==2
   [Duu,Dll]=meshgrid(15:1:20,DLarg(1):DLarg(2):DLarg(3));
else
   [Duu,Dll]=meshgrid(15:1:20,10:10:80);
end
if (rank(obsv(A,C))~=size(A,1))
   'Plant Not Observable'
elseif (max(eig(A-B*F))>1)
   'Unstable nominal Closed-Loop  System'
else
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %Now, the real Sweep Begins
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % SWEEP PARAMETERS
   % Script A1
         mu=1./(Duu*T);
         la=1./(Dll*T);
```

```
    dlvec=[];
savec=[];
        KA0=kron(A0,A0);
        KA3=kron(A3worst,A3worst);
        t_sim=cputime;
  for j=1:size(Duu,2)
       for k=1:size(Dll,1)
       fprintf('Calculation for Du= %d , Dl = %d',Duu(k,j), Dll(k,j));
       tic
       e11=(mu(k,j)+la(k,j)*exp(-T*la(k,j)-T*mu(k,j)))/(mu(k,j)+la(k,j));
  e12=-la(k,j)*(exp(-T*la(k,j)-T*mu(k,j))-1)/(mu(k,j)+la(k,j));
  e21=-mu(k,j)*(exp(-T*la(k,j)-T*mu(k,j))-1)/(mu(k,j)+la(k,j));
       e22=(la(k,j)+mu(k,j)*exp(-T*la(k,j)-T*mu(k,j)))/(mu(k,j)+la(k,j));
       sA=[e11*KA0,e21*KA3;
           e12*KA0,e22*KA3];
       savec(k,j)=max(abs(eig(sA)));
       fprintf(', took %3.2f secs\n\d',toc );
       end
   end
     fprintf('Total Calculation time: %3.2f secs \n \d',cputime-t_sim);
     figure
 hidden off
     mesh(T*Dll,T*Duu,savec);
     hold on;
```

```
  hidden on;
      mesh(T*[DLarg(1),DLarg(1);DLarg(3),DLarg(3)],...
          ... T*[DUarg(1),DUarg(3);DUarg(1),DUarg(3)],ones(2,2));
  xlabel('1/\lambda');
      ylabel('1/\mu');
      zlabel ('\rho_{\sl{A_1}}=1');
     hold off;
      figure;
      contour(T*Duu,T*Dll,savec,[1 1],'b');
      ylabel('1/\lambda');
      xlabel('1/\mu');
      sRad=savec;
      sDl=Dll;
      sDu=Duu;
end
```

CURRICULUM VITA
for
ARTURO TEJADA RUIZ

DEGREES:

- Bachelor of Science (Electrical Engineering), Pontificia Universidad Católica del Perú, Lima, Perú, December 1996.

PROFESSIONAL CHRONOLOGY:

- Department of Electrical and Electronic Engineering, Pontificia Universidad Católica del Perú

  Part Time Professor, March 1997 - May 2000.

HONORS AND AWARDS:

- Best Master Student of the Year, IEEE and ECE Department of Old Dominion University, 2002.
- Best Design on the Project Contest of the Affine Branches, Peruvian National Congress on Mechanical and Electrical Engineering and Affine Branches, CONIMERA'99, 1999.

SCHOLARSHIPS AWARDED:

- Presidential Dominion Scholar Fellowship, Old Dominion University, 2000-2002

SCHOLARLY ACTIVITIES COMPLETED:

- O. R. González, A. Tejada and W. S. Gray 'Analysis of Design Trade-Offs in the Rollback Recovery Method for Fault Tolerant Digital Control Systems,' *Proc. 2002 American Control Conference*, Anchorage, Alaska, pp. 4801-4806, 2002.
- O. R. González, W. S. Gray, A. Tejada and S. Patilkulkarni, 'Stability Analysis of Electromagnetic Interference Upset Recovery Methods,' *Proc. 40th IEEE Conference on Decision and Control*, Orlando, Florida, pp. 4134-4139, 2001.
- O. R. González, W. S. Gray, A. Tejada and S. Patilkulkarni, 'Stability Analysis of Upset Recovery Methods for Electromagnetic Interference,' *Proc. 20th DASC Digital Avionics Systems Conference*, Daytona Beach, Florida, 2001.
- O. R. Gonzalez, W. S. Gray and A. Tejada, 'Analytical Tools for the Design and Verification of Safety Critical Control Systems,' *Proc. of the 2001 International Conference on Lightning and Static Electricity*, Seattle, Washington, 2001.

DATE DUE
