Summer 2004

Stability Analysis of Jump-Linear Systems Driven by Finite-State Machines with Markovian Inputs

Sudarshan S. Patilkulkarni
Old Dominion University

STABILITY ANALYSIS OF JUMP-LINEAR SYSTEMS DRIVEN BY FINITE-STATE MACHINES WITH MARKOVIAN INPUTS

by

Sudarshan S. Patilkulkarni
B.E. E.C. June 1996, Karnatak University, Dharwad, India
M.E. E.E. December 2000, Old Dominion University, Norfolk, Virginia

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

DOCTOR OF PHILOSOPHY

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
August 2004

Approved by:

_____

W. Steven Gray (Director)

_____

Oscar R. González (Member)

_____

Linda L. Vahala (Member)

_____

Dayanand N. Naik (Member)

ABSTRACT

STABILITY ANALYSIS OF JUMP-LINEAR SYSTEMS
DRIVEN BY
FINITE-STATE MACHINES WITH MARKOVIAN INPUTS

Sudarshan S. Patilkulkarni
Old Dominion University, 2004
Director: Dr. W. Steven Gray

A control system with a fault recovery mechanism in the feedback loop and with faults occurring in a non-deterministic manner can be modeled as a class of hybrid systems, i.e., a dynamical system switched by a finite-state machine or an automaton. When the plant and controller are linear, such a system can be modeled as a jump-linear system driven by a finite-state machine with a random input process. Such fault recovery mechanisms are found in flight control systems and distributed control systems with communication networks. In these critical applications, closed-loop stability of the system in the presence of fault recoveries becomes an important issue.

Finite-state machines as mathematical constructs are widely used by computer scientists to model and analyze algorithms. In particular, fault recovery mechanisms that are implemented in hardware with logic based circuits and finite memory can be modeled appropriately with finite-state machines. In this thesis, mathematical tools are developed to determine the mean-square stability of a closed-loop system, modeled as a jump-linear system in series with a finite-state machine driven by a random process. The random input process is in general assumed to be any r-th order Markov process, where $r \geq 0$. While stability tests for a jump-linear system with a

Markovian switching rule are well known, the main contribution of the present work arises from the fact that output of a finite-state machine driven by a Markov process is in general *not* Markovian. Therefore, new stability analysis tools are provided for this class of systems and demonstrated through Monte Carlo simulations.

"The whole is more than the sum of its parts" - constitutive characteristics of a phenomenon are not explainable from the characteristics of the isolated parts alone, it is also the interconnection and the interrelationship. -Ludwig Von Bertalanffy, *General System Theory.*

ACKNOWLEDGMENTS

Setting and achieving a higher academic goal is insurmountable without adequate moral, emotional and financial support. Supervisors, coworkers, friends, family and institutions make us what we are.

Dr. W. Steven Gray, my advisor, whose immense patience, support, wisdom and advice is the foremost contribution in my achievement. Dr. Oscar R. Gonzalez's subtle comments during our many discussions have always been an eye-opener. Dr. Dayanand Naik, Dr. Linda Vahala, Dr. Sacharia Albin and Dr. Steven Zahorian have all provided me valuable assistance along the way.

I am indebted to Prof. Anand Kumar and Prof. B. N. Devaraj, my mentors during my undergraduate years at S.D.M. College of Engineering in Dharwad, India. They instilled in me a passion for electrical engineering, especially in the area of signals and systems.

I am also grateful for the support of my co-workers: Adeel Jafri, James Barkley, Alicia Hoffler, Mark Adams, Hong Zhang, Li Yaqin and Heber Zapana-Herencia. Special thanks to Arturo Tejada who has provided me with coffee, food and lodging along with the moral support whenever I was in need.

Old Dominion University has provided me with valuable course-work, a good library and a supportive academic environment.

This research was supported by the NASA Langley Research Center under grants NCC-1-392, NCC-1-03026 and NNL04AA03A, and by the National Science Foundation under grant CCR-0209094. I wish to acknowledge the support of these organi-

zations.

Finally, the love and warmth of my family: Vinayak, Vani, Vallabh, Vasuki, father and belated mother have been my guide. Moral and emotional support of Preethi and many other friends whom I have met during the course of my graduate studies, whose names I am unable to list here due to lack of space, cannot be left unacknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation for this Research

Fault tolerance and recovery is a wide area of research in computer engineering, communications and controls [45]. System stability is always a critical concern in control systems equipped with fault tolerance and error recovery mechanisms [1, 52, 58]. The main motivation for the research presented in this thesis is the need to perform stability analysis of hybrid models for closed-loop digital control systems implementing an error recovery algorithm as the fault tolerant technique and subjected to multiple upsets [16]. The mathematical model in this case is a discrete-time, jump-linear system driven by a finite-state machine with Markovian inputs shown in Figure 1.1.

Figure 1.1: The hybrid model under consideration.

A more precise mathematical description of the model is as follows. A Markov process $\boldsymbol{\nu}$ takes on symbols from the set $\Sigma_I = \{\eta_1, \eta_2, \ldots, \eta_M\}$ according to a probability transition matrix Π_I, by convention the column sums of Π_I are assumed to be unity.

The Markov process $\boldsymbol{\nu}$ in turn drives the finite-state machine $\mathcal{M} = \{\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega\}$. The state of the machine, $\boldsymbol{z}(k)$, takes on values from the set $\Sigma_S = \{e_1, e_2, \ldots, e_N\}$, which is simply the collection of elementary vectors $e_j = [0 \cdots 0 \underbrace{1}_{j\text{-th position}} 0 \cdots 0]^T$. The next state function $\delta : \Sigma_I \times \Sigma_S \mapsto \Sigma_S$ is a mapping of the form

$$\boldsymbol{z}(k+1) = S_{\boldsymbol{\nu}(k)} \boldsymbol{z}(k),$$

where each $N \times N$ matrix S_η, $\eta \in \Sigma_I$, is a deterministic transition matrix, i.e., a matrix where each column contains exactly a single one and $N-1$ zeros. The output function $\omega : \Sigma_I \times \Sigma_S \mapsto \Sigma_O$ is uniquely specified by

$$\omega(\eta_l, e_j) = \xi_{i_j}, \quad j = 1, \ldots, N.$$

It assigns to each pair of input and state symbol in $\Sigma_I \times \Sigma_S$ an output symbol from the set $\Sigma_O = \{\xi_1, \xi_2, \ldots, \xi_L\}$. It can be written in the form

$$\boldsymbol{c}(k) = T_{\boldsymbol{\nu}(k)} \boldsymbol{z}(k),$$

where $\boldsymbol{c}(k)$ is a Boolean vector corresponding to $\boldsymbol{\theta}(k) = \xi_{i_j}$, i.e. a vector with exactly a single one at $i_j - th$ position and $L-1$ zeros. Here $T_{\boldsymbol{\nu}(k)}$ is a $L \times N$ matrix with exactly a single one and $L-1$ zeros in each column. Finally, the output from the machine, $\boldsymbol{\theta}(k) = \omega(\boldsymbol{\nu}(k), \boldsymbol{z}(k))$, is used to drive an n-dimensional jump-linear dynamical system

$$\boldsymbol{x}(k+1) = A_{\boldsymbol{\theta}(k)} \boldsymbol{x}(k), \tag{1.1}$$

where $A_\xi \in \mathbb{R}^{n \times n}$, $\|A_\xi\| < \infty$ with $\xi \in \Sigma_O$. For conciseness, this hybrid model consisting of a jump-linear system driven by the output process $\boldsymbol{\theta}$ of a finite-state

machine $\mathcal{M}$ with Markovian input $\boldsymbol{\nu}$ will be denoted throughout this thesis by the notation $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$. Although this hybrid model is applicable to a wide variety of applications, two particularly motivating applications of interest will be described next. The first application is a networked control system equipped with a recovery mechanism for packet loss. The second example is a digital flight controller equipped with a recovery system to provide reliable operation in harsh environments where electromagnetic or particle radiation may be present. Both examples illustrate situations where the stability of the closed-loop system is a critical issue.

1.1.1 Networked Control Systems with Packet Dropout and Recovery

Networked control systems (NCS) are control systems with a communication network in the feedback loop. They have found applications in unmanned aerial vehicle control systems, automated highway systems and many other distributed remote control applications [49]. In a NCS, a communication network can exist between the plant and the controller communicating a sensor signal and between the controller and the plant communicating a control signal. The output from the plant sensor gets encoded by a quantization, compression and coding technique into packets which are transmitted to the controller. But before the controller can process the data, it must be decoded to recover the sensor signal. In this process, there can be quantization error and packet loss due to channel congestion. A similar situation exists for the control signal being sent between the controller and the plant. Stability of such closed-loop

systems, assuming a deterministic rate of packet losses, have been studied by various researchers either by modeling them as asynchronous dynamical systems [19] or as linearized hybrid systems [58].

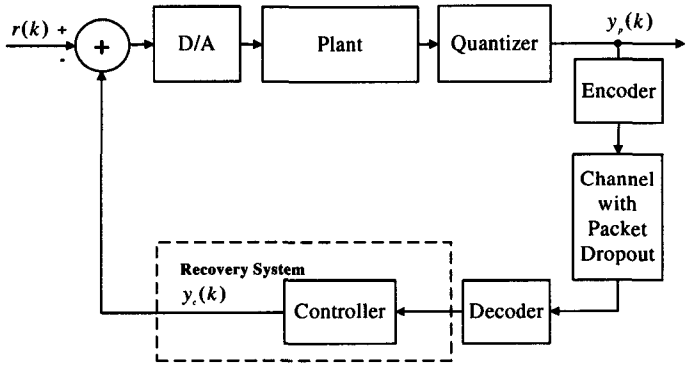

Figure 1.2: Block diagram of a networked control system with packet dropout and recovery in the feedback path.

Suppose a network exists only between the plant sending the sensor signal and the controller as shown in Figure 1.1.1. Assume packets containing an encoded form of the sensor signal are being dropped and the drop process is Markovian in nature. System stability in this case has been analyzed by introducing a Markovian jump-linear model [49, 56]. In [33] an optimal controller was designed to compensate for the packet losses under similar assumptions. Suppose the last known good packet is recovered from a buffer, using some typical recovery techniques like those in [25]. A simple example of one such recovery technique is shown in Figure 1.1.1. Here the packet drop process is modeled by the process ν, where $\nu(k) = A$ denotes no loss of packets (absence) and $\nu(k) = E$ denotes a packet loss (existence). As long as there is no loss of packets, the nominal control law is in operation and the closed-system

system dynamics $\boldsymbol{x}(k)$ are governed by the transition matrix A_0. However, if there is a packet drop, the system enters a recovery mode, and a new control law comes into operation. In this situation, closed-loop system dynamics $\boldsymbol{x}(k)$ are now governed by the transition matrix A_1. Once there are no further packet losses, the system resumes its nominal operation. Thus the closed-loop system switches back and forth between two system parameters A_0 and A_1 based on the packet loss process $\boldsymbol{\nu}$ being under no loss ($\boldsymbol{A}$) or loss ($\boldsymbol{E}$). Such a system behavior is known to be a jump-linear system. It is possible that although system parameter A_0 is always stable during normal operation, the system parameter A_1 is more likely to be unstable during the packet loss and recovery. Hence, the knowledge of stability conditions for this system becomes critical. In case of a more complex recovery technique and Markovian characteristics for the packet loss process $\boldsymbol{\nu}$, the overall system behavior can be modeled as a jump-linear system driven by the finite-state machine with Markovian inputs, the one shown in Figure 1.1, stability analysis problem for which this thesis is specifically concerned.

1.1.2 Digital Flight Control Systems with Upset and Recovery

Digital fly-by-wire aircraft with safety critical computer systems are required to operate reliably in harsh environments. The most commonly used fault-tolerant computing techniques like triple modular redundancy, error correcting codes and others can help achieve this. These fault-tolerant systems, however, may not be suit-

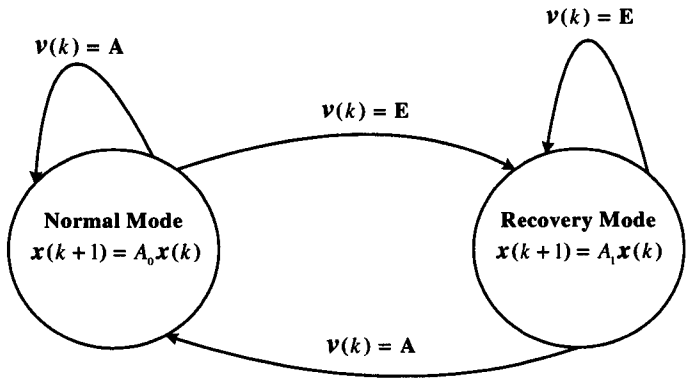

Figure 1.3: A simple example illustrating recovery of packet loss in a NCS.

able to handle correlated or common-mode faults like those produced for example in electromagnetic environments [18, 40]. The design, validation, and verification of fault-tolerant systems that are subjected to common-mode faults is an active area of research [2, 22, 32]. Recently, more sophisticated techniques for error recovery using rollback and roll forward have been proposed in the fault tolerant community [45]. A technique that is being investigated to address transient or soft common-mode fault is error recovery with multiple dual-lock-step processors together with new fault tolerant architectures and communication subsystems [21, 22]. An example is NASA Langley Research Center's Recoverable Computer System (RCS), which is being used to study recovery from transient faults introduced by high intensity electromagnetic radiation [35, 36] and atmospheric neutrons [17]. As depicted in Figure 1.5, two microprocessors with their own memory, expected to be carrying the same data are

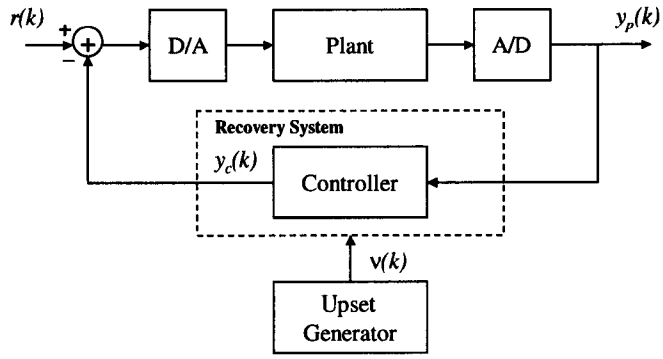

Figure 1.4: Block diagram of a closed-loop digital flight-control system with error recovery in the feedback path.

executed and compared at every clock cycle. The error recovery technique imple-

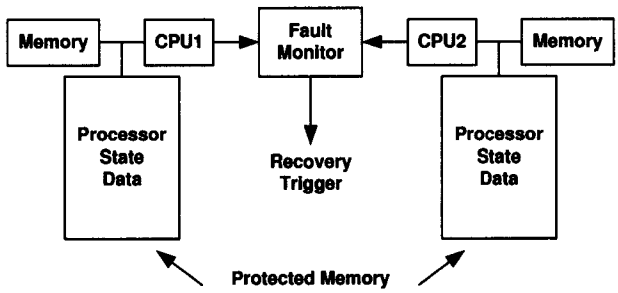

Figure 1.5: Architecture of duplex fault control system

mented on the RCS is a variation of rollback recovery; it has the following steps: checkpointing, fault-tolerant comparison, rollback, and retry [43,46]. During a checkpoint, the state of each microprocessor module is stored. When an upset is detected, rollback of both microprocessor modules to a previous checkpoint takes place, and then the system is allowed to proceed with normal execution. But once the execution of the normal control program is interrupted, the execution of a different control law

takes place, one that has significantly different dynamics and is on a time scale that can alter the overall closed-loop dynamics of the flight control system. A stability analysis procedure for such a closed-loop system has been presented in [13–15,52] under simplifying assumptions such as no new upset can occur during an active recovery process. This model is accurate for low radiation levels, but begins to breakdown as the upset probability increases. The essential limitation in this existing approach is that the jump-linear model employed does not permit complex recovery algorithms to be easily encoded into the model's structure.

The new class of hybrid model shown in Figure 1.1 and introduced in [16] can capture the essential behavior of a closed-loop digital control system implemented on an RCS to handle multiple upsets. The finite-state Markov process is used to characterize the upset generator which is supplying the random external disturbances to the control system. Typically, from the recovery system's point of view, there are only two possible states: an upset is absent (A) or one exists (E). Upset statistics are completely determined by the transition probabilities $\Pi_{A|A}$ and $\Pi_{E|E}$, which are assumed to be known. In general, the upset generator can have more than two states.

The finite-state machine models the recovery algorithms. Figure 1.6 shows a four state recovery algorithm which is particularly well suited for persistent upset conditions. As long as an upset is absent the controller stays in the Normal (No) Mode and executes the nominal control law which produces the closed-loop dynamical system (A_{No}, B_{No}, C_{No}). But once an upset condition is detected, a rollback procedure is engaged. In the Reload (Rd) Mode the current value of the control is frozen in time and the system remains in this mode for N_{Rd} time samples to model the

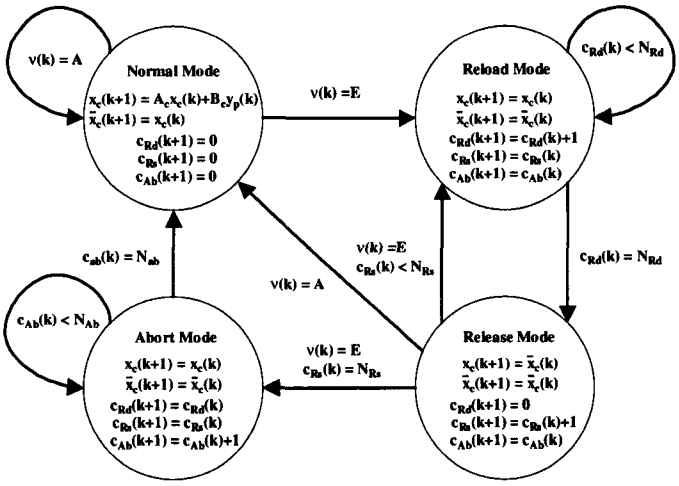

Figure 1.6: A rollback recovery algorithm to handle persistent upsets.

delay encountered while the computer memory is reloaded with the last known good values stored in the checkpointing memory $\bar{x}_c(k)$. Next, the system proceeds to the Release (Rs) Mode, where the last known good state values are actually released into the control state variables. If no upset condition is present then at the next time instant, the execution of the nominal control law is resumed. But if an upset condition is detected, then the reload mode is re-entered since the current released state values could have been corrupted. To prevent an unacceptably long release-reload loop, an Abort (Ab) Mode is available. It is entered only after N_{Rs} successive visits to the Release Mode. This fail-safe Abort Mode produces a penalty, however, by introducing N_{Ab} additional units of delay in the feedback path before normal execution is restored. In general the total number of states in the finite-state machine is $N = 1+(N_{Rd}+1)N_{Rs}+N_{Ab}$. The overall effect of the recovery algorithm as a *filter* for the control signals is shown in Figure 1.1.2 for the case where the ideal control signal

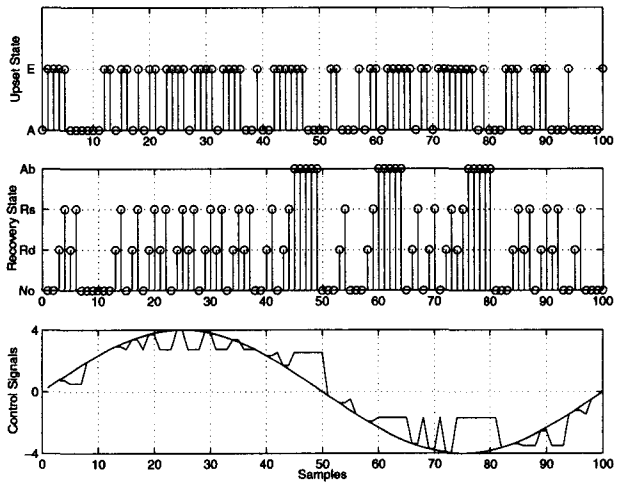

Figure 1.7: A sinusoid filtered by rollback recovery where $N_{Rs} = 2$, $N_{Ab} = 5$, $\Pi_{A|A} = 0.6$, and $\Pi_{E|E} = 0.5$.

is a sinusoid. While the persistent upset condition introduces significant distortion to the signal, in many cases the slower time constants of the plant will low-pass filter the high frequency signal components introduced by the recovery algorithm. The open-loop system is represented as a discrete-time dynamical system

$$\begin{aligned} x_p(k+1) &= A_p x_p(k) + B_p u_p(k), \quad x_p(0) \\ y_p(k) &= C_p x_p(k). \end{aligned}$$

Likewise, the nominal control law is assumed to be realizable as a dynamical system with realization (A_c, B_c, C_c). The jump-linear dynamical system is formed using the closed-loop models derived from the actions on the control signal in each recovery state. Hence, the state equations of the closed-loop dynamical system are switched through the set of A matrices: $\{A_{No}, A_{Rd}, A_{Rs}, A_{Ab}\}$. It is the stability conditions this jump-linear dynamical system which is the main concern of this thesis.

1.2 Literature Review

The study and analysis of models involving an interconnection of a finite-state machine (finite automata) with a dynamical system having an uncountable infinite state space comes under the broad area of theory of hybrid systems. In general, the dynamical system component with the infinite state space can be discrete or continuous, linear or non-linear and deterministic or stochastic. Hybrid dynamical systems have found direct application in manufacturing, power, communication and transport systems [48]. A basic hybrid framework of interconnection for the interconnection of automata and linear systems was suggested by Edwardo Sontag in 1995 [51]. There a hybrid system assumed the form

$$\begin{aligned} x^+ &= A_q x + B_q u \\ q^+ &= \delta(q, h(x,u)), \end{aligned} \tag{1.2}$$

where x is the state of the dynamical system, q is the state of the finite-state automaton, u is the external input, h is a function which maps the external input and the state of the dynamical system to a symbol, δ is the state transition function for the automaton and $'+'$ denotes the evolution of the system in either continuous or discrete time. Sontag suggested this piecewise linear system formalism in a deterministic setting and showed how several control problems can be solved with this formalism using tools from piecewise linear algebra. Since then the theory of hybrid systems has evolved in several directions. Stability of hybrid systems under deterministic conditions has been studied by [6,57]. The class of hybrid systems where the state evolution of the automata is coupled with the state evolution of a stochastic

dynamical system (described by either a stochastic differential or difference equation) has been studied by various researchers including Hespanha [20], Hu et al. [24] and Tejada et al. [53].

The research in this thesis mainly focuses on the stability of class of the class of jump-linear systems whose switching rule is governed by the output of a finite-state machine whose input is a Markovian random process. The hybrid model framework presented here is distinct from that given by the equation (1.2) and the systems in [20,24,53] in the sense that the state evolution of the finite-state machine does not depend on the state evolution of the jump-linear dynamical system, but only on its own state and the external Markov input.

Jump-linear systems as a class of hybrid systems has been an active area of research for several decades. The jump-linear as model has found application in the areas of control (power systems [55], tracking systems [39]) and communication [56]. The stability of jump-linear systems driven by Markov processes has been studied extensively by many researchers [8,27,42]. Lyapunov based stability tests for jump-linear systems whose switching rules are governed by independent identically distributed (also known as multinomial) processes and time inhomogeneous Markov processes appears in [10,11].

When a jump-linear system is driven by the output of a finite-state machine with Markovian input process, the output process of the finite-state machines is in general *not* a Markov process. It belongs to the more general class of random processes known as linear dependent processes [5,26]. Therefore the stability tests developed by various researchers for Markovian jump-linear systems cannot be directly applied.

In addition, stability criteria for jump-linear systems whose switching rule is governed by linear dependent processes has not been explored. This thesis hence provides the stability conditions for jump-linear systems driven by the output of finite-state machines with i.i.d. and higher order Markov inputs.

1.3 Problem Statement

The main research goals of this thesis are the following:

1. To characterize two random processes generated by a finite-state machine driven by a Markov process of order $r \geq 0$. The random processes considered are: $(\boldsymbol{\nu}, \boldsymbol{z})$ resulting from the cross-product of input process and the state process from the given finite-state machine and, $\tilde{\boldsymbol{z}}$, the state process of the equivalent unifilar type finite-state machine of the given finite-state machine.
2. To develop necessary and sufficient conditions for the mean-square stability of the hybrid model under consideration, $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$. More specifically, the hybrid model considered is the jump-linear system driven by minimal finite-state machine that is of
   (a) Moore type with an isomorphic state to output map
   (b) Mealy type with an isomorphic state to output map
   (c) Moore or Mealy type with a non-isomorphic state to output map
   (d) Unifilar type
   (e) Non-unifilar type

   and it is driven by a Markovian process order $r \geq 0$, i.e., a Markov process

with an

(a) independent identical distribution (i.i.d.) ($r = 0$)

(b) first-order distribution ($r = 1$)

(c) higher order distribution ($r > 1$).

3. To test and validate the necessary and sufficient conditions for mean-square stability of the model on several examples using Monte Carlo type simulations.

1.4 Thesis Overview

The thesis is organized as follows. In Chapter 2, after a preliminary introduction to finite-state machines, Markov processes and stochastic matrices, the two random processes: $(\boldsymbol{\nu}, \boldsymbol{z})$ and $\tilde{\boldsymbol{z}}$ are characterized. In Chapter 3, following appropriate definitions of stability for jump-linear systems, conditions for the mean-square stability of the model $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ are developed. The main stability results of Chapter 3 focus initially on the special case of a Moore type finite-state machine with an isomorphic state to output mapping, that is, each state in Σ_S has a corresponding unique output symbol in Σ_O, specifically, $L = N$, $\omega(\eta, e_j) = \omega(e_j) = \xi_j$ for $j = 1, \ldots, N$, $T_{\boldsymbol{\nu}(k)} = T$ and $A_{\omega(\eta_i, e_j)} = A_{\xi_j}$. In the later sections of Chapter 3, these results are extended to the more general case of non-isomorphic state to output mapping and Mealy type finite-state machines. Mean-square stability of such models are further characterized by the Lyapunov exponent method. A brief discussion concerning numerical and computational issues regarding the tests for mean-square stability conditions developed for $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ is also provided. In Chapter 4, the theoretical results are demon-

strated by simulating several simple examples of closed-loop systems with recovery algorithms. In the final chapter, the main conclusions from this research are presented along with a discussion about the directions for future research.

CHAPTER II

THEORY OF FINITE-STATE MACHINES WITH MARKOVIAN INPUTS

In this chapter, the class of finite-state machines used to model recovery algorithms is introduced. Elements of theory of Markov processes and stochastic matrices relevant for our problem is briefly discussed. It is well known that Markov chains and finite-state machines with random input process are closely connected mathematical entities [9, 26, 41]. It is also known that the output of a finite-state machine, $\boldsymbol{\theta}$, is in general not a Markov process. Instead, it belongs to a more general class of random processes known as linearly dependent processes [26]. But it is possible to characterize certain random processes associated with the finite-state machines with Markovian inputs as Markovian. Such a characterization is essential for the stability analysis purpose of the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, and hence forms the main content of this chapter.

This chapter is organized in the following manner. In Section 2.1, the formal definition of a finite-state machine is provided, along with other relevant terms and concepts. In Section 2.2, the theory for discrete-time, finite-state Markov chains and the theory of stochastic matrices relevant for the problem under consideration are described. In Section 2.3 and 2.4, two important random processes associated with finite-state machines with Markovian inputs are introduced. It is shown that the joint processes of input and state of every finite-state machine and the state process

of class of finite-state machines known as unifilar type are in fact always Markovian.

2.1 Introduction to the Theory of Finite-State Machines

In this section, definitions and terms from the finite-state machine theory will be introduced. Distinction will be made between Mealy type and Moore type machines, as well as, unifilar and non-unifilar type machines. A key result, useful in the stability analysis is the procedure to derive a unifilar type finite-state machine from a given non-unifilar machine. Most of the discussion here is based on [5, 31]. An excellent description of the algebraic approach to the theory of finite-state machine can be found in [23, 28].

A Mealy type finite-state machine is a five-tuple $\mathcal{M} = \{\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega\}$, where $\Sigma_I = \{\eta_1, \eta_2, \ldots, \eta_M\}$ is the finite set of input symbols, $\Sigma_S = \{e_1, e_2, \ldots, e_N\}$ is the set of states, $\Sigma_O = \{\xi_1, \xi_2, \ldots, \xi_L\}$ is the set of output symbols, $\delta : \Sigma_I \times \Sigma_S \mapsto \Sigma_S$ is the next state map, and $\omega : \Sigma_I \times \Sigma_S \mapsto \Sigma_O$ is the output map. A Moore type finite-state machine is the special case where the output function is $\omega : \Sigma_S \mapsto \Sigma_O$. Every Moore machine can be represented as a Mealy machine by assigning the output symbol in a manner independent of the current input symbol. The following definitions are fundamental.

Definition 2.1.1. *A Moore type finite-state machine $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$ is said to have an **isomorphic** state to output mapping, if $L = N$ and $\omega(e_j) \mapsto \xi_j$, $\forall e_j \in$*

Σ_S *and* $\forall \xi_j \in \Sigma_O$. *Similarly, a Mealy type finite-state machine is said to have an* ***isomorphic*** *state to output mapping, if* $L = N$ *and* $\omega(\eta, e_j) \mapsto \xi_j$, $\forall \eta \in \Sigma_I$, $\forall e_j \in \Sigma_S$, *and* $\forall \xi_j \in \Sigma_O$. *Otherwise the mapping is said to be* ***non-isomorphic***.

Definition 2.1.2. *[5] A finite-state machine is* ***completely specified****, if for every pair* $(\eta_i, e_j) \in \Sigma_I \times \Sigma_S$, *there exists a next state* $e_k \in \Sigma_S$ *such that* $\delta(\eta_i, e_j) = e_k$ *and for each state* $e_j \in \Sigma_S$, *there exists an output* $\xi_k \in \Sigma_O$ *such that* $\omega(e_j) = \xi_k$.

Definition 2.1.3. *[5] Let* J *be an arbitrary sequence of input symbols. Let* W_i *be the sequence of output symbols generated by* J *with* $e_i \in \Sigma_S$ *as the initial condition. Then the two initial states* $e_i, e_j \in \Sigma_S$ *are said to be* ***equivalent****, denoted as* $e_i \equiv e_j$, *if for all possible values of input sequence* J, *it follows that* $W_i = W_j$.

The following lemma is a consequence of Definition 2.1.3.

Lemma 2.1.1. *[5] The set of all equivalent states form an* ***equivalent class*** C_i, *and the set of equivalent classes forms a partition of the state space* Σ_S, *i.e.,* $\bigcup_i C_i = \Sigma_S$ *and* $C_i \bigcap C_j = \emptyset, \forall i, j$.

Definition 2.1.4. *[5] A finite-state machine is* ***minimal****, if every equivalent class* C_i *contains only one state* e_i.

Let J_l be a sequence of input symbols of length l, and $\triangle(J_l, e_j)$ be the state transition function with initial state e_j.

Definition 2.1.5. *[5] A finite-state machine is* ***strongly connected****, if for any pair of states* $e_j, e_k \in \Sigma_S$ *there exists at least one input sequence* J_l *such that* $\triangle(J_l, e_j) = e_k$.

Definition 2.1.6. *[5] Consider two finite-state machines* $\mathcal{M}_1 = \{\Sigma_I, \Sigma_S, \Sigma_O, \delta_1, \omega_1\}$ *and* $\mathcal{M}_2 = \{\Sigma_I, \Sigma_T, \Sigma_O, \delta_2, \omega_2\}$. *Let* $\triangle_1(J_l, e_i)$, $\triangle_2(J_l, f_j)$ *be the respective transition functions of the machines* $\mathcal{M}_1$ *and* $\mathcal{M}_2$. *The machines* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *are said to be* ***equivalent****, if for each state* $e_i \in \Sigma_S$ *there exists a state* $f_j \in \Sigma_T$ *such that* $\triangle_1(J_l, e_i) = \triangle_2(J_l, f_j)$ *and, conversely, for each state* $f_i \in \Sigma_T$ *there exists a state* $e_j \in \Sigma_S$ *such that* $\triangle_2(J_l, f_i) = \triangle_1(J_l, e_j)$.

Definition 2.1.7. *[37] A finite-state machine* $\tilde{\mathcal{M}} = (\Sigma_I, \tilde{\Sigma}_S, \tilde{\Sigma}_O, \tilde{\delta}, \tilde{\omega})$ *is* ***unifilar*** *if for any fixed state* $e_m \in \tilde{\Sigma}_S$, $\tilde{\delta}(\eta_i, e_m) \neq \tilde{\delta}(\eta_j, e_m)$ *whenever* $\eta_i \neq \eta_j$, *otherwise, the finite-state machine is said to be* ***non-unifilar***.

Theorem 2.1.1. *Let* $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$ *be an arbitrary non-unifilar finite-state machine. For any such finite-state machine* $\mathcal{M}$, *there always exists a unifilar finite-state machine* $\tilde{\mathcal{M}} = (\Sigma_I, \tilde{\Sigma}_S, \tilde{\Sigma}_O, \tilde{\delta}, \tilde{\omega})$ *which is* ***equivalent***.

Proof: The proof is constructive and it is applicable to both the cases of isomorphic state to output mapping and non-isomorphic state to output mapping. The state transition and output mappings are assigned to preserve the input-output mapping in order to produce an equivalent machine. Since $\mathcal{M}$ is non-unifilar, there exists a pair of inputs symbols η_i and η_j and machine states e_m and e_l such that $\delta(\eta_i, e_m) = \delta(\eta_j, e_m) = e_l$. Therefore, augment the machine's state space, Σ_S, by replacing the state e_l with new states e_{l_i} and e_{l_j}. Define a corresponding transition map $\tilde{\delta}$, which is identical to δ except now: $\tilde{\delta}(\eta_i, e_m) = e_{l_i}$, $\tilde{\delta}(\eta_j, e_m) = e_{l_j}$, and $\tilde{\delta}(\eta, e_{l_i}) = \delta(\eta, e_{l_j}) = \delta(\eta, e_l)$ for all input symbols η. Similarly, redefine the output function $\tilde{\omega}$ to be identical to ω except at the old state, $\tilde{\omega}(e_m) = \omega(e_m)$ and at the new state define $\tilde{\omega}(e_{l_i}) =$

$\tilde{\omega}(e_{l_j}) = \omega(e_l)$, so that the input and output relation is preserved. It is clear that the states and output symbols of the new machine can be re-indexed by the integers $1, 2, \ldots, N+1$, and the whole process above repeated if the new machine is not unifilar. Since the number of input symbols and machine states is finite, this procedure need only be repeated a finite number of times before a unifilar machine is produced. ■

2.2 Introduction to Discrete-Time Markov Processes and Stochastic Matrices

In this section, a brief description of the theory of homogeneous, discrete-time, finite-state Markov processes, Markov chains and stochastic matrices is given. Except where indicated, most of the presentation is based on [5,30,50].

Let $(\Omega, \mathcal{F}, P)$ be the underlying probability space, where Ω is the sample space, $\mathcal{F}$ is a σ-algebra on Ω, and P is a probability measure. At each time instance of k, $\boldsymbol{\nu}(k)$ is a discrete random variable which takes on values $x_j \in \Omega, 1 \leq j \leq n$. A homogeneous, discrete-time, finite-state r-th order Markov process $\boldsymbol{\nu}$ is a random process which satisfies the property

$$\begin{aligned} & P\{\boldsymbol{\nu}(k) = x_j | \boldsymbol{\nu}(k-1) = x_{j_1}, \boldsymbol{\nu}(k-2) = x_{j_2}, \ldots, \boldsymbol{\nu}(0) = x_{j_k}\} \\ = \; & P\{\boldsymbol{\nu}(k) = x_{j_1} | \boldsymbol{\nu}(k-1) = x_{j_2}, \boldsymbol{\nu}(k-2) = x_{j_3}, \ldots, \boldsymbol{\nu}(k-r) = x_{j_r}\}, \end{aligned}$$

for any $1 < r \leq \infty$.

For an r-th order Markov process $\boldsymbol{\nu}$, consider at each instance k, the two events $\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r+1)\}$ and $\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r)\}$ as two consecutive

states of another random process $\boldsymbol{\nu}_1$ at instance k and $(k-1)$, respectively. Thus, at each instance k, $\boldsymbol{\nu}_1(k) \in \underbrace{\Omega \times \ldots \times \Omega}_{\text{r times}}$. Then the random process $\boldsymbol{\nu}_1$ forms a first-order Markov process. Its transition probability can be derived in the following manner:

$$\begin{aligned} & P\{\boldsymbol{\nu}_1(k)|\boldsymbol{\nu}_1(k-1)\} \\ &= P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r+1)|\boldsymbol{\nu}(k-1), \boldsymbol{\nu}(k-2), \ldots, \boldsymbol{\nu}(k-r)\} \\ &= \frac{P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r+1), \boldsymbol{\nu}(k-r)\}}{P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r)\}} \\ &= P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r)\}. \end{aligned} \tag{2.1}$$

A stochastic matrix Π is a specific class of nonnegative matrix whose entries lie between 0 and 1, and column sums equal 1. An entry $[\Pi]_{ij}$ can be used to denote the transition probability from j-th state to i-th state of $\boldsymbol{\nu}_1(k) \in \underbrace{\Omega \times \ldots \times \Omega}_{\text{r times}}$. The following example illustrates these concepts for a second-order Markov process $\boldsymbol{\nu}$.

Example 2.2.1. *Consider the second-order Markov process $\boldsymbol{\nu}$ which takes its values from*

$$\boldsymbol{\Sigma_I} = \{\eta_1, \eta_2, \eta_3\}.$$

Then the corresponding first-order Markov process $\boldsymbol{\nu}_1$ takes its values from the set

$$\boldsymbol{\Sigma_I} \times \boldsymbol{\Sigma_I} = \{\eta_1\eta_1, \eta_1\eta_2, \eta_1\eta_3, \eta_2\eta_1, \eta_2\eta_2, \eta_2\eta_3, \eta_3\eta_1, \eta_3\eta_2, \eta_3\eta_3\}.$$

Using equation (2.1), its transition probability matrix has the following structure:

$$\Pi = \begin{bmatrix}
p_{\eta_1|\eta_1\eta_1} & p_{\eta_1|\eta_1\eta_2} & p_{\eta_1|\eta_1\eta_3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & p_{\eta_1|\eta_2\eta_1} & p_{\eta_1|\eta_2\eta_2} & p_{\eta_1|\eta_2\eta_3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & p_{\eta_1|\eta_3\eta_1} & p_{\eta_1|\eta_3\eta_2} & p_{\eta_1|\eta_3\eta_3} \\
p_{\eta_2|\eta_1\eta_1} & p_{\eta_2|\eta_1\eta_2} & p_{\eta_2|\eta_1\eta_3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & p_{\eta_2|\eta_2\eta_1} & p_{\eta_2|\eta_2\eta_2} & p_{\eta_2|\eta_2\eta_3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & p_{\eta_2|\eta_3\eta_1} & p_{\eta_2|\eta_3\eta_2} & p_{\eta_2|\eta_3\eta_3} \\
p_{\eta_3|\eta_1\eta_1} & p_{\eta_3|\eta_1\eta_2} & p_{\eta_3|\eta_1\eta_3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & p_{\eta_3|\eta_2\eta_1} & p_{\eta_3|\eta_2\eta_2} & p_{\eta_3|\eta_2\eta_3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & p_{\eta_3|\eta_3\eta_1} & p_{\eta_3|\eta_3\eta_2} & p_{\eta_3|\eta_3\eta_3}
\end{bmatrix}$$

□

A collection of states $\mathcal{E} \subseteq \underbrace{\Omega \times \ldots \times \Omega}_{\text{r times}}$ is called an *ergodic class* if it has the property that once the Markov process enters this class it will never leave it in the future. The collection of states $\mathcal{N}_{\mathcal{E}}$, called the *non-ergodic class*, has the property that once the Markov process leaves this class, it never re-enters it in the future. Every finite-state Markov chain $\mathcal{C}$ contains at least one ergodic class.

Let D_i be a set of positive integers corresponding to the state i such that for only those integers $m \in D_i$, $[\Pi^m]_{ii} > 0$, otherwise the quantity is zero. Then the greatest common divisor of all the elements of D_i is called the *period* of the i-th state and is denoted by $d(i)$. All the states within the same ergodic class have the same period. If all the states in all the ergodic classes have period $d = 1$, then the chain is said to be *aperiodic*. If $[\Pi]_{ii} > 0$ then the period of i-th state is 1. The following definition is

essential.

Definition 2.2.1. *A finite-state Markov chain $\mathcal{C}$ and its transition probability matrix Π are said to be* ***monodesmic*** *if $\mathcal{V} = \mathcal{E}_1 \cup \mathcal{N}_\mathcal{E}$, irrespective of whether $\mathcal{N}_\mathcal{E}$ is empty or not.*

A monodesmic chain $\mathcal{C}$ with a non-ergodic class $\mathcal{N}_\mathcal{E} \neq \emptyset$ is said to be *reducible.* There always exists a permutation matrix $\mathcal{T}$ such that Π can be put in the following canonical form:

$$\mathcal{T}\Pi\mathcal{T}' = \left[\begin{array}{c|c} \Pi_{\mathcal{E}_1} & R \\ \hline \mathbf{0} & T_{\mathcal{N}_\mathcal{E}} \end{array}\right].$$

In this case, the matrix Π is said to be *reducible.* A monodesmic chain with $\mathcal{N}_\mathcal{E} = \emptyset$ is said to be *irreducible* or *ergodic.* The matrix Π is referred to as *irreducible*, and there always exists a positive integer m_{ji} for each pair (i, j) such that $[\Pi^{m_{ji}}]_{ji} > 0$.

Definition 2.2.2. *A stochastic matrix Π is said to be* ***primitive*** *if there exists a positive integer m such that $[\Pi^m]_{ji} > 0$ for all (i, j). The smallest integer m satisfying this condition is called the* ***index of primitivity***.

An irreducible stochastic matrix has period $d = 1$ if and only if it is primitive. The corresponding chain is called a *regular chain.* It is both ergodic and aperiodic. When $[\Pi]_{ji} > 0$ for all (i, j), denoted by $\Pi > 0$, Π is obviously primitive with index of primitivity $m = 1$.

This section is concluded with a concise summary of the key results used in the next section. The first theorem gives eigen-structure characterizations of monodesmic matrices, irreducible matrices and primitive matrices. The next two theorems concern

monodesmic stochastic matrices. They provide tests for the existence and uniqueness of the stationary vector of $\Pi_{I/O}$. Corollary 2.2.2 will be used to show that the condition $p_\xi > 0$ is equivalent to reachability.

Theorem 2.2.1. *Let* Π *be a stochastic matrix. Then:*

1. Π *is monodesmic if and only if* Π *has a single eigenvalue at* $\lambda_r = 1$ *[29, p. 4]. The corresponding eigenvector* q *is unique to within a constant multiple [30, p. 117].*
2. Π *is irreducible if and only if* Π *is monodesmic with* $q > 0$ *[30, pp. 100, 117].*

(Here $q > 0$ *means that every component of* q *is positive.)*

Theorem 2.2.2. *[30, pp. 70-71, 117], If a stochastic matrix* Π *is monodesmic and aperiodic then*

$$\lim_{n\to\infty} \Pi^n = \lim_{n\to\infty} \begin{bmatrix} \Pi^n_{\mathcal{E}_1} & * \\ 0 & T^n_{\mathcal{N}_\mathcal{E}} \end{bmatrix} = \begin{bmatrix} q, & q, & q, & \cdots & q \end{bmatrix}$$

is a rank one matrix with rows of zeroes corresponding to the non-ergodic states and columns q *corresponding to the unique stationary vector of* Π.

Corollary 2.2.1. *[30, pp. 70-71] If a stochastic matrix* Π *is primitive then*

$$\lim_{n\to\infty} \Pi^n \to \begin{bmatrix} q, & q, & q, & \cdots & q \end{bmatrix},$$

where $q > 0$ *is the unique stationary probability vector of* Π.

Theorem 2.2.3. *[30, pp. 100,117] If a stochastic matrix* Π *is monodesmic and the ergodic class has period* $d > 1$ *then for some* $0 < k < 1$

$$\lim_{n\to\infty}(kI + (1-k)\Pi)^n = \begin{bmatrix} q & q & \dots & q \end{bmatrix},$$

where q *is the unique stationary probability vector of* Π.

Corollary 2.2.2. *[30, p. 100] If a stochastic matrix* Π *is irreducible with period* $d > 1$ *then for some* $0 < k < 1$

$$\lim_{n\to\infty}(kI + (1-k)\Pi)^n = \begin{bmatrix} q & q & \dots & q \end{bmatrix},$$

where $q > 0$ *is the unique stationary probability vector of* Π, *in which case for any initial probability vector* q_0, $\lim_{n\to\infty}(kI + (1-k)\Pi)^n q_0 = q$.

2.3 Characterization of Random Processes Generated by Finite-State Machines with Markovian Inputs

Two important types of random processes result from a finite-state machine when a Markovian input is applied. Each will be characterized in the following subsections. The first process is the cross-chain process $\boldsymbol{\rho}$. It is obtained from the cross-product of the input process and the state process, namely $(\boldsymbol{\nu}, \boldsymbol{z})$. It takes symbols from $\Sigma_I \times \Sigma_S$. The second process $\tilde{\boldsymbol{z}}$ is obtained when the given finite-state machine is replaced by its equivalent unifilar type finite-state machine and driven by the original input. First,

the notion of *strongly connected* finite-state machines for the deterministic input case is extended to the case of Markovian inputs.

Definition 2.3.1. *A finite-state machine $\mathcal{M}$ with Markovian input (Σ_I, Π_I) is said to be* ***reachable*** *if for every initial state $e_j \in \Sigma_S$, there exists a finite sequence of input symbols from Σ_I which occurs with nonzero probability and drives the machine to any other state in Σ_S.*

A finite-state machine can be strongly connected under deterministic input conditions, yet, it may not be *reachable* for certain Markovian inputs.

2.3.1 Cross-Chain of Input and State Process $\boldsymbol{\rho} = (\boldsymbol{\nu}, \boldsymbol{z})$

Given a finite-state machine $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$, the Markov property of $\boldsymbol{\rho}$ is ascertained first in the case of an independent identically distributed input process $\boldsymbol{\nu}$, followed by the case when the input process $\boldsymbol{\nu}$ is Markovian of order $r \geq 1$. Finally, a first-order representation for the cross-chain process is provided when the input process $\boldsymbol{\nu}$ is Markovian of any order $r \geq 1$.

Theorem 2.3.1. *Consider a process $\boldsymbol{\rho} = (\boldsymbol{\nu}, \boldsymbol{z})$, where $\boldsymbol{\nu}$ and $\boldsymbol{z}$ are, respectively, the input process and state process of a finite-state machine $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$. If $\boldsymbol{\nu}$ is an independent identically distributed process independent of the initial state of the machine, $\boldsymbol{z}(0)$, then $\boldsymbol{\rho}$ is a first-order Markov process.*

Proof: Consider an event in the σ-algebra $\mathcal{F}$ denoted by

$$\{\boldsymbol{\rho}(k), \boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\},$$

where $r \leq m \leq k$. Precisely, this denotes the set of all outcomes that produce a fixed but arbitrary sequence $\rho(k), \rho(k-1), \ldots, \rho(k-m)$. Now in the case where

$$\delta(\boldsymbol{\nu}(i), \boldsymbol{z}(i)) = (\boldsymbol{z}(i+1)), \quad \forall i = k-m, \ldots, k-1,$$

it follows immediately that

$$P\{\boldsymbol{\rho}(k), \boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\} = P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}.$$

All other such events are impossible. From the assumption that $\boldsymbol{\nu}$ is independent identically distributed and independent of $\boldsymbol{z}(0)$, it follows that $\boldsymbol{\nu}(k)$ is independent of $\boldsymbol{z}(k-m)$ for all $1 \leq m \leq k$. Therefore,

$$\begin{aligned}
&P\{\boldsymbol{\rho}(k), \boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\} \\
&= P\{\boldsymbol{\nu}(k), \boldsymbol{z}(k-m) | \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} \\
&= P\{\boldsymbol{\nu}(k) | \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} P\{\boldsymbol{z}(k-m) | \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} \cdot \\
&\qquad P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} \\
&= \frac{P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}}{P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}} \cdot \\
&\qquad\qquad P\{\boldsymbol{z}(k-m), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}. \qquad (2.2)
\end{aligned}$$

Using a similar argument it follows that

$$\begin{aligned}
&P\{\boldsymbol{\rho}(k-1), \boldsymbol{\rho}(k-2), \ldots, \boldsymbol{\rho}(k-m)\} \\
&\quad = P\{\boldsymbol{\nu}(k-1), \boldsymbol{\nu}(k-2), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}. \qquad (2.3)
\end{aligned}$$

Dividing equation (2.2) by equation (2.3) gives for all $1 \leq m \leq k$,

$$P\{\boldsymbol{\rho}(k) | \boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\} = P\{\boldsymbol{\nu}(k) | \boldsymbol{\nu}(k-1), \boldsymbol{\nu}(k-2), \ldots, \boldsymbol{\nu}(k-m)\}.$$

However, $\boldsymbol{\nu}$ is an i.i.d. process and hence for all $1 \leq m \leq k$, the right hand side reduces to

$$P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-m)\} = P\{\boldsymbol{\nu}(k)\}.$$

Thus,

$$P\{\boldsymbol{\rho}(k)|\boldsymbol{\rho}(k-1),\ldots,\boldsymbol{\rho}(k-m)\} \;=\; P\{\boldsymbol{\nu}(k)\}.$$

Therefore $\boldsymbol{\rho}$ is a first-order Markov process with transition probability

$$P\{\boldsymbol{\rho}(k)|\boldsymbol{\rho}(k-1)\} = P\{\boldsymbol{\nu}(k)\}. \tag{2.4}$$

■

As an aside, note that $\boldsymbol{\rho}$ is never an i.i.d. process since,

$$\begin{aligned} P\{\boldsymbol{\rho}(k)\} &= P\{(\boldsymbol{\nu}(k),\boldsymbol{z}(k))\} \\ &= P\{\boldsymbol{\nu}(k)|\boldsymbol{z}(k)\}P\{\boldsymbol{z}(k)\}, \end{aligned}$$

and $\boldsymbol{\nu}$ is i.i.d.,

$$\begin{aligned} P\{\boldsymbol{\rho}(k)\} &= P\{\boldsymbol{\nu}(k)\}P\{\boldsymbol{z}(k)\} \\ &\neq P\{\boldsymbol{\nu}(k)\}. \end{aligned}$$

The following theorem characterizes the Markov property of the cross-chain process $\boldsymbol{\rho} = (\boldsymbol{\nu}, \boldsymbol{z})$ when input process $\boldsymbol{\nu}$ is Markovian of order $r \geq 1$:

Theorem 2.3.2. *[38] [44] Consider a process $\boldsymbol{\rho} = (\boldsymbol{\nu}, \boldsymbol{z})$, where $\boldsymbol{\nu}$ and $\boldsymbol{z}$ are, respectively, the input and state of a finite-state machine $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$. If $\boldsymbol{\nu}$ is an r-th order Markov process with $r \geq 1$, independent of the initial state of the machine, $\boldsymbol{z}(0)$, then $\boldsymbol{\rho}$ is also an r-th order Markov process.*

Proof: Consider an event denoted by

$$\{\boldsymbol{\rho}(k), \boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\},$$

where $r \leq m \leq k$. Precisely, this denotes the set of all outcomes in $\underbrace{\Omega \times \ldots \Omega}_{\text{r times}}$. Now in the case where

$$\delta(\boldsymbol{\nu}(i), \boldsymbol{z}(i)) = (\boldsymbol{z}(i+1)), \ \ \forall i = k-m, \ldots, k-1,$$

it follows immediately that

$$P\{\boldsymbol{\rho}(k), \boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\} = P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}.$$

All other such events are impossible. From the assumption that $\boldsymbol{\nu}$ is r-th order Markov and independent of $\boldsymbol{z}(0)$, it follows that $\boldsymbol{\nu}(k)$ is independent of $\boldsymbol{z}(k-m)$ for all $r \leq m \leq k$. Therefore,

$$\begin{aligned}
&P\{\boldsymbol{\rho}(k), \boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\} \\
&= P\{\boldsymbol{\nu}(k), \boldsymbol{z}(k-m) | \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} \\
&= P\{\boldsymbol{\nu}(k) | \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} P\{\boldsymbol{z}(k-m) | \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} \cdot \\
&\qquad P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} \\
&= \frac{P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}}{P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}} \cdot \\
&\qquad P\{\boldsymbol{z}(k-m), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}. \qquad (2.5)
\end{aligned}$$

Using a similar argument it follows that

$$\begin{aligned}
&P\{\boldsymbol{\rho}(k-1), \ldots, \boldsymbol{\rho}(k-m)\} \\
&\qquad = P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}. \qquad (2.6)
\end{aligned}$$

Dividing equation (2.5) by equation (2.6) gives

$$P\{\boldsymbol{\rho}(k)|\boldsymbol{\rho}(k-1),\ldots,\boldsymbol{\rho}(k-m)\} = P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\boldsymbol{\nu}(k-2),\ldots,\boldsymbol{\nu}(k-m)\}.$$

Finally, since again the input process $\boldsymbol{\nu}$ is assumed to be r-th order Markov, for any $m \geq r$:

$$\begin{aligned} P\{\boldsymbol{\rho}(k)|\boldsymbol{\rho}(k-1),\ldots,\boldsymbol{\rho}(k-m)\} &= P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\boldsymbol{\nu}(k-2),\ldots,\boldsymbol{\nu}(k-r)\} \\ &= P\{\boldsymbol{\rho}(k)|\boldsymbol{\rho}(k-1),\boldsymbol{\rho}(k-2),\ldots,\boldsymbol{\rho}(k-r)\}, \quad (2.7) \end{aligned}$$

and hence the proof. ■

A first-order Markov representation of the cross-chain process, which reduces the dimensionality of the transition matrix is described in the following theorem. This theorem will be later used to derive the stability criterion when input process is Markov of order $r \geq 0$, instead of the result in Theorem 2.3.2.

Theorem 2.3.3. *[54] Consider a process $\boldsymbol{\rho}_1 = (\boldsymbol{\nu}_1, \boldsymbol{z})$, where $\boldsymbol{\nu}_1$ is the first-order representation of the r-th order input process $\boldsymbol{\nu}$ and $\boldsymbol{z}$ is the state process of the finite-state machine $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$. If $\boldsymbol{\nu}$ is an r-th order Markov process independent of the initial state of the machine, $\boldsymbol{z}(0)$, where $r \geq 0$, then $\boldsymbol{\rho}_1$ is a first-order Markov process.*

Proof: If $\boldsymbol{\nu}$ is an r-th order Markov process, then the random process $\boldsymbol{\nu}_1$ with $\boldsymbol{\nu}_1(k) = (\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r+1)) \in \underbrace{\Sigma_I \times \ldots \Sigma_I}_{\text{r times}}$ forms a first-order Markov

process with transition probabilities

$$\begin{aligned} P\{\boldsymbol{\nu}_1(k)|\boldsymbol{\nu}_1(k-1)\} &= P\{\boldsymbol{\nu}(k),\ldots,\boldsymbol{\nu}(k-r+1)|\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r)\} \\ &= \frac{P\{\boldsymbol{\nu}(k),\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r)\}}{P\{\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r)\}} \\ &= P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r)\}. \end{aligned}$$

Now by Theorem 2.3.2, for any first-order process $\boldsymbol{\nu}$ with $\boldsymbol{\nu}(k) \in \Sigma_I$, the random process $\boldsymbol{\rho} = (\boldsymbol{\nu}, \boldsymbol{z})$ is a first-order Markov process with $\boldsymbol{\rho}(k) \in \Sigma_I \times \Sigma_S$. Hence, it follows that for the first-order Markov process $\boldsymbol{\nu}_1$ with $\boldsymbol{\nu}_1(k) \in \underbrace{\Sigma_I \times \ldots \Sigma_I}_{\text{r times}}$, the random process $\boldsymbol{\rho}_1 = (\boldsymbol{\nu}_1, \boldsymbol{z})$ with $\boldsymbol{\rho}_1(k) \in \underbrace{\Sigma_I \times \ldots \Sigma_I}_{\text{r times}} \times \Sigma_S$ is also a first-order Markov process with

$$\begin{aligned} P\{\boldsymbol{\rho}_1(k)|\boldsymbol{\rho}_1(k-1)\} &= P\{\boldsymbol{\nu}_1(k)|\boldsymbol{\nu}_1(k-1)\} \\ &= P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r)\}, \end{aligned} \tag{2.8}$$

and hence the proof. ■

2.3.2 State Process of a Non-Unifilar Finite-State Machine $\boldsymbol{z}$

It is known that the state process $\boldsymbol{z}$ of a non-unifilar type finite-state machine is in general *not* Markovian [41]. In this section however, it will be shown that the state process of the non-unifilar finite-state machine driven by the i.i.d. process is Markovian. The corresponding transition probabilities for the process $\boldsymbol{z}$ will be derived in terms of the transition probabilities of the input process.

Theorem 2.3.4. *Consider a finite-state machine* $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$. *If the*

input process $\boldsymbol{\nu}$ is an i.i.d. process whose distribution is independent of the initial state of the machine, $\boldsymbol{z}(0)$, then the state process $\boldsymbol{z}$ is a first-order Markov process.

Proof: Consider an event of the form $\{\boldsymbol{z}(k+1), \boldsymbol{z}(k), \ldots, \boldsymbol{z}(k-m)\}$ where $0 \leq m \leq k$. Let $\mathcal{L}_j = \{\eta : \delta(\eta, \boldsymbol{z}(j)) = \boldsymbol{z}(j+1)\}$ with $k-m \leq j \leq k$ be the set of input symbols of size $1 \leq n_j \leq M$. Notice that, if $\mathcal{L}_j = \emptyset$ for any $k-m \leq j \leq k$, then the event never occurs. Hence,

$$P\{\boldsymbol{z}(k+1), \boldsymbol{z}(k), \ldots, \boldsymbol{z}(k-m)\}$$
$$= \sum_{\eta \in \mathcal{L}_k} \cdots \sum_{\eta \in \mathcal{L}_{(k-m)}} P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}.$$

Now for the event $\{\boldsymbol{z}(k), \ldots, \boldsymbol{z}(k-m)\}$,

$$P\{\boldsymbol{z}(k), \ldots, \boldsymbol{z}(k-m)\}$$
$$= \sum_{\eta \in \mathcal{L}_{(k-1)}} \cdots \sum_{\eta \in \mathcal{L}_{(k-m)}} P\{\boldsymbol{\nu}(k-1), \boldsymbol{\nu}(k-2), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}$$

Thus

$$\frac{P\{\boldsymbol{z}(k+1), \boldsymbol{z}(k), \ldots, \boldsymbol{z}(k-m)\}}{P\{\boldsymbol{z}(k), \ldots, \boldsymbol{z}(k-m)\}}$$
$$= \frac{\sum_{\eta \in \mathcal{L}_k} \cdots \sum_{\eta \in \mathcal{L}_{(k-m)}} P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}}{\sum_{\eta \in \mathcal{L}_{(k-1)}} \cdots \sum_{\eta \in \mathcal{L}_{n_{(k-m)}}} P\{\boldsymbol{\nu}(k-1), \boldsymbol{\nu}(k-2), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}} \quad (2.9)$$

Since $\boldsymbol{\nu}$ is an i.i.d. process and its distribution is independent of $\boldsymbol{z}(0)$, for all $0 \leq m \leq k$, $\boldsymbol{\nu}(k)$ is independent of $\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}$ and $\boldsymbol{z}(k-m)$. Hence, $\boldsymbol{\nu}(k)$ is independent of the event $\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \boldsymbol{z}(k-m)\}$, for all $0 \leq m \leq k$.

Therefore the right hand side of equation (2.9) becomes

$$\frac{\sum_{\eta\in\mathcal{L}_k}\cdots\sum_{\eta\in\mathcal{L}_{(k-m)}} P\{\boldsymbol{\nu}(k)\}P\{\boldsymbol{\nu}(k-1),\boldsymbol{\nu}(k-2),\ldots,\boldsymbol{\nu}(k-m),\boldsymbol{z}(k-m)\}}{\sum_{\eta\in\mathcal{L}_{(k-1)}}\cdots\sum_{\eta\in\mathcal{L}_{(k-m)}} P\{\boldsymbol{\nu}(k-1),\boldsymbol{\nu}(k-2),\ldots,\boldsymbol{\nu}(k-m),\boldsymbol{z}(k-m)\}}$$

$$= \sum_{\eta\in\mathcal{L}_k} P\{\boldsymbol{\nu}(k)\}\cdot \frac{\sum_{\eta\in\mathcal{L}_{(k-1)}}\cdots\sum_{\eta\in\mathcal{L}_{(k-m)}} P\{\boldsymbol{\nu}(k-1),\boldsymbol{\nu}(k-2),\ldots,\boldsymbol{\nu}(k-m),\boldsymbol{z}(k-m)\}}{\sum_{\eta\in\mathcal{L}_{(k-1)}}\cdots\sum_{\eta\in\mathcal{L}_{(k-m)}} P\{\boldsymbol{\nu}(k-1),\boldsymbol{\nu}(k-2),\ldots,\boldsymbol{\nu}(k-m),\boldsymbol{z}(k-m)\}}$$

$$= \sum_{\eta\in\mathcal{L}_k} P\{\boldsymbol{\nu}(k)\}.$$

Since this is true for all $0 \leq m \leq k$,

$$P\{\boldsymbol{z}(k+1)|\boldsymbol{z}(k)\} = \sum_{\eta\in\mathcal{L}_k} P\{\boldsymbol{\nu}(k)\}, \tag{2.10}$$

and hence the proof. ■

Now, we state a well known related theorem due to Davis [9]. This theorem is inverse of the previous theorem.

Theorem 2.3.5. *([9, 26]) Every first-order Markov process with N states and its transition probability Π can be realized by driving a N state Moore type finite-state machine with isomorphic state to output mapping driven by an i.i.d. process having at most $N \times (N-1)$ input symbols. This also implies that every stochastic matrix of dimension N can be written as a convex combination of at the most $N \times (N-1)$ Boolean matrices.*

Proof: Proof is the algorithm to achieve

$$\Pi = \sum_{\eta} p_\eta S_\eta.$$

1. Let $\hat{\Pi} = \Pi$ and initialize $k = 1$.

2. Find the nonzero minimum number m_{ji} in each column i of $\hat{\Pi}$. If there is more than one m_{ji} for a certain column i, consider only one of them as m_{ji}.

3. Let $p_k = minimum(m_{ji})$.

4. Form a matrix S_k by replacing the each i^{th} column with $e_j = \begin{bmatrix} 0 & 0 & \dots & 1 & \dots & 0 \end{bmatrix}^T$ with 1 in the position of m_{ji}.

5. Let $\hat{\Pi} = \hat{\Pi} - \sum_k p_k S_k$ and increment k by one.

6. Repeat the above procedure starting from step 2, till $\hat{\Pi} = 0$.

Since there are only N^2 elements, there are at the most $N \times (N-1)$ number of iterations required to achieve this and hence let $M \leq N \times (N-1)$ be the maximum number of p_k and S_k used for the given Π. Now, consider a Moore type finite-state machine $\mathcal{M} = \{\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega\}$ with input set $\Sigma_I = \{\eta_1, \eta_2, \dots, \eta_M\}$, $\Sigma_S = \{e_1, e_2, \dots, e_N\}$, $\Sigma_O = \{\xi_1, \xi_2, \dots, \xi_N\}$, with mapping δ determined according to $S_\eta = S_k$ and $\omega(e_j) = \xi_j$. Apply an i.i.d. process as input ν with probabilities $p_\eta = p_k$ to the machine $\mathcal{M}$. By Theorem 2.3.4, the resulting output is a first-order Markov process governed by the transition probabilities Π. ■

2.3.3 State Process of a Unifilar Finite-State Machine $\tilde{z}$

In this section, it will be shown that the state process $\tilde{\boldsymbol{z}}$ of any unifilar finite-state machine when driven by Markovian input of order $r \geq 0$ is always Markovian. Following theorem explains this result.

Theorem 2.3.6. *[37, 44] Consider a unifilar finite-state machine*

$$\tilde{\mathcal{M}} = (\Sigma_I, \tilde{\Sigma}_S, \tilde{\Sigma}_O, \tilde{\delta}, \tilde{\omega})$$

with input process $\boldsymbol{\nu}$ and state process $\tilde{\boldsymbol{z}}$. If $\boldsymbol{\nu}$ is an r-th order Markov process, which is independent of the initial machine state $\tilde{\boldsymbol{z}}(0)$, with $r \geq 0$, then $\tilde{\boldsymbol{z}}$ is an $(r+1)$-st order Markov process.

Proof: Consider an event of the form $\{\tilde{\boldsymbol{z}}(k+1), \tilde{\boldsymbol{z}}(k), \ldots, \tilde{\boldsymbol{z}}(k-m)\}$ where $r \leq m \leq k$. Since $\tilde{\mathcal{M}}$ is assumed to be unifilar, this event is equivalent to the event $\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \tilde{\boldsymbol{z}}(k-m)\}$, where $\boldsymbol{\nu}(j) \in \mathcal{L}_j = \{\eta : \delta(\boldsymbol{\nu}(j), \boldsymbol{z}(j)) = \boldsymbol{z}(j+1)\}, k \leq j \leq (k-m)$. Therefore,

$$P\{\tilde{\boldsymbol{z}}(k+1), \tilde{\boldsymbol{z}}(k), \ldots, \tilde{\boldsymbol{z}}(k-m)\} = P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m), \tilde{\boldsymbol{z}}(k-m)\}$$

and similarly,

$$P\{\tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{z}}(k-1), \ldots, \tilde{\boldsymbol{z}}(k-m)\} = P\{\boldsymbol{\nu}(k-1), \boldsymbol{\nu}(k-2), \ldots, \boldsymbol{\nu}(k-m), \tilde{\boldsymbol{z}}(k-m)\}.$$

The assumption that $\boldsymbol{\nu}$ is an r-th order Markov process independent of $\tilde{\boldsymbol{z}}(0)$ implies that $\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\}$ is independent of $\tilde{\boldsymbol{z}}(k-m)$ when $r \leq m \leq k$. Thus,

$$\frac{P\{\tilde{\boldsymbol{z}}(k+1), \tilde{\boldsymbol{z}}(k), \ldots, \tilde{\boldsymbol{z}}(k-m)\}}{P\{\tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{z}}(k-1), \ldots, \tilde{\boldsymbol{z}}(k-m)\}} = \frac{P\{\boldsymbol{\nu}(k), \boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} P\{\tilde{\boldsymbol{z}}(k-m)\}}{P\{\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-m)\} P\{\tilde{\boldsymbol{z}}(k-m)\}},$$

or equivalently,

$$P\{\tilde{\boldsymbol{z}}(k+1)|\tilde{\boldsymbol{z}}(k),\tilde{\boldsymbol{z}}(k-1),\ldots,\tilde{\boldsymbol{z}}(k-m)\} = P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-m)\}.$$

Finally, since $\boldsymbol{\nu}$ is Markov process of order r, for every $m \geq r$:

$$\begin{aligned} P\{\tilde{\boldsymbol{z}}(k+1)|\tilde{\boldsymbol{z}}(k),\tilde{\boldsymbol{z}}(k-1),\ldots,\tilde{\boldsymbol{z}}(k-m)\} &= P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r)\} \\ &= P\{\tilde{\boldsymbol{z}}(k+1)|\tilde{\boldsymbol{z}}(k),\ldots,\tilde{\boldsymbol{z}}(k-r)\}, \end{aligned}$$

and hence the proof. ■

Observe that, when the input is an i.i.d. process then $P\{\tilde{\boldsymbol{z}}(k+1)|\tilde{\boldsymbol{z}}(k)\} = P\{\boldsymbol{\nu}(k)\}$, where $\boldsymbol{\nu}(k) \in \mathcal{L}_k$. This is consistent with Theorem 2.3.4.

Corollary 2.3.1. *If $\tilde{\boldsymbol{z}}$ is an $(r+1)$-th order process with $\tilde{\boldsymbol{z}}(k) \in \Sigma_S$, then $\tilde{\boldsymbol{z}}_1$ with $\tilde{\boldsymbol{z}}_1(k) = (\boldsymbol{z}(k), \boldsymbol{z}(k-1),\ldots,\boldsymbol{z}(k-r)) \in \underbrace{\Sigma_S \times \ldots, \Sigma_S}_{r+1 \text{ times}}$ is a first-order Markov process.*

Proof: The proof follows from applying equation (2.1) to the $(r+1)$-th order process $\tilde{\boldsymbol{z}}$. The process $\tilde{\boldsymbol{z}}_1$ then has the transition probabilities

$$\begin{aligned} P\{\tilde{\boldsymbol{z}}_1(k+1)|\tilde{\boldsymbol{z}}_1(k)\} &= P\{\tilde{\boldsymbol{z}}(k+1)|\tilde{\boldsymbol{z}}(k),\ldots,\tilde{\boldsymbol{z}}(k-r)\} \\ &= P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1),\ldots,\boldsymbol{\nu}(k-r)\}. \end{aligned} \tag{2.11}$$

■

In particular, note that when $r = 0$, i.e., when the input is an independent i.i.d. process then for any given finite-state machine,

$$\begin{aligned} P\{\boldsymbol{\rho}(k)|\boldsymbol{\rho}(k-1)\} &= P\{\boldsymbol{\nu}(k)\} \\ &= P\{\boldsymbol{\rho}_1(k)|\boldsymbol{\rho}_1(k-1)\} \\ P\{\boldsymbol{z}(k+1)|\boldsymbol{z}(k)\} &= \sum_{\eta \in \mathcal{L}_k} P\{\boldsymbol{\nu}(k)\}. \end{aligned}$$

When $r = m = 1$ equation (2.7) gives for any given finite-state machine

$$P\{\boldsymbol{\rho}(k)|\boldsymbol{\rho}(k-1)\} = P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1)\}, \tag{2.12}$$

and (2.11) gives for a unifilar type finite-state machine

$$\begin{aligned} P\{\tilde{\boldsymbol{z}}(k+1)|\tilde{\boldsymbol{z}}(k), \tilde{\boldsymbol{z}}(k-1)\} &= P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1)\} \\ &= P\{\tilde{\boldsymbol{z}}_1(k+1)|\tilde{\boldsymbol{z}}_1(k)\} \end{aligned} \tag{2.13}$$

For $r > 1$ for any given finite-state machine

$$P\{\boldsymbol{\rho}_1(k)|\boldsymbol{\rho}_1(k-1)\} = P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r)\}$$

and (2.11) gives for a unifilar type finite-state machine

$$P\{\tilde{\boldsymbol{z}}_1(k+1)|\tilde{\boldsymbol{z}}_1(k)\} = P\{\boldsymbol{\nu}(k)|\boldsymbol{\nu}(k-1), \ldots, \boldsymbol{\nu}(k-r)\}.$$

2.4 Summary of Key Points

In this chapter, elements of the theory of finite-state machines, the theory of Markov processes and stochastic matrices relevant to this research were briefly described. In addition, two random processes associated with a finite-state machine were characterized: the cross product of the input process and the state process for a finite-state machine and the state process of a unifilar equivalent machine were characterized when the inputs were independent identically distributed process, a first-order Markov process and a higher order Markov process. It turns out that the cross-chain process and state process of any finite-state machine is first-order Markov when the input process is independent identically distributed. Where as for first and

higher order Markov inputs, the order of cross-chain process is the same as that of the input process. The output process of a unifilar equivalent finite-state machine is one order higher than that of the input process. Table 2.1 summarizes these key results.

Table 2.1: Order of the cross-chain process for a finite-state machine and the state process for a unifilar machine for Markovian inputs of various order r.

Order of the input Markov Process	Order of the Cross-Chain Process of FSM		Order of the State Process of a Unifilar FSM	
	$\boldsymbol{\rho}$	$\boldsymbol{\rho}_1$	$\tilde{\boldsymbol{z}}$	$\tilde{\boldsymbol{z}}_1$
$r = 0$	1	1	1	1
$r = 1$	1	1	2	1
$r > 1$	r	1	$r+1$	1

CHAPTER III

STABILITY ANALYSIS

3.1 Introduction

In this chapter, using the main results from Chapter 2, stability conditions for jump-linear systems driven by finite-state machines with Markov inputs are provided. The main approach is to develop the mean-square stability tests for the class of hybrid systems comprising of a jump-linear system driven by a Markovian cross-chain process $\boldsymbol{\rho}$. Mean-square stability tests will also be developed for the class of hybrid systems comprising of a jump-linear system driven by a Markovian state process of a unifilar finite-state machine, $\tilde{\boldsymbol{z}}$. Then, it will be shown using the tests developed for these specific classes of hybrid systems, that the mean-square stability conditions for a jump-linear system driven by an output process $\boldsymbol{\theta}$ of any finite-state machine (which in general may not be Markovian) can be synthesized.

This chapter is organized in the following manner. In the next section, the key assumptions about the model are stated, followed by various tests to validate these assumptions in practice. It is followed by a section providing the main definitions for the stability of a jump-linear system currently used by various researchers in this field (specifically, in [3,8,11,13,27]). In the subsequent section, mean-square stability conditions for a jump-linear system driven by the cross-chain process $\boldsymbol{\rho} = (\boldsymbol{\nu}, \boldsymbol{\theta})$ of the Moore type finite-state machine with isomorphic state to output mapping will

be developed. This is followed by the mean-square stability conditions for the jump-linear system driven by the output process of a Moore type finite-state machine with isomorphic state to output mapping, $\boldsymbol{\theta}$, when input is an i.i.d. process and the output process of the Moore type unifilar finite-state machine with isomorphic state to output mapping, $\tilde{\boldsymbol{\theta}}$, when input is Markovian of order $r \geq 1$. The concept of A-equivalent systems is then introduced and their mean-square stability characteristics are related. A brief discussion will show how these stability tests can be extended to systems comprising of Mealy type finite-state machines and finite-state machines with non-isomorphic state to output mappings using the concept of A-equivalency between two systems. In the next section, a brief discussion about the Lyapunov exponents, developed specifically in the context of Markovian jump-linear systems by Boukas et al. [3], and their relationship with the mean-square stability of the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ is given. This chapter is concluded with an overview of the computation and dimensionality issues with regards to new stability tests.

3.2 Key Model Assumptions

In this section, a set of assumptions on the model class of jump-linear system driven by the finite-state machines with Markov inputs is presented. Later in the section, tests to validate these assumption will be provided. The tests are independent of the order of the Markovian input. Following model assumptions are essential for the stability theory under consideration:

1. The input process $\boldsymbol{\nu}$ is either an independent identically distributed process or

an irreducible Markov process of order r with $p_\eta > 0$ for all $\eta \in \Sigma_I$.

2. The finite-state machine is Moore type with an isomorphic state to output mapping, strongly connected, minimal and completely specified.
3. The cross-chain process, $\boldsymbol{\rho}$, has a unique stationary probability vector q.
4. The finite-state machine is *reachable* for the given input Markov process $\boldsymbol{\nu}$ with Π_I and hence the output probabilities $p_\xi > 0$ for all $\xi \in \Sigma_O$ always exist.

It is not necessary, in fact, for the finite-state machine always be a Moore type with an isomorphic state to output mapping. These type of machines are considered only as the initial focus. Many of the finite-state machine properties can be tested directly using the Definitions 2.1.2, 2.1.4 and 2.1.5 from Chapter 2. But, a set of theorems is needed to determine when the process $(\boldsymbol{\nu}, \boldsymbol{\theta})$ has a unique stationary probabilities, when the finite-state machine is reachable under the given Markovian process and when the probabilities p_ξ are positive. The following example illustrates what can happen in practice when some of the assumptions do not hold.

Consider the finite-state machine $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$ shown in Figure 3.2, where $\Sigma_I = \{\eta_1, \eta_2\}$, $\Sigma_S = \{e_1, e_2, e_3, e_4\}$, and $\Sigma_O = \{\xi_1, \xi_2, \xi_3, \xi_4\}$. The next state mapping δ is defined by

$$S_{\eta_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad S_{\eta_2} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

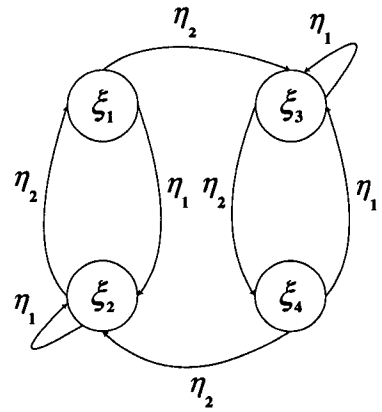

Figure 3.1: State diagram of a finite-state machine $\mathcal{M}$ driven by a Markov process that results into two ergodic classes.

The output mapping is $\omega(e_j) = \xi_j$, and the transition probability matrix for the input process $\boldsymbol{\nu}$ is

$$\Pi_I = \begin{bmatrix} 0.6 & 1 \\ 0.4 & 0 \end{bmatrix}.$$

The resulting cross chain $(\boldsymbol{\nu}, \boldsymbol{\theta})$ consists of two disjoint chains $\mathcal{C}_1$ and $\mathcal{C}_2$. $\mathcal{C}_1$ has the ergodic class $\mathcal{E}_1 = \{(\eta_1, \xi_1), (\eta_1, \xi_2), (\eta_2, \xi_2)\}$ and the non-ergodic class $\mathcal{N}_{\mathcal{E}_1} = \{(\eta_2, \xi_4)\}$. The chain $\mathcal{C}_2$ has the ergodic class $\mathcal{E}_2 = \{(\eta_1, \xi_3), (\eta_1, \xi_4), (\eta_2, \xi_3)\}$ and the non-ergodic class $\mathcal{N}_{\mathcal{E}_2} = \{(\eta_2, \xi_1)\}$. The corresponding transition probability matrix

for $(\boldsymbol{\nu}, \boldsymbol{\theta})$ is given by

$$\Pi_{I/O} = \left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0.6 & 0.6 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0.6 & 0.6 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

$\Pi_{I/O}$ has an infinite number of stationary probability vectors, specifically, linear combination of the following *two* stationary probability vectors:

$$q_1 = \left[\begin{array}{cccccccc} 0.2857 & 0.4286 & 0 & 0 & 0 & 0.2857 & 0 & 0 \end{array}\right]^T,$$

$$q_2 = \left[\begin{array}{cccccccc} 0 & 0 & 0.4286 & 0.2857 & 0 & 0 & 0.2857 & 0 \end{array}\right]^T.$$

Furthermore, if the process is always initialized in only one of the two ergodic sets, then it will never be able to enter the state in the other ergodic cycle. This implies that for the particular Π_I, the machine is not reachable and $p_\xi = 0$ for some $\xi \in \Sigma_O$.

The following tests provide sufficient conditions to ensure a unique stationary vector q, reachability of the finite-state machine under a Markovian input with transition matrix Π_I and $p_\xi > 0$ for all $\xi \in \Sigma_O$.

Theorem 3.2.1. *For a given finite-state machine $\mathcal{M}$ with Markovian input (Σ_I, Π_I), the process $(\boldsymbol{\nu}, \boldsymbol{\theta})$ is stationary with unique stationary probabilities if it is monodesmic.*

Proof: : If $(\boldsymbol{\nu}, \boldsymbol{\theta})$ is monodesmic then $\Pi_{I/O}$ is monodesmic, and by Theorem 2.2.2 (when $d = 1$) or Theorem 2.2.3 (when $d > 1$) it follows directly that there exists a unique stationary vector q such that $\Pi_{I/O} q = q$. ■

The next theorem provides an explicit test for the *reachability* of a finite-state machine $\mathcal{M}$ with Markovian input (Σ_I, Π_I). It also addresses the required assumption of $p_\xi > 0$.

Theorem 3.2.2. *A finite-state machine $\mathcal{M}$ with Markovian input (Σ_I, Π_I) is reachable and $p_\xi > 0$ for all $\xi \in \Sigma_O$, if the process $(\boldsymbol{\nu}, \boldsymbol{\theta})$ is ergodic, i.e., if the transition probability matrix $\Pi_{I/O}$ is irreducible.*

Proof: Since $\Pi_{I/O}$ is irreducible, it follows from Corollary 2.2.1 (if $d = 1$) or Corollary 2.2.2 (if $d > 1$), that a unique stationary probability $q > 0$ exists. Also, the unique stationary probability can be attained by any initial probability vector q_0. Since $p_\xi = \sum_{\eta \in \Sigma_I} q_{(\eta,\xi)}$, it follows that $p_\xi > 0$ for every $\xi \in \Sigma_O$. This implies that there exists a sequence of input symbols from Σ_I which occurs with nonzero probability and drives the finite-state machine from any initial state to any other state in Σ_S. This further implies $\mathcal{M}$ with Markovian input $\boldsymbol{\nu}$ is reachable, by definition. ■

As a consequence of Theorem 3.2.2 and the Definition 2.1.5, it follows that if the finite-state machine $\mathcal{M}$ is reachable under the Markovian input $\boldsymbol{\nu}$, then it is also strongly connected. Hence, it is sufficient to simply test for the reachability property. The following theorem is viewed as the main result of this section. It ties together all the previous results of this section and provides a single sufficient test for $\Pi_{I/O}$ to validate the model assumptions.

Theorem 3.2.3. *For a finite-state machine $\mathcal{M}$ with Markovian input $\boldsymbol{\nu}$, unique stationary probability vector q exists, the machine $\mathcal{M}$ is reachable, and state probabilities p_ξ are all positive, if the process $(\boldsymbol{\nu}, \boldsymbol{\theta})$ is ergodic, i.e., if the transition probability matrix $\Pi_{I/O}$ is irreducible.*

Proof: If $\Pi_{I/O}$ is irreducible, then it is also monodesmic. It follows that there exists unique strictly positive stationary probabilities $q_{(\eta,\xi)}$. From Theorem 3.2.2, $p_\xi > 0$ for all $\xi \in \Sigma_O$ and machine $\mathcal{M}$ is reachable. This completes the proof. ∎

3.3 Definitions of Stability for Jump-Linear Systems

Consider the jump-linear system

$$\boldsymbol{x}(k+1) = A_{\boldsymbol{\theta}(k)}\boldsymbol{x}(k) \tag{3.1}$$

with independent initial conditions $\boldsymbol{\theta}(0)$ and $\boldsymbol{x}(0) = x_0$. $\boldsymbol{\theta}$ is a discrete-time random process with state space Σ having N number of states. This jump-linear system will be denoted as $(A, \boldsymbol{\theta})$. The following definitions are frequently found in the stability literature for jump-linear systems.

Definition 3.3.1. *The jump-linear system (3.1) with discrete-time finite-state random process $\boldsymbol{\theta}$ is said to be* ***second-moment stable*** *or* ***mean-square stable****, if for any $x_0 \in \mathbb{R}^n$ and any initial probability distribution ψ of $\boldsymbol{\theta}(0)$,*

$$\lim_{k\to\infty} E\{\|\boldsymbol{x}(k)\|^2\} = 0$$

Definition 3.3.2. *The jump-linear system (3.1) is said to be* ***exponentially second-moment stable****, if for any* $x_0 \in \mathbb{R}^n$ *and any probability distribution of* $\boldsymbol{\theta}(0)$*, there exist constants* $\alpha, \beta > 0$ *independent of* x_0 *and such that*

$$E\{\|\boldsymbol{x}(k)\|^2\} \leq \alpha \|x_0\|^2 \exp^{-\beta k}, \forall k \geq 0.$$

Definition 3.3.3. *The jump-linear system (3.1) is said to be* ***stochastically second-moment stable****, if for any* $\bar{x}_0 \in \mathbb{R}^n$ *and any probability distribution of* $\boldsymbol{\theta}(0)$*,*

$$\lim_{k\to\infty} \sum_{k=0}^{\infty} E\{\|\boldsymbol{x}(k)\|^2\} < \infty.$$

Definition 3.3.4. *The jump-linear system (3.1) is said to be* ***almost surely (asymptotically) stable****, if for any* $x_0 \in \mathbb{R}^n$ *and any probability distribution of* $\boldsymbol{\theta}(0)$*,*

$$P\{\lim_{k\to\infty} \|\boldsymbol{x}(k)\| = 0\} = 1.$$

It is obvious that the stability definitions provided here can be used directly for the hybrid system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$. It is shown in [27] that for the jump-linear system $(A, \boldsymbol{\theta})$ with a finite-state time homogeneous Markov process $\boldsymbol{\theta}$, Definitions 3.3.1, 3.3.2 and 3.3.3 are all equivalent. It is also shown that in this case, second-moment stability implies almost sure (asymptotic) stability, but the converse is not always true. Thus, in this chapter, after developing the tools to test for the mean-square stability of the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, as a corollary, it will be shown that these tools can also be used to test for the exponential second-moment stability and stochastic second-moment stability of the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ even when $\boldsymbol{\theta}$ is *not* Markovian.

The authors of [8] provide an alternate definition for mean-square stability in the case of a finite-state time homogeneous process $\boldsymbol{\theta}$ and it has been consistently used by the researchers in [16, 52]. The definition follows.

Definition 3.3.5. *The jump-linear system (3.1) is* ***mean-square stable*** *if for any initial condition* $\boldsymbol{x}(0)$ *and for any initial state probability for* $\boldsymbol{\theta}(0)$ *it follows that*

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

where $Q(k) := E\{\boldsymbol{x}(k)\boldsymbol{x}^T(k)\}$, *or equivalently,*

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

where $\hat{Q}(k) := E\{\|\boldsymbol{x}(k)\|^2\}$ *(cf. [3, 52]).*

This definition happens to be more practical in many cases. For example, the researchers in [16, 52] found it is numerically more efficient to compute the statistics of $\hat{Q}(k) = trace(Q(k))$ than $\|Q(k)\|$. A clean mathematical proof showing Definition 3.3.1 and Definition 3.3.5 are equivalent can be found in [52].

3.4 Stability Analysis of the System Driven by ρ

In this section, the stability analysis of a jump-linear system driven by the cross-product of the Markovian input and the output of a Moore type finite-state machine is considered. Mean-square stability tests for such a system, when the input process is i.i.d., first-order Markov and higher order Markov are provided.

Consider the hybrid system shown in Figure 3.2. Here the jump-linear system

$$\hat{\boldsymbol{x}}(k+1) = \hat{A}_{\boldsymbol{\rho}(k)}\hat{\boldsymbol{x}}(k)$$

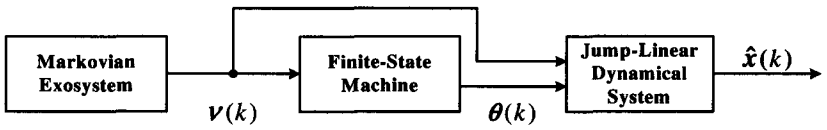

Figure 3.2: A jump-linear system driven by the process ρ.

has a driving process $\rho = (\nu, \theta)$. It is comprised of the r-th order input process ν and the corresponding output process θ from a finite-state machine $\mathcal{M}$. This system will be denoted as $(\nu, \mathcal{M}, \hat{A}, \rho)$. An alternate representation is to consider the jump-linear system

$$\hat{x}(k+1) = \hat{A}_{\rho_1(k)}\hat{x}(k)$$

that is being driven by the first-order Markov process $\rho_1 = (\nu_1, \theta)$ which is comprised of the first-order representation ν_1 of the r-th order input process ν and the output process θ, with $\hat{A}_{\rho_1(k)} = \hat{A}_{\rho(k)}$. This system will be denoted as $(\nu, \mathcal{M}, \hat{A}, \rho_1)$. The following theorem provides the stability test for the system $(\nu, \mathcal{M}, \hat{A}, \rho_1)$ when ν is an i.i.d. process.

Theorem 3.4.1. *The hybrid system $(\nu, \mathcal{M}, \hat{A}, \rho_1)$ is mean-square stable when ν is an i.i.d. whose probability distribution given by, $\pi = \begin{bmatrix} p_{\eta_1} & p_{\eta_2} & \cdots & p_{\eta_M} \end{bmatrix}^T$ is independent of the distribution of $\theta(0)$ and the initial state of the system $x(0)$, if and only if the matrix*

$$\mathcal{A}_0 := (\Pi_{I/O} \otimes I_{n^2})\, diag(\hat{A}_{\eta_1,\xi_1} \otimes \hat{A}_{\eta_1,\xi_1}, \ldots, \hat{A}_{\eta_M,\xi_N} \otimes \hat{A}_{\eta_M,\xi_N})$$

has a spectral radius less than one, where

$$\Pi_{I/O} = (\Pi_I \otimes I_N)\, diag(S_{\eta_1}, \ldots, S_{\eta_M}).$$

with $\Pi_I = \pi \times \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$.

Proof: The claim follows directly from well known results in [8]. The transition probability matrix for the process $\boldsymbol{\rho}_1$, namely $\Pi_{I/O}$, is obtained from equation (2.4).

■

The following theorem provides a test for mean-square stability conditions when $\boldsymbol{\nu}$ is a first-order Markov process.

Theorem 3.4.2. *The hybrid system* $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ *is mean-square stable when* $\boldsymbol{\nu}$ *is a first-order Markov process whose initial distribution is independent of* $\theta(0)$ *and the initial state of the system* $x(0)$*, if and only if the matrix*

$$\mathcal{A}_1 := (\Pi_{I/O} \otimes I_{n^2})\, diag(\hat{A}_{\eta_1,\xi_1} \otimes \hat{A}_{\eta_1,\xi_1}, \dots, \hat{A}_{\eta_M,\xi_N} \otimes \hat{A}_{\eta_M,\xi_N})$$

has a spectral radius less than one, where

$$\Pi_{I/O} = (\Pi_I \otimes I_N)\, diag(S_{\eta_1}, \dots, S_{\eta_M}).$$

Proof: Again the claim follows directly from well known results in [8]. The transition probability matrix for the process $\boldsymbol{\rho}_1$, namely $\Pi_{I/O}$, is obtained from equation (2.12).

■

Next Theorem 3.4.2 is generalized below to the case of a Markovian input of order $r \geq 1$.

Theorem 3.4.3. *The hybrid system* $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ *is mean-square stable when* $\boldsymbol{\nu}$ *is an* r*-th order Markov process whose initial distribution is independent of* $\theta(0)$ *and the*

initial state of the system $x(0)$*, if and only if the matrix*

$$\mathcal{A}_r := (\Pi_{I^r/O} \otimes I_{n^2})\, diag(\hat{A}_{\gamma_1,\xi_1} \otimes \hat{A}_{\gamma_1,\xi_1}, \ldots, \hat{A}_{\gamma_{M^r},\xi_N} \otimes \hat{A}_{\gamma_{M^r},\xi_N})$$

has a spectral radius less than one, where

$$\Pi_{I^r/O} = (\Pi_{I^r} \otimes I_N)\, diag(I_r \otimes (S_{\eta_1}, \ldots, S_{\eta_M}))$$

is the transition matrix of $\boldsymbol{\rho}_1$ *with* $\boldsymbol{\rho}_1(k) \in \underbrace{\Sigma_I \times \ldots \Sigma_I}_{r \text{ times}} \times \Sigma_S$ *and* Π_{I^r} *is the transition matrix of the first-order representation* $\boldsymbol{\nu}_1$ *with* $\boldsymbol{\nu}_1(k) = \gamma \in \underbrace{\Sigma_I \times \ldots \Sigma_I}_{r \text{ times}}$*, for the* r*-th order Markovian input process* $\boldsymbol{\nu}$*.*

Proof: The proof is based on Theorem 2.3.3, that is, if $\boldsymbol{\nu}$ is a Markov process of order r, then $\boldsymbol{\rho}_1 \in \underbrace{\Sigma_I \times \Sigma_I}_{\text{r times}} \times \Sigma_S$ is a first-order Markov process. The transition probability matrix $\Pi_{I^r/O}$ can be obtained from equations (2.8) and then apply the usual mean-square stability criterion for a first-order Markovian jump-linear case system in [8].

■

3.5 Stability Analysis of the System Driven by $\tilde{\theta}$

In this section, an alternative stability criterion is developed using jump-linear models driven by the output random process $\tilde{\boldsymbol{\theta}}$ of a *unifilar* finite-state machine. But the starting point is the special case where $\boldsymbol{\theta}$ is generated using a *non-unifilar* machine with an i.i.d. input process. As stated in Theorem 2.3.4, when the input to the finite-state machine $\boldsymbol{\nu}$ is i.i.d., its output $\boldsymbol{\theta}$ is a first-order Markov process. Therefore, the stability criterion in this case is a straight forward application of the method in [8], as given below.

Theorem 3.5.1. *Consider a hybrid system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ comprising of a jump-linear system driven by a finite-state machine $\mathcal{M}$ with an i.i.d. input process $\boldsymbol{\nu}$. Input process $\boldsymbol{\nu}$ has a distribution $\pi = \begin{bmatrix} p_{\eta_1} & p_{\eta_2} & \cdots & p_{\eta_M} \end{bmatrix}^T$ that is independent of the initial state of the machine $z(0)$, the output $\theta(0)$, and the initial state of the system $x(0)$. Then the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ is mean-square stable if and only if the matrix*

$$\mathcal{B}_1 := (\Pi_O \otimes I_{n^2})\, diag(A_{\xi_1} \otimes A_{\xi_1}, \ldots, A_{\xi_N} \otimes A_{\xi_N})$$

has a spectral radius less than one, where

$$\Pi_O = \sum_{\eta \in \Sigma_I} p_\eta S_\eta.$$

Proof: From Theorem 2.3.4, a Moore finite-state machine with an isomorphic state-to-output mapping, has an output process $\boldsymbol{\theta}$ which is a first-order Markov with transition probability as in equation (2.10), i.e., $P\{\boldsymbol{\theta}(k+1)|\boldsymbol{\theta}(k)\} = \sum_{\eta \in \mathcal{L}_k} P\{\boldsymbol{\nu}(k)\}$, where $\mathcal{L}_k = \{\eta \in \Sigma_I : \delta(\boldsymbol{\nu}(k), \boldsymbol{\theta}(k)) = \boldsymbol{\theta}(k+1)\}$. This implies directly that in terms of matrix notation, $\Pi_O = \sum_{\eta \in \Sigma_I} p_\eta S_\eta$. ∎

Now consider the case of a hybrid system comprising of a jump-linear system driven by the output process $\tilde{\boldsymbol{\theta}}$ of a unifilar finite-state machine with Markovian input of order $r \geq 1$, as shown in Figure 3.5. The jump-linear system is referred by

$$\tilde{\boldsymbol{x}}(k+1) = \tilde{A}_{\tilde{\boldsymbol{\theta}}(k)} \tilde{\boldsymbol{x}}(k),$$

and the hybrid system is denoted as $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$. Clearly, Theorem 3.5.1 is also applicable to the system $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ with i.i.d. inputs.

The stability analysis of $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ for the case of a Markovian input of order

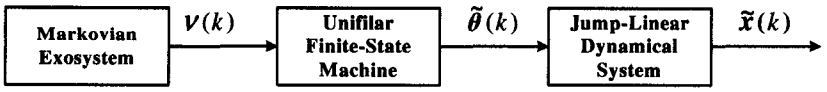

Figure 3.3: A jump-linear system driven by the output process $\tilde{\theta}$ of a unifilar finite-state machine.

$r = 1$ will be treated using a jump-linear system driven by a second-order Markov process. However, to avoid notational complexity, the higher order case will be treated using a jump-linear system driven by the first-order representation of $\tilde{\theta}$.

Theorem 3.5.2. *Consider the hybrid system $(\nu, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\theta})$, where $\tilde{\theta}$ is generated by a unifilar finite-state machine $\tilde{\mathcal{M}} = (\Sigma_I, \tilde{\Sigma}_O, \tilde{\Sigma}_S, \tilde{\delta}, \tilde{\omega})$ with a first-order Markov input process ν, which is independent of the initial machine state $\tilde{z}(0)$, the initial output $\theta(0)$ and the initial state of the system $x(0)$. Let $\tilde{S}_\eta : \tilde{\Sigma}_S \mapsto \tilde{\Sigma}_S$ be the state transition matrix for a given input $\eta \in \Sigma_I$ and $w(e_j) :\mapsto \tilde{\xi}_j$ be the isomorphic state to output map. The system is mean-square stable if and only if the matrix*

$$\mathcal{B}_2 = (\Pi_{O,2} \otimes I_{n^2})\, diag(I_{\tilde{N}} \otimes (\tilde{A}_{\tilde{\xi}_1} \otimes \tilde{A}_{\tilde{\xi}_1}), \ldots, I_{\tilde{N}} \otimes (\tilde{A}_{\tilde{\xi}_{\tilde{N}}} \otimes \tilde{A}_{\tilde{\xi}_{\tilde{N}}})) \tag{3.2}$$

has a spectral radius less than one, where $\Pi_{O,2}$ is a matrix of size $\tilde{N}^2 \times \tilde{N}^2$ composed of elementary block matrices of size $\tilde{N} \times \tilde{N}$ with each (J, I)-th block matrix having components

$$[[\Pi_{O,2}]_{JI}]_{ji} = \begin{cases} P\{\tilde{\xi}_J | \tilde{\xi}_i, \tilde{\xi}_I\} = P\{\eta_m | \eta_l\} & : \ I = j,\ \tilde{\delta}(\eta_l, e_I) = e_i,\ \tilde{\delta}(\eta_m, e_i) = e_J \\ 0 & : \ otherwise. \end{cases}$$

(Here $\tilde{N} = card(\tilde{\Sigma}_O)$ and, for brevity, probabilities like $P\{\nu(k+1) = \eta_m | \nu(k) = \eta_l\}$ are written as $P\{\eta_m | \eta_l\}$.)

Proof: By Theorem 2.3.6, under the stated conditions, $\tilde{\boldsymbol{\theta}}$ is a second-order Markov process. Observe that

$$Q(k) = E\{\tilde{\boldsymbol{x}}(k)\tilde{\boldsymbol{x}}^T(k)\} = \sum_{i,j} Q_{ij}(k),$$

where

$$Q_{ij}(k) := E\left\{\tilde{\boldsymbol{x}}(k)\tilde{\boldsymbol{x}}^T(k)\,\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k)=\tilde{\xi}_i\}}\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k-1)=\tilde{\xi}_j\}}\right\},$$

and $\mathbf{1}_{\{\cdot\}}$ denotes the Dirac function. Therefore,

$$\begin{aligned} Q_{ij}(k+1) &= \tilde{A}_{\tilde{\xi}_j} E\left\{\tilde{\boldsymbol{x}}(k)\tilde{\boldsymbol{x}}^T(k)\,\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k+1)=\tilde{\xi}_i\}}\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k)=\tilde{\xi}_j\}}\right\}\tilde{A}^T_{\tilde{\xi}_j} \\ &= \tilde{A}_{\tilde{\xi}_j} E\left\{E\left\{\tilde{\boldsymbol{x}}(k)\tilde{\boldsymbol{x}}^T(k)\,\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k+1)=\tilde{\xi}_i\}}\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k)=\tilde{\xi}_j\}}\middle|\,\mathcal{F}_k\right\}\right\}\tilde{A}^T_{\tilde{\xi}_j} \\ &= \tilde{A}_{\tilde{\xi}_j} E\left\{\tilde{\boldsymbol{x}}(k)\tilde{\boldsymbol{x}}^T(k)\,\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k)=\tilde{\xi}_j\}}P\{\tilde{\xi}_i|\tilde{\xi}_j,\tilde{\boldsymbol{\theta}}(k-1)\}\right\}\tilde{A}^T_{\tilde{\xi}_j}, \end{aligned}$$

where $\mathcal{F}_k$ denotes the σ-field generated by the random variables $\{\tilde{\boldsymbol{\theta}}(l), \tilde{\boldsymbol{x}}(l) : l = 0, 1, \ldots, k\}$. In addition,

$$\begin{aligned} &E\left\{\tilde{\boldsymbol{x}}(k)\tilde{\boldsymbol{x}}^T(k)\,\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k)=\tilde{\xi}_j\}}P\{\tilde{\xi}_i|\tilde{\xi}_j,\tilde{\boldsymbol{\theta}}(k-1)\}\right\} \\ &= \sum_l E\left\{\tilde{\boldsymbol{x}}(k)\tilde{\boldsymbol{x}}^T(k)\,\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k)=\tilde{\xi}_j\}}\mathbf{1}_{\{\tilde{\boldsymbol{\theta}}(k-1)=\tilde{\xi}_l\}}P\{\tilde{\xi}_i|\tilde{\xi}_j,\tilde{\xi}_l\}\right\}. \end{aligned}$$

Therefore,

$$Q_{ij}(k+1) = \tilde{A}_{\tilde{\xi}_j}\left(\sum_l P\{\tilde{\xi}_i|\tilde{\xi}_j,\tilde{\xi}_l\}Q_{jl}(k)\right)\tilde{A}^T_{\tilde{\xi}_j}.$$

Now apply the column stacking operator *vec* operator to produce

$$\begin{aligned} \vec{Q}_{ij}(k+1) &:= vec\left(\tilde{A}_{\tilde{\xi}_j}\sum_l P\{\tilde{\xi}_i|\tilde{\xi}_j,\tilde{\xi}_l\}Q_{jl}(k)\tilde{A}^T_{\tilde{\xi}_j}\right) \\ &= (\tilde{A}_{\tilde{\xi}_j}\otimes\tilde{A}_{\tilde{\xi}_j})\sum_l P\{\tilde{\xi}_i|\tilde{\xi}_j,\tilde{\xi}_l\}\vec{Q}_{jl}(k). \end{aligned}$$

Then it can be shown that

$$\vec{Q}(k+1) = \mathcal{B}_2 \vec{Q}(k), \tag{3.3}$$

where

$$\vec{Q}(k) := \left[\begin{array}{cccccccc} \vec{Q}_{11}^T(k) & \cdots & \vec{Q}_{1N}^T(k) & \cdots & \vec{Q}_{N1}^T(k) & \cdots & \vec{Q}_{NN}^T(k) \end{array} \right]^T,$$

$\Pi_{O,2}$ is obtained using equation (2.13) and $\mathcal{B}_2$ is as given in (3.2). Finally, the mean-square stability test for system $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ follows directly from the given spectral radius condition for the linear system (3.3). ∎

Next Theorem 3.5.2 is generalized for a jump-linear system driven by the output process of a unifilar finite-state machine $\tilde{\mathcal{M}}$ with Markovian input of any order r in the following manner:

Theorem 3.5.3. *Consider the hybrid system $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$, where $\tilde{\boldsymbol{\theta}}$ is generated by a unifilar finite-state machine $\tilde{\mathcal{M}} = (\Sigma_I, \tilde{\Sigma}_O, \tilde{\Sigma}_S, \tilde{\delta}, \tilde{\omega})$ with an r-th order Markov input process, which is independent of the initial machine state $\tilde{\boldsymbol{z}}(0)$, the initial output $\theta(0)$ and the initial state of the system $x(0)$. Let $\tilde{S}_\eta : \tilde{\Sigma}_S \mapsto \tilde{\Sigma}_S$ be the corresponding state transition matrix for a given input $\eta \in \Sigma_I$, $\tilde{w}(e_j) :\mapsto \tilde{\xi}_j$ be the isomorphic state to output map. The system is mean-square stable if and only if the matrix*

$$\mathcal{B}_{r+1} = (\Pi_{O,r+1} \otimes I_{n^2})\, diag(I_{\tilde{N}^r} \otimes (\tilde{A}_{\tilde{\xi}_1} \otimes \tilde{A}_{\tilde{\xi}_1}), \ldots, I_{\tilde{N}} \otimes (\tilde{A}_{\tilde{\xi}_{\tilde{N}}} \otimes \tilde{A}_{\tilde{\xi}_{\tilde{N}}}))$$

has a spectral radius less than one, where $\Pi_{O,r+1}$ is a matrix of size $\tilde{N}^{(r+1)} \times \tilde{N}^{(r+1)}$ composed of elementary block matrices of size $\tilde{N} \times \tilde{N}$ with each (J, I)-th block matrix

having components

$$[[\Pi_{O,r+1}]_{JI}]_{ji} = \begin{cases} P\{\tilde{\xi}_J|\tilde{\xi}_i, \tilde{\xi}_{I_1}, \tilde{\xi}_{I_2}, \ldots, \tilde{\xi}_{I_r}\} = P\{\eta_m|\eta_{l_1}, \ldots, \eta_{l_r}\} & : I_1 = j, \\ \tilde{\delta}(\eta_{l_r}, e_{I_r}) = e_{I_{r-1}}, \ \ldots, \ \tilde{\delta}(\eta_l, e_{I_1}) = e_i, \tilde{\delta}(\eta_m, e_i) = e_J, & \\ 0 & : otherwise. \end{cases}$$

(Here $\tilde{N} = card(\tilde{\Sigma}_O)$ *and, for brevity, probabilities like* $P\{\boldsymbol{\nu}(k+1) = \eta_m|\boldsymbol{\nu}(k) = \eta_{l_1}, \boldsymbol{\nu}(k-1) = e_{l_2}, \boldsymbol{\nu}(k-r) = e_{l_{r-1}}\}$ *are written as* $P\{\eta_m|\eta_{l_1}, \ldots, \eta_{l_{r+1}}\}$.*)*

Proof: Since the input to the Moore type finite-state machine is an r-th order Markov process, by Theorem 2.3.6, the output process $\boldsymbol{\theta}$ is a $(r+1)$-th order process. Now, consider the jump-linear system $(\tilde{A}, \tilde{\boldsymbol{\theta}}_1)$ described by the equation:

$$\tilde{\boldsymbol{x}}(k+1) = \tilde{A}_{\tilde{\boldsymbol{\theta}}_1(k)}\tilde{\boldsymbol{x}}(k). \tag{3.4}$$

Let $\tilde{\boldsymbol{\theta}}_1$ be the first-order representation of the $(r+1)$-th order process $\boldsymbol{\theta}$. Then its transition probabilities can be obtained from equation (2.11). It also implies $\tilde{A}_{\tilde{\boldsymbol{\theta}}_1(k)} = \tilde{A}_{\tilde{\boldsymbol{\theta}}(k)}$. Finally, apply the standard results from [8] to complete the proof. ■

3.6 Stability Equivalence

In this section, the mean-square stability conditions developed for the system $(\boldsymbol{\nu}, \mathcal{M}\hat{A}, \boldsymbol{\rho}_1)$ and $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ are employed to develop a stability test for the main system of interest, $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$. Later mean-square stability conditions developed for the case of Moore type finite-state machine with isomorphic state-to-output mapping are extended to the case of Mealy type machines as well as machines with non-isomorphic state-to-output mapping. Mean-square stability conditions developed for

the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ will also be related with its exponentially second-moment property and stochastically second-moment property. In order to relate the mean-square stability conditions of any two systems, the concept of *A-equivalency* is needed.

Definition 3.6.1. *Two hybrid systems comprising of jump-linear systems*

$$\boldsymbol{x}(k+1) = A_{\boldsymbol{\theta}(k)}\boldsymbol{x}(k)$$

and

$$\bar{\boldsymbol{x}}(k+1) = \bar{A}_{\bar{\boldsymbol{\theta}}(k)}\bar{\boldsymbol{x}}(k),$$

*driven by machines $\mathcal{M}$ and $\bar{\mathcal{M}}$, respectively, with the same Markov input process $\boldsymbol{\nu}$, denoted as $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ and $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ respectively, are said to be A-**equivalent** if*

$$A_{\boldsymbol{\theta}(k)} = \bar{A}_{\bar{\boldsymbol{\theta}}(k)}.$$

Note that A-equivalence does not imply that any A_{e_i} necessarily be equal to any $\bar{A}_{\bar{e}_j}$, or that the processes $\boldsymbol{\theta}$ and $\bar{\boldsymbol{\theta}}$ even take on the same symbols. But from a state evolution point of view, $\boldsymbol{x}(k) = \bar{\boldsymbol{x}}(k)$ for all $k > 0$ if $\boldsymbol{x}(0) = \bar{\boldsymbol{x}}(0)$. Based on the Definitions 2.1.6 and 3.6.1 the following claims are obvious.

Theorem 3.6.1. *Consider the two systems $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, $(\boldsymbol{\nu}, \bar{\mathcal{M}}, A, \boldsymbol{\theta})$ with finite-state machines $\mathcal{M} = \{\Sigma_I, \Sigma_S, \Sigma_O, \delta_1, \omega_1\}$ and $\bar{\mathcal{M}} = \{\Sigma_I, \Sigma_T, \Sigma_O, \delta_2, \omega_2\}$, respectively. The two systems $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ and $(\boldsymbol{\nu}, \bar{\mathcal{M}}, A, \boldsymbol{\theta})$ are A-equivalent if the machines $\mathcal{M}$ and $\bar{\mathcal{M}}$ are* equivalent.

Theorem 3.6.2. *Let $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ and $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ be two A-equivalent systems with two jump-linear systems $(A, \boldsymbol{\theta})$ and $(\bar{A}, \bar{\boldsymbol{\theta}})$ driven by the finite-state machines $\mathcal{M}$*

and $\bar{\mathcal{M}}$ *respectively with the same Markovian input process* $\boldsymbol{\nu}$. *Then* $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ *is mean-square stable if and only if* $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ *is mean-square stable.*

The following result establishes the equivalency between the mean-square stability conditions of $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ and $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$:

Theorem 3.6.3. *The hybrid system* $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ *with* $\hat{A}_{\eta_i,\xi} = \hat{A}_{\eta_j,\xi} =: A_\xi$, $\eta_i, \eta_j \in \Sigma_I$ *when input is i.i.d. or a first-order Markovian and* $\hat{A}_{\gamma_i,\xi} = \hat{A}_{\gamma_j,\xi} =: A_\xi$ *for all symbols* $\gamma_i, \gamma_j \in \underbrace{\Sigma_I \times \Sigma_I, \ldots, \Sigma_I}_{r \text{ times}}$ *when input is Markovian of order* $r > 1$ *is A-equivalent to a hybrid system* $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ *with the jump-linear system driven only by the output process* $\boldsymbol{\theta}$ *of the finite-state machine* $\mathcal{M}$ *with input* $\boldsymbol{\nu}$ *of order* r. *Therefore, under these conditions* $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ *is mean-square stable if and only if* $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ *is mean-square stable.*

Proof: The proof is just an application of Theorem 3.6.2. ∎

The next theorem shows that any finite-state machine has a unifilar companion machine that produces an A-equivalent jump-linear system. Therefore, Theorem 3.5.2 can be applied to this new system in order to determine the mean-square stability of the original system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$. There is, however, some *overhead* involved in determining the unifilar equivalent system as described in the proof.

Theorem 3.6.4. *Let* $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$ *be an arbitrary non-unifilar finite-state machine with input* $\boldsymbol{\nu}$ *and output* $\boldsymbol{\theta}$. *For any corresponding hybrid system* $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, *there always exists an A-equivalent system* $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$, *where* $\tilde{\theta}$ *is generated by a unifilar finite-state machine* $\tilde{\mathcal{M}} = (\Sigma_I, \tilde{\Sigma}_S, \tilde{\Sigma}_O, \tilde{\delta}, \tilde{\omega})$ *with input* $\boldsymbol{\nu}$.

Proof: The proof is constructive and based on the proof of Theorem 2.1.1 which states that for every non-unifilar finite-state machine, there always exists an equivalent unifilar type finite-state machine. However, the goal here is to produce an A-equivalent system $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ for the given system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, rather than merely preserving the input-output relationship for the finite-state machine. For the sake of clarity and completeness, steps in the proof for Theorem 2.1.1 are repeated.

Since $\mathcal{M}$ is non-unifilar, there exists a pair of inputs symbols η_i and η_j and a machine state e_m and e_l such that $\delta(\eta_i, e_m) = \delta(\eta_j, e_m) = e_l$. Therefore, augment the machine's state space, Σ_S, by replacing the state e_l with new states e_{l_i} and e_{l_j}. Define a corresponding new transition map $\tilde{\delta}$, which is identical to δ except now: $\tilde{\delta}(\eta_i, e_m) = e_{l_i}$, $\tilde{\delta}(\eta_j, e_m) = e_{l_j}$, and $\tilde{\delta}(\eta, e_{l_i}) = \delta(\eta, e_{l_j}) = \delta(\eta, e_l)$ for all input symbols η. Similarly, redefine the output function $\tilde{\omega}$ to be identical to ω except, it maps e_{l_i} to $\tilde{\xi}_l$ and e_{l_j} to the output symbol $\tilde{\xi}_k$, where k is just the next available index for the new output symbol $\tilde{\xi}$ that avoids any duplication of symbols. Next define for every output symbol the transition matrix $\tilde{A}_{\tilde{\xi}_l} = \tilde{A}_{\tilde{\xi}_k} = A_{\xi_l}$. It can then be verified that the systems $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ and $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ are A-equivalent, i.e.,

$$\tilde{A}_{\tilde{\omega}(\tilde{\boldsymbol{z}}(k))} = A_{\omega(\boldsymbol{z}(k))}.$$

It is clear that the states and output symbols of the new machine can be re-indexed by the integers $1, 2, \ldots, N+1$, and the whole process above repeated if the new machine is not unifilar. Since the number of input symbols and machine states is finite, this procedure need only be repeated a finite number of times before a unifilar

machine is produced. ■

Next consider a hybrid system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ with a jump-linear system $(\bar{A}, \bar{\boldsymbol{\theta}})$ driven by an output process $\bar{\boldsymbol{\theta}}$ of a Moore type finite-state machine with non-isomorphic many-to-one mapping. Specifically, let $\bar{\mathcal{M}} = \{\Sigma_I, \bar{\Sigma}_S, \bar{\Sigma}_O, \bar{\delta}, \bar{\omega}\}$ with $\bar{\Sigma}_S = \{e_1, e_2, \ldots, e_N\}$, $\bar{\Sigma}_O = \{\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_L\}$, state transition function $\delta : \Sigma_S \times \Sigma_I \mapsto \Sigma_S$, output function $\bar{\omega}(e_j) = \bar{\xi}_l, \quad l = 1, \ldots, L$. In order to determine the mean-square stability conditions of the system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$, it will be shown that an A-equivalent system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ with a Moore finite-state machine having an isomorphic state-to-output mapping can always be produced. Hence, by Theorem 3.6.2 $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ will be mean-square stable if and only if $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ is mean-square stable.

Theorem 3.6.5. *For every hybrid system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ with a jump-linear system $(\bar{A}, \bar{\boldsymbol{\theta}})$ driven by the output process $\bar{\boldsymbol{\theta}}$ of a Moore type finite-state machine, $\bar{\mathcal{M}}$, having a non-isomorphic many-to-one state-to-output mapping, there always exists an A-equivalent system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ with a jump-linear system $(A, \boldsymbol{\theta})$ driven by the output process $\boldsymbol{\theta}$ of a Moore type finite-state machine, $\mathcal{M}$, with an isomorphic state-to-output map.*

Proof: Consider a new machine $\mathcal{M}$ with the same state set $\Sigma_S = \bar{\Sigma}_S$, the same state transition function $\delta = \bar{\delta}$, and a new output set $\Sigma_O = \{\xi_1, \xi_2, \ldots, \xi_N\}$ and new output function $\omega(e_j) = \xi_j$. Clearly, the new machine $\mathcal{M}$ is a Moore finite-state machine with an isomorphic state to output mapping. Next define for each output symbol a transition matrix $A_{\xi_j} = A_{\omega(e_j)} = \bar{A}_{\bar{\omega}(e_j)} = \bar{A}_{\bar{\xi}_l}$, where $\bar{\omega}(e_j)$ is the output function of the finite-state machine $\bar{\mathcal{M}}$. This clearly implies that the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ with

the jump-linear system $(A, \boldsymbol{\theta})$ driven by the output process $\boldsymbol{\theta}$ of finite-state machine $\mathcal{M}$ is A-equivalent to the system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$, and thus the proof. ∎

Recall that in the case of fault recovery applications, when system transition behavior is dependent on both the type of upset encountered and the current recovery state, the recovery algorithms can be modeled as Mealy type finite-state machines. For such systems, in the most general case with a non-isomorphic mapping between the states and outputs, the output function evolves according to the expression $\bar{\boldsymbol{\theta}}(k) = \bar{\omega}(\boldsymbol{\nu}(k), \boldsymbol{z}(k))$, and hence, the closed-loop system parameter $A_{\xi_l} = A_{\bar{\omega}(\eta_i, e_j)}$. Therefore, now consider the hybrid system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ with jump-linear system $(\bar{A}, \bar{\boldsymbol{\theta}})$ driven by the output process $\bar{\boldsymbol{\theta}}$ of a Mealy type finite-state machine with non-isomorphic state-to-output mapping $\bar{\mathcal{M}}$. Let $\bar{\mathcal{M}} = \{\Sigma_I, \bar{\Sigma}_S, \bar{\Sigma}_O, \bar{\delta}, \bar{\omega}\}$ with $\bar{\Sigma}_S = \{e_1, e_2, \ldots, e_N\}$, $\bar{\Sigma}_O = \{\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_L\}$, state transition function $\delta : \Sigma_S \times \Sigma_I \mapsto \Sigma_S$, and output function $\bar{\omega}(\eta_i, e_j) = \bar{\xi}_l, \quad l = 1, \ldots, L$. In order to determine the mean-square stability conditions of the system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$, it will be shown that an A-equivalent system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ with a jump-linear system $(A, \boldsymbol{\theta})$ driven by the Moore type finite-state machine $\mathcal{M}$ with an isomorphic state-to-output mapping can always be produced.

Theorem 3.6.6. *For every hybrid system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$ with a jump-linear system $(\bar{A}, \bar{\boldsymbol{\theta}})$ driven by the output process $\bar{\boldsymbol{\theta}}$ of a Mealy type finite-state machine, $\bar{\mathcal{M}}$, having a non-isomorphic state-to-output mapping, there always exists an A-equivalent system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ with a jump-linear system $(A, \boldsymbol{\theta})$ driven by the output process $\boldsymbol{\theta}$ of a Moore type finite-state machine, $\mathcal{M}$, with an isomorphic state-to-output map.*

Proof: Define a new machine $\mathcal{M}$ with $\Sigma_S = \{\gamma_1, \ldots, \gamma_{MN}\}$, such that Σ_S is an

isomorphism of $\bar{\Sigma}_I \times \bar{\Sigma}_S$, i.e., there exists an invertible map T such that $T(\eta_i e_m) = \gamma_j$. Let $\Sigma_O = \{\xi_1, \xi_2, \ldots, \xi_{MN}\}$ and output function $\omega : \Sigma_S \mapsto \Sigma_O$ be an an isomorphic mapping $\omega(\gamma_j) = \xi_j$. Clearly, the new finite-state machine $\mathcal{M}$ is a Moore finite-state machine with an isomorphic state-to-output mapping. Next define for each output symbol a transition matrix $A_{\xi_j} = A_{\omega(\gamma_j)} = \bar{A}_{\bar{\omega}(\eta_i, e_m)} = \bar{A}_{\bar{\xi}_l}$. This implies that the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ with the jump-linear system $(A, \boldsymbol{\theta})$ driven by the output process $\boldsymbol{\theta}$ of Moore type finite-state machine, $\mathcal{M}$, with an isomorphic state-to-output mapping is A-equivalent to the system $(\boldsymbol{\nu}, \bar{\mathcal{M}}, \bar{A}, \bar{\boldsymbol{\theta}})$, and hence the proof. ∎

Now based on Theorem 3.6.3, 3.6.4, 3.6.5 and 3.6.6 following theorem is obvious, and summarizes the important contribution of this thesis. It is also illustrated in Figure 3.6.

Theorem 3.6.7. *For any given hybrid system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, there always exist A-equivalent systems $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho})$ and $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$. Therefore, the given system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ is mean-square stable if and only if its A-equivalent systems $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho})$ and $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ are mean-square stable.*

Now, using the concept of A-equivalency it is also possible to relate the mean-square stability property of the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, with its exponentially second-moment stability property and stochastically second-moment stability property.

Theorem 3.6.8. *The hybrid system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ with a jump-linear system driven by an arbitrary finite-state machine with Markovian input is mean-square stable if and only if it is exponentially second-moment stable, if and only if it is stochastically second-moment stable.*

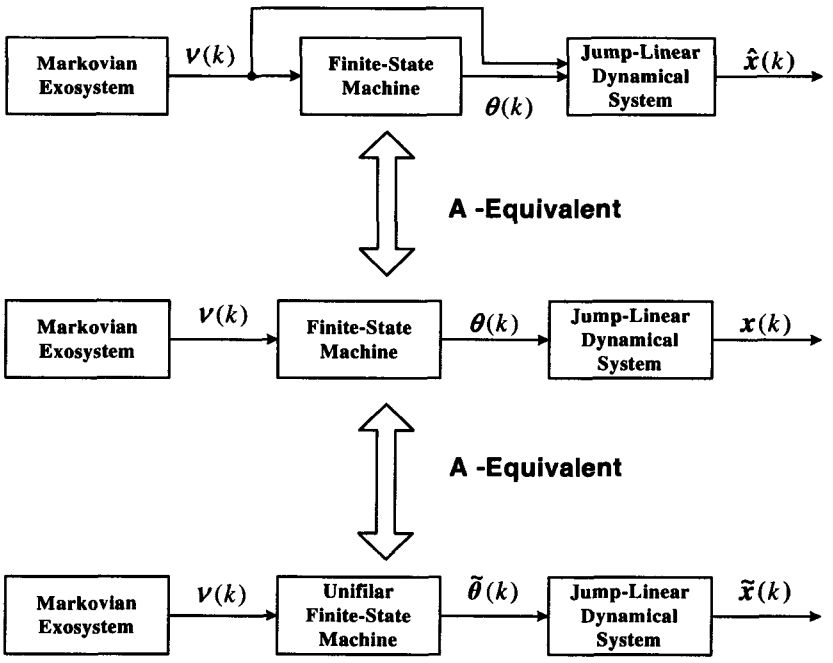

Figure 3.4: A-equivalency of the systems $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho})$, $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ and $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$.

Proof: Proof is based on the result from [27] that for the system $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ with Markovian $\boldsymbol{\rho}_1$, mean-square stability, exponential stability and stochastically second-moment stability property are equivalent. Initially, consider the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ to be mean-square stable. Then by Definition 3.3.1, for any $x_0 \in \mathbb{R}^n$ and any initial probability distribution of $\boldsymbol{\theta}(0)$,

$$\lim_{k\to\infty} E\{\|\boldsymbol{x}(k)\|^2\} = 0.$$

This implies that the corresponding A-equivalent system $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ with Markovian $\boldsymbol{\rho}_1$ is mean-square stable and hence

$$\lim_{k\to\infty} E\{\|\hat{\boldsymbol{x}}(k)\|^2\} = 0$$

for every $x_0 = \hat{x}_0$. Then by Theorem 1 of [27], it follows that there always exist constants $\alpha, \beta > 0$ independent of $\hat{x}_0$ such that

$$E\{\|\hat{\boldsymbol{x}}(k)\|^2\} \leq \alpha \|\hat{x}_0\|^2 \exp^{-\beta k}, \forall k \geq 0.$$

This in turn implies, for the A-equivalent system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$,

$$E\{\|\boldsymbol{x}(k)\|^2\} \leq \alpha \|x_0\|^2 \exp^{-\beta k}, \forall k \geq 0, \text{for } x_0 = \hat{x}_0.$$

Hence the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ is exponentially second-moment stable. In a similar manner it follows that if the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ is exponentially second-moment stable, it implies that the corresponding A-equivalent system $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ is exponentially second-moment stable. Therefore A-equivalent systems $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ and $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ are mean-square stable. In a very similar manner equivalence between stochastically second-moment stability property and mean-square stability property of $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ can be ascertained. ■

3.7 Characterizing Mean-Square Stability via the Second Lyapunov Exponent

The concept of Lyapunov exponents for characterizing the stability of dynamical systems originated in the seminal work by A. Lyapunov [34]. Later Arnold developed an extensive theory of Lyapunov exponents for random dynamical systems. In this context, Lyapunov exponents are functions that capture the essence of moment stability of random dynamical system. Thus, second Lyapunov exponent as given

below was applied specifically to characterize the mean-square stability of Markovian jump-linear systems by Boukas et.al. in [3].

Definition 3.7.1. *The **second Lyapunov exponent** for a discrete-time jump-linear system is defined as the function $\lambda_s(\boldsymbol{x}_0)$, given by,*

$$\lambda_s(\boldsymbol{x}_0) = \limsup_{k\to\infty} \frac{1}{k} \log[E\{\|\boldsymbol{x}(k, \boldsymbol{x}_0, \boldsymbol{\theta}(0))\|^2\}], \boldsymbol{x}_0 \neq 0. \tag{3.5}$$

It has been shown in [3] and [52] that, for the discrete-time jump-linear system (3.1) driven by homogeneous first-order Markaov process $\boldsymbol{\theta}$, the second Lyapunov exponent is a constant number, given by $\lambda_s(\boldsymbol{x}_0) = log(\rho(\mathcal{A}))$, where ρ denotes the spectral radius of

$$\mathcal{A} = (\Pi_I \otimes I_{n^2})\ diag(A_0 \otimes A_0, \ldots, A_N \otimes A_N),$$

with Π_I being the transition probability matrix for the Markov process $\boldsymbol{\theta}$. From the Definition 3.7.1, it is clear that the second-moment of the given jump-linear system approaches the second Lyapunov exponent asymptotically. Therefore, the second Lyapunov exponent can be used as a measure to compare the robustness of mean-square stability of two jump-linear systems. Now, it is expected that, under the same set of initial conditions and identical Markov input conditions, the second Lyapunov exponent of three A-equivalent systems $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ and $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ are equal. More formal theorem follows:

Theorem 3.7.1. *Consider a fixed initial state $\boldsymbol{z}(0)$, and hence the fixed initial output $\boldsymbol{\theta}(0)$ of a given finite-state machine $\mathcal{M}$ with the Markovian input $\boldsymbol{\nu}$ of order r, whose initial distribution is independent of $\boldsymbol{x}_0$. The second Lyapunov exponent $\lambda_s(\boldsymbol{x}_0)$*

for the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ *is equal to* $\log(\rho(\mathcal{A}_r)) = \log(\rho(\mathcal{B}_{r+1}))$ *when* $\boldsymbol{x}_0 \neq 0$, *if* $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ *and* $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$ *are A-equivalent to the system* $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$.

Proof: For the system $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ with Markovian $\boldsymbol{\rho}_1$, based on the results in [3] and [52],

$$\begin{aligned} \lambda_s(\hat{\boldsymbol{x}}_0) &= \limsup_{k\to\infty} \frac{1}{k} \log[E\{\|\hat{\boldsymbol{x}}(k, \hat{\boldsymbol{x}}_0, \boldsymbol{\rho}_1(0))\|^2\}], \ \hat{\boldsymbol{x}}_0 \neq 0 \\ &= \log(\mathcal{A}_r) \end{aligned}$$

for any $\hat{\boldsymbol{x}}_0 \neq 0$. Whenever $(\boldsymbol{\nu}, \mathcal{M}, \hat{A}, \boldsymbol{\rho}_1)$ is A-equivalent to the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$, for any $\boldsymbol{x}_0 = \hat{\boldsymbol{x}}_0 \neq 0$,

$$E\{\|\hat{\boldsymbol{x}}(k, \hat{\boldsymbol{x}}_0, \boldsymbol{\rho}(0))\|^2\} = E\{\|\boldsymbol{x}(k, \boldsymbol{x}_0, \boldsymbol{\theta}(0))\|^2\}$$

Therefore,

$$\begin{aligned} \lambda_s(\boldsymbol{x}_0) &= \limsup_{k\to\infty} \frac{1}{k} \log[E\{\|\boldsymbol{x}(k, \boldsymbol{x}_0, \boldsymbol{\theta}(0))\|^2\}], \ \boldsymbol{x}_0 \neq 0 \\ &= \log(\rho(\mathcal{A}_r)). \end{aligned}$$

In a very similar fashion, under the conditions of A-equivalency between the systems $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ and $(\boldsymbol{\nu}, \tilde{\mathcal{M}}, \tilde{A}, \tilde{\boldsymbol{\theta}})$, it can be shown that

$$\begin{aligned} \lambda_s(\boldsymbol{x}_0) &= \limsup_{k\to\infty} \frac{1}{k} \log[E\{\|\boldsymbol{x}(k, \boldsymbol{x}_0, \boldsymbol{\theta}(0))\|^2\}], \ \boldsymbol{x}_0 \neq 0 \\ &= \log(\rho(\mathcal{B}_{r+1})), \end{aligned}$$

and hence the proof. ■

3.8 Dimensionality Issues in the Tests for Mean-Square Stability of the System $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$

In this section, some dimensionality, and hence the computational issues, associated with the tests for mean-square stability of the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ developed in the previous sections will be discussed. Observe that in general the numerical efficiency of mean-square stability conditions presented in the previous sections, more specifically the size of matrices $\mathcal{A}_r$, $\mathcal{B}_{r+1}$ are dependent on following factors: the number of input symbols M, the order of the Markovian input r, the number of states of the finite-state machine N, and the nature of the finite-state machine, whether it is unifilar or non-unifilar. The size of the matrix $\mathcal{A}_r$, denoted as $dim(\mathcal{A}_r)$ is $M^r \times N \times n^2$. It is independent of whether the given machine is of type unifilar or non-unifilar.

If the given machine is of unifilar type, the dimension of $dim(\mathcal{B}_{r+1}) = N^{r+1} \times n^2$ is independent of number of input symbols, and only depends on the number of states and the order of the Markovian input. Figure 3.8 shows that for $M < N$, $dim(\mathcal{A}_r) < dim(\mathcal{B}_{r+1})$. At $M = N$, clearly they are equal. $M > N$ is not possible since the given machine is already unifilar. Thus, it is always advantageous to use cross-chain driven equivalent jump-linear system to determine the mean-square stability when given machine is already of unifilar type.

If the given machine is non-unifilar, $dim(\mathcal{B}_{r+1}) = \tilde{N}^{(r+1)} \times n^2$. Let $p \leq M$ be the bound on the number of input symbols that cause the same state transition. Then the number of states in the equivalent unifilar type finite-state machine $\tilde{N}$ is dependent on p and N. More precisely, $(N-1)+p \leq \tilde{N} \leq pN$. In the worst case, if $p = M$,

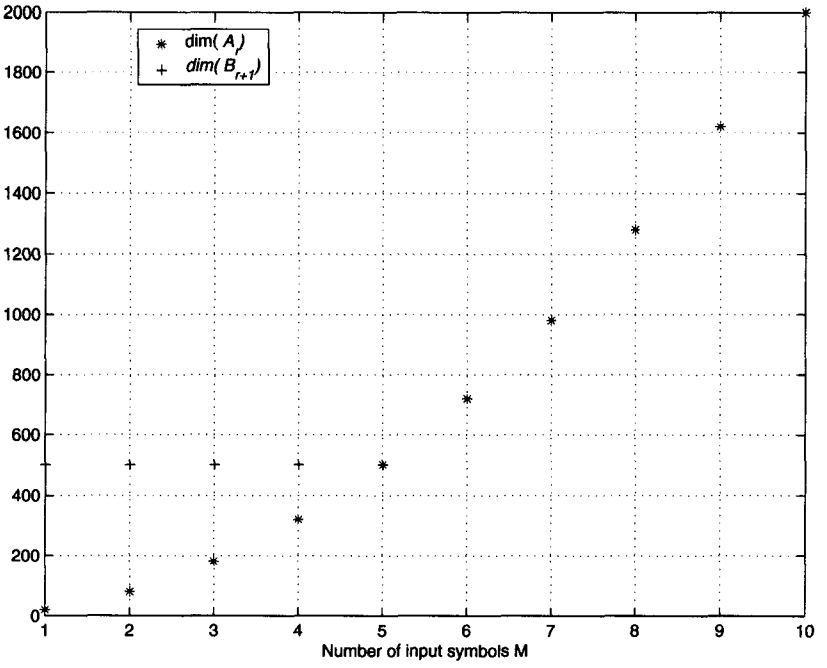

Figure 3.5: Plot of $dim(\mathcal{A}_r)$ and $dim(\mathcal{B}_{r+1})$ for a unifilar finite-state machine with N=5 for various number of inputs M.

then $(N-1)+M \leq \tilde{N} \leq MN$. (The best case, of course is when $\tilde{N} = N$ and it is already unifilar.) Figure 3.8 illustrates that in the non-unifilar case, there are situations when it is more advantageous to use the stability criterion derived from an equivalent jump-linear system driven by the state process from the corresponding unifilar type machine over the one derived from cross-chain driven equivalent jump-linear system. For example, clearly when $r = 2$, $p = 2$, $M = 10$, the number of states in the non-unifilar machine is 5 and number of states in the equivalent unifilar machine is $\tilde{N} = N - 1 + p = 5 - 1 + 2$, and hence $dim(\mathcal{B}_3) < dim(\mathcal{A}_2)$.

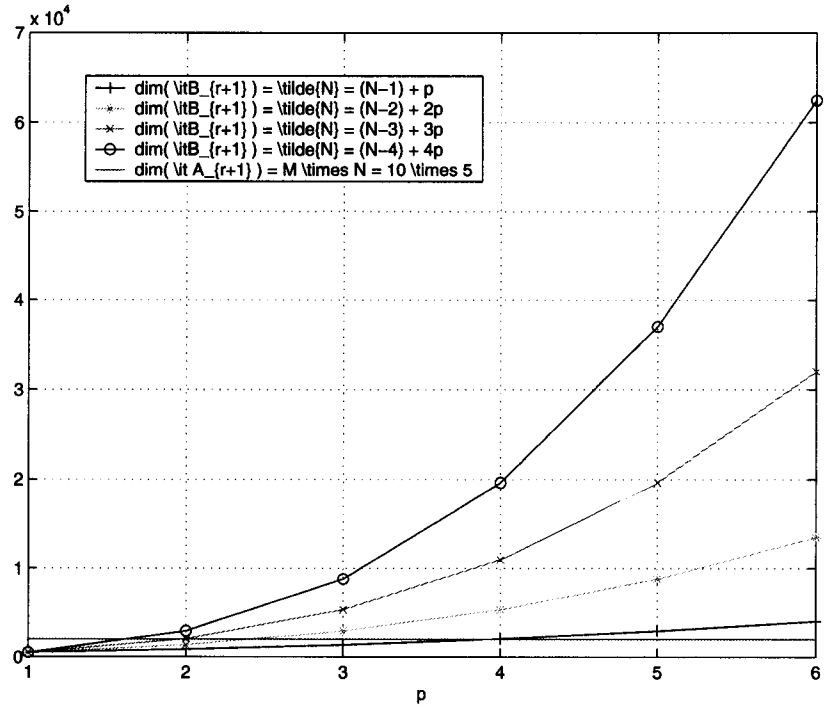

Figure 3.6: Plot of $dim(\mathcal{B}_{r+1})$ for various values of $\tilde{N}$ and $dim(\mathcal{A}_r)$, as p varies for a given non-unifilar finite-state machine. M is fixed to 10 and $r = 2$.

3.9 Summary of Key Points

In this chapter, key assumptions about the various components of the model under consideration were stated, and tests to validate the assumptions were provided. Next, appropriate stability terms and definitions for a jump-linear system driven by an arbitrary random process were introduced. Two approaches to determine the mean-square stability of the jump-linear system driven by the output of a Moore finite-state machine with an isomorphic state-to-output mapping and a Markovian input were developed. The first approach was based on analyzing an equivalent jump-linear

system driven by the input and the output of the finite-state machine, where as second approach was based on analysis of an equivalent jump-linear system driven by the output of a unifilar finite-state machine. Later, methods to determine the mean-square stability of jump-linear systems with Mealy finite-state machines as well as machines with non-isomorphic state-to-output mapping were also demonstrated. Thus it was possible to derive the mean-square stability conditions for a jump-linear system driven by any arbitrary finite-state machine with Markovian inputs of any order r. It was also shown that the mean-square stability conditions are the same as those for exponential second-moment stability and stochastic second-moment stability of the system. Lyapunov exponents were then used to characterize the mean-square stability of the system. A brief discussion of dimensionality and computational issues involved in using these results concluded this chapter.

CHAPTER IV

SIMULATION STUDIES

4.1 Introduction

In this chapter, the mean-square stability criteria for a hybrid model $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ are demonstrated using three simple examples involving recovery algorithms. The first example is a recovery algorithm with two modes, where the recovery duration is fixed and the statistics of the upset process are varied. Three cases of upset models are considered: i.i.d., first-order and second-order Markov. The second example is similar to the first one, but illustrates the mean-square stability property changes with respect to a variation in the recovery duration for the first-order Markov input. The third example examines the mean-square stability of a closed-loop system employing the complex recovery algorithm shown in Figure 1.6 when the upset process is first-order Markov.

4.2 Demonstration Procedure for the Mean-Square Stability Criteria of the System $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$

There are two main steps involved in the demonstration procedure of the mean-square stability criteria: producing a theoretical prediction of the stability boundary

and then verifying the prediction via Monte Carlo simulation. The procedure can be further subdivided into the following steps:

1. Verify whether the assumptions in Theorem 3.2.3 needed to apply the mean-square stability tests are valid.
2. For a Markovian input of order r, produce plots of the spectral radii of $\mathcal{A}_r$ and $\mathcal{B}_{r+1}$ as a function of the system parameters and identify the mean-square stability boundary.
3. Initialize the state of the Markov chain, the initial state of the finite-state machine, i.e., the initial mode of the recovery algorithm and the initial state of the closed-loop dynamical system, independently.
4. Simulate the Markov chain that models the upset process for a given transition probability matrix Π_I.
5. Simulate the given recovery algorithm modeled as a finite-state machine.
6. Simulate the closed-loop dynamical system during each mode of the recovery cycle.
7. Compute the second moment $\hat{Q}$ of the state of the closed-loop dynamical system by averaging at each instance k over the given number of Monte Carlo runs.
8. Compare the statistics of $\hat{Q}$ with the expected theoretical results predicted using $\rho(\mathcal{A}_r)$ and $\rho(\mathcal{B}_{r+1})$.
9. Plot $k \times log(\rho(\mathcal{A}_r))$ to characterize the mean-square stability prediction and compare it to $log(\hat{Q})$.

4.3 Three Recovery System Examples

In this section, three simple recovery examples implemented with scalar dynamics are given to illustrate the mean-square stability tests developed in the previous chapter. For the first two examples, a simple recovery algorithm is considered where the upsets are assumed to be not affecting the recovery cycle and the system stays in the recovery mode for a specified duration. In the first example, upsets will be modeled as Markov processes of different orders and the resulting effect on the mean-square stability conditions is illustrated. In the second example, the effect of changing the recovery duration on mean-square stability is illustrated. The third example illustrates the mean-square stability tests for the more complex recovery algorithm introduced in Chapter 1, Figure 1.6, where upsets are modeled as a first-order Markov process.

4.3.1 Recovery Example 1

Consider a closed-loop control system implemented on a recoverable computer. Perhaps the simplest model for such a system is shown in Figure 4.1. As long as there is no computer upset, the system operates in the *normal* mode, as per the dynamics

$$x(k+1) = A_0 x(k).$$

As soon as an upset occurs, the recovery system places the computer in the *recovery* mode for a fixed duration of $M_R = 2$ clock cycles. During the recovery process the system dynamics are described by

$$x(k+1) = A_1 x(k).$$

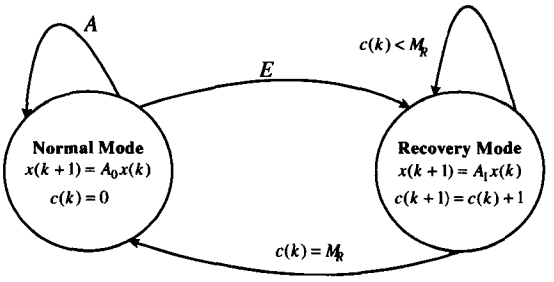

Figure 4.1: A simple model for a recoverable computer control system.

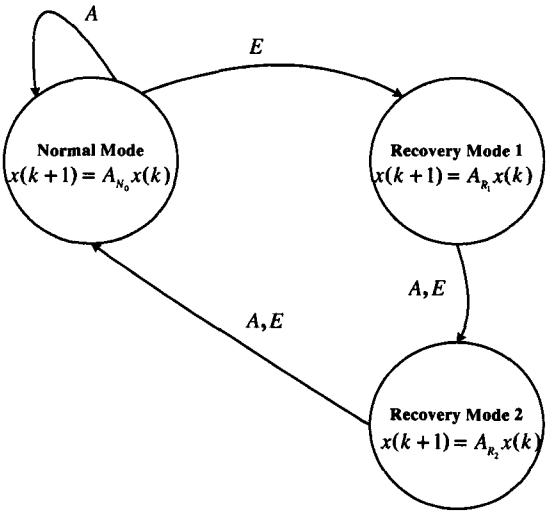

Figure 4.2: A finite-state machine representation of a recoverable computer control system.

After this duration, the nominal dynamics are restored. Here the counter, $c(k)$, keeps track of the lapse-time for the recovery process. To model this system as a finite-state machine, a new machine state must be introduced for each possible counter value. The state diagram for such a machine is shown in Figure 4.2. It is clearly not

unifilar. Specifically, $\mathcal{M} = (\Sigma_I, \Sigma_S, \Sigma_O, \delta, \omega)$ with $\Sigma_I = \{A, E\}$, $\Sigma_S = \{e_1, e_2, e_3\}$, $\Sigma_O = \{N_0, R_1, R_2\}$, δ is defined by

$$S_A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad S_E = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

and $\omega(e_j) = \xi_j$. The corresponding jump-linear system is specified by setting $A_{N_0} = A_0$ and $A_{R_1} = A_{R_2} = A_1$. For the purpose of validation, assume the Markovian inputs to the finite-state machine to be either i.i.d., first-order or second-order Markov.

When the input $\boldsymbol{\nu}$ is i.i.d. the stability Theorems 3.4.1 and 3.5.1 can be applied directly by setting $A_{A,N_0} = A_{E,N_0} = A_0$ and $A_{A,R_1} = A_{E,R_1} = A_{A,R_2} = A_{E,R_2} = A_1$. The system and simulation parameters are shown in Table A-1. The upset process is characterized by the probabilities which is varied between 0 and 1. The spectral radii of $\mathcal{A}_0$ and $\mathcal{B}_1$ as a function of Π_E is plotted in Figure 4.3, which predicts the mean-square stability boundary to be at $\Pi_E = 0.5$. Therefore two values of Π_E, 0.45 and 0.55 were chosen for Monte Carlo simulation, predicting that the closed-loop system will be mean-square stable at $p_E = 0.45$ and not mean-square stable at $p_E = 0.55$. As shown in Figure 4.3.1, the simulations agree with the theoretical predictions. In addition, the second Lyapunov exponent number $\log_{10}(\rho(\mathcal{A}_0)) = \log_{10}(\rho(\mathcal{B}_1))$ is an asymptote for the statistics $log_{10}(\hat{Q})$.

Now suppose, the upset process $\boldsymbol{\nu}$ is modeled as a Markovian process of first-order. The system and simulation parameters are given in Table A-2. Assuming transition probabilities $\Pi_{A|A} = 0.45$, the spectral radius of $\mathcal{A}_1$ is plotted in Figure 4.3.1 as a function of $\Pi_{E|E}$. It predicts the mean-square stability boundary for the system to

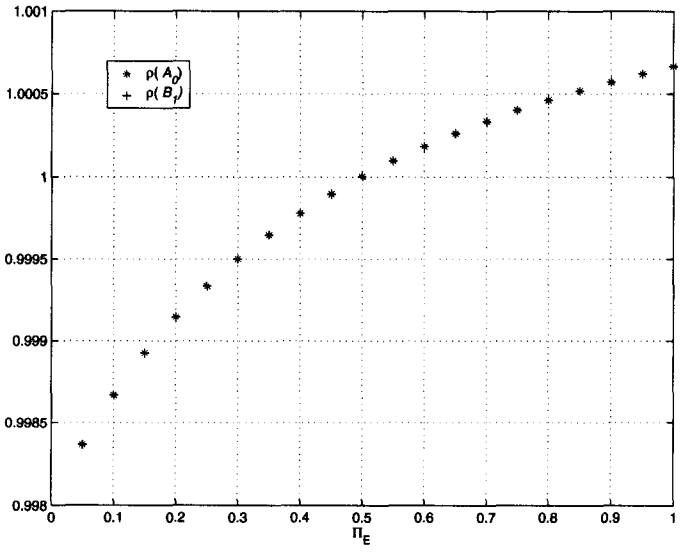

Figure 4.3: The spectral radius of $\mathcal{A}_0$ and $\mathcal{B}_1$ as a function of Π_E for the Recovery Example 1.

be at $\Pi_{E|E} = 0.35$.

To illustrate the stability test in Theorem 3.5.2, an A-equivalent unifilar finite-state machine is constructed as shown in Figure 4.3.1. The new machine $\tilde{\mathcal{M}} = (\Sigma_I, \tilde{\Sigma}_S, \tilde{\Sigma}_O, \tilde{\delta}, \tilde{\omega})$ where $\tilde{\delta}$ is defined to duplicate the states e_1 with e_{1_1}, e_{1_2}, and e_3 with new states e_{3_1} and e_{3_2} such that $\delta'(A, e_2) = e_{3_1}$, $\delta'(E, e_2) = e_{3_2}$, $\delta'(A, e_{3_1}) = e_{1_1}$, $\delta'(E, e_{3_1}) = e_{1_2}$, $\delta'(A, e_{3_2}) = e_{1_1}$, $\delta'(E, e_{3_2}) = e_{1_2}$. The new state set is therefore $\tilde{\Sigma}_S = \{e_{1_1}, e_{1_2}, e_2, e_{3_1}, e_{3_2}\}$ and the new output map is specified by: $\omega(e_{1_1}) = \tilde{\xi}_1$, $\omega(e_{1_2}) = \tilde{\xi}_5$, $\omega(e_2) = \tilde{\xi}_2$, $\omega(e_{3_1}) = \tilde{R}_2$, $\omega(e_{3_2}) = \tilde{\xi}_4$. Finally, let $A_{\tilde{\xi}_1} = A_{N_0} = A_0$, $A_{\tilde{\xi}_2} = A_{R_1}$, $A_{\tilde{\xi}_3} = A_{R_2} = A_1$, $A_{\tilde{\xi}_4} = A_{R_2} = A_1$, $A_{\tilde{\xi}_5} = A_{N_0} = A_0$. $A_{\tilde{\xi}_5} = A_{N_0} = A_0$. Setting $\Pi_{A|A}$=0.45, a plot of the spectral radius of $\mathcal{B}_2$ is also shown in Figure 4.3.1 as a function of $\Pi_{E|E}$. As expected, the two plots cross the stability boundary at

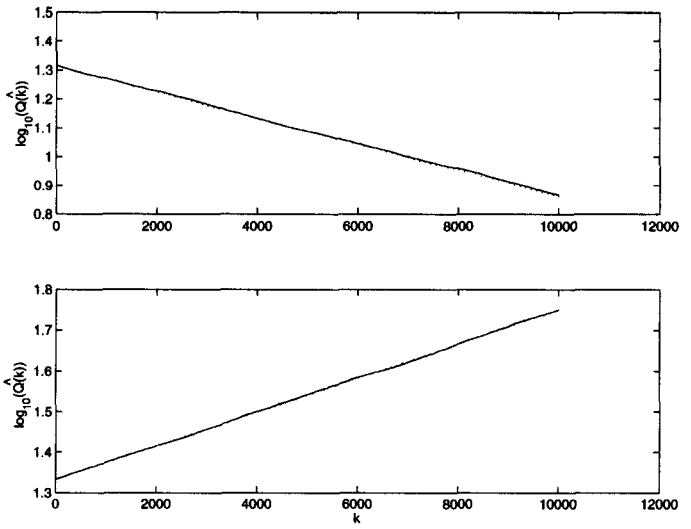

Figure 4.4: The Monte Carlo generated statistic $log_{10}(\hat{Q})$ for the Recovery Example 1 when the input is i.i.d. with $\Pi_E = 0.45$ and $\Pi_E = 0.55$. The dotted line is $k\ log_{10}(\rho(\mathcal{A}_1)) = k\ log_{10}(\rho(\mathcal{B}_1))$.

exactly the same value of $\Pi_{E|E} = 0.35$. In fact the plots coincide exactly for all values of $\Pi_{E,E}$. Two values of $\Pi_{E|E}$ were chosen for Monte Carlo simulation: 0.3 (mean-square stable) and 0.4 (mean-square unstable). As shown in Figure 4.3.1, the simulations agree with the theoretical predictions and the second Lyapunov exponent $\log_{10}(\rho(\mathcal{A}_1)) = \log_{10}(\rho(\mathcal{B}_2))$ serves as an asymptote for the statistics $\log_{10}(\hat{Q})$.

Finally, for the same example, assume $\boldsymbol{\nu}$ is second-order Markov with $\Pi_{A|AA} = 0.7$, $\Pi_{A|AE} = 0.4$ and $\Pi_{A|EA} = 0.45$. The system and simulation parameters are given in Table A-3. Varying $\Pi_{E|EE}$, the spectral radii of $\mathcal{A}_2$ and $\mathcal{B}_3$ are shown in Figure 4.3.1. Two values of $\Pi_{E|EE}$ were chosen for Monte Carlo simulation: 0.7 (mean-square stable) and 0.8 (not mean-square stable). As shown in Figure 4.3.1 shows, the

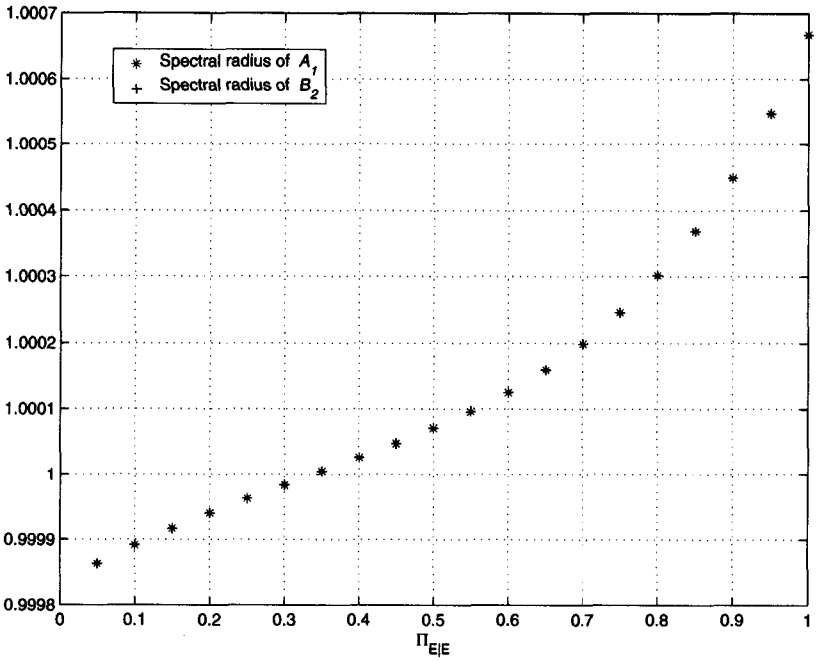

Figure 4.5: A plot of the spectral radius of $\mathcal{A}_1$ and $\mathcal{B}_2$ versus $\Pi_{E|E}$ for the Recovery Example 1 when input $\boldsymbol{\nu}$ is first-order Markov with $\Pi_{A|A} = 0.45$.

simulations agree with the theoretical predictions.

4.3.2 Recovery Example 2

In this section, the same recovery algorithm as in Recovery Example 1 is considered, however, the purpose here is to determine the effect of the recovery duration on mean-square stability. The full state diagram for the simplified recovery algorithm is shown in Figure 4.10. It consists of $1 + M_R$ states: *No*, $R_1, \ldots, R_{M_R}$. When $M_R = 1$, the corresponding transition matrices are:

$$S_A = \begin{bmatrix} e_1 & e_1 \end{bmatrix} \quad S_E = \begin{bmatrix} e_2 & e_1 \end{bmatrix},$$

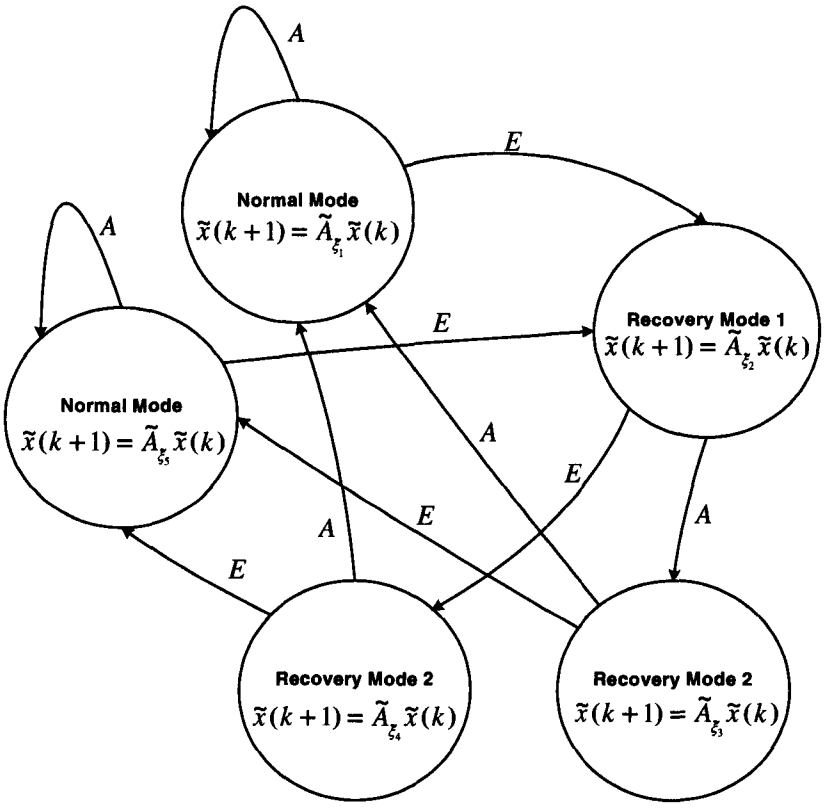

Figure 4.6: A unifilar finite-state machine representation of a recoverable computer control system.

when $M_R = 2$,

$$S_A = \begin{bmatrix} e_1 & e_3 & e_1 \end{bmatrix} \quad S_E = \begin{bmatrix} e_2 & e_3 & e_1 \end{bmatrix},$$

and when $M_R = 5$,

$$S_A = \begin{bmatrix} e_1 & e_3 & e_4 & e_5 & e_6 & e_1 \end{bmatrix} \quad S_E = \begin{bmatrix} e_2 & e_3 & e_4 & e_5 & e_6 & e_1 \end{bmatrix}.$$

The complete set of system and simulation parameters are summarized in Table A-4. Based on the dimensionality discussion of Section 3.9, for the given values of M_R, $dim(\mathcal{A}_r) < dim(\mathcal{B}_{(r+1)})$. Hence, only the approach of using the jump-linear system driven by the cross-chain process $\boldsymbol{\rho}_1$ is considered here. The plot of spectral radius

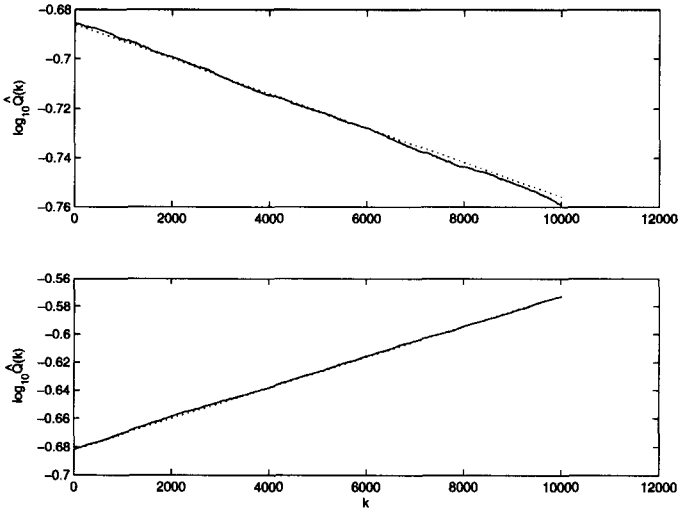

Figure 4.7: The Monte Carlo generated statistic $log_{10}(\hat{Q})$ for the Recovery Example 1 when input is first-order Markovian with $\Pi_{E|E} = 0.3$ and $\Pi_{E|E} = 0.4$. The dotted line is $k\ log_{10}(\rho(\mathcal{A}_1)) = k\ log_{10}(\rho(\mathcal{B}_2))$.

of $\mathcal{A}_1$ as a function of transition probabilities $\Pi_{E|E}$ and stationary probabilities p_E is shown in Figures 4.3.2 and 4.3.2, respectively. Clearly the more persistent the upset or the longer the recovery process, the more likely the closed-loop will not be mean-square stable. The particular case where $\Pi_{E|E} = 0.8$ was simulated by Monte Carlo methods and the resulting statistic $log_{10}(\hat{Q})$ is plotted over time for $M_R = 1$, 2, and 5 as shown in Figure 4.3.2. As per the theoretical predictions, the closed-loop system implementing the recovery algorithm is mean-square stable for $M_R = 1$, but it is not mean-square stable when $M_R = 2$ and $M_R = 5$. The simulations agree with these theoretical predictions.

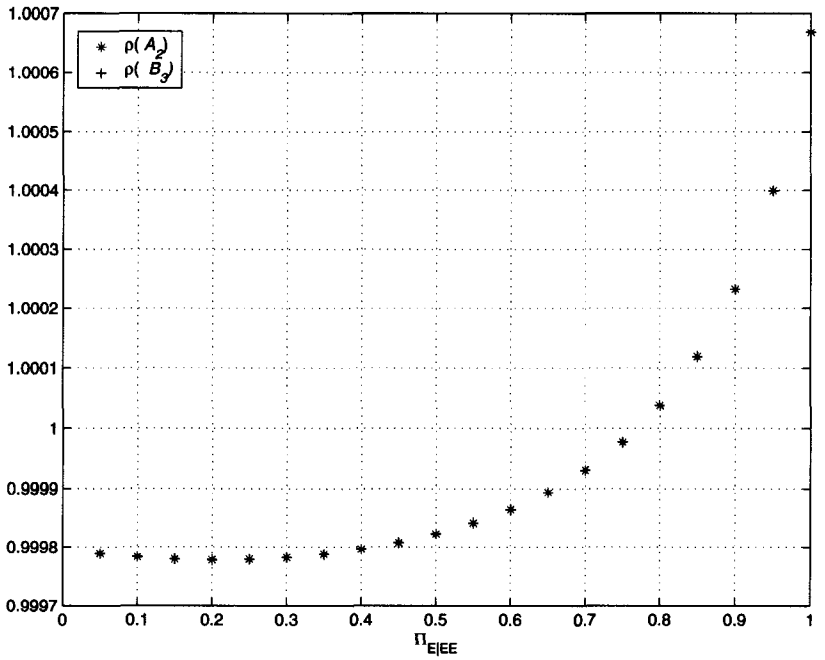

Figure 4.8: The spectral radius of $\mathcal{A}_2$ and $\mathcal{B}_3$ as a function of $\Pi_{E|E,E}$ for the Recovery Example 1.

4.3.3 Recovery Example 3

The third example is a closed-loop system with a controller that is implemented on a RCS with the persistent upset algorithm shown in Figure 1.6. The recovery algorithm has four modes, normal mode, reload mode, release mode and an abort mode. System is always in the normal mode as long as there is no upset, denoted by symbol A. As soon as an upset occurs, denoted by symbol E, the system goes to the reload mode. It stays in the reload mode for a fixed duration N_{Rd} to fetch the good data previously stored. The system then enters the release mode. If an upset occurs during its visit to the release mode, the system revisits the reload mode or else returns to normal mode. However, the system is allowed to revisit the reload

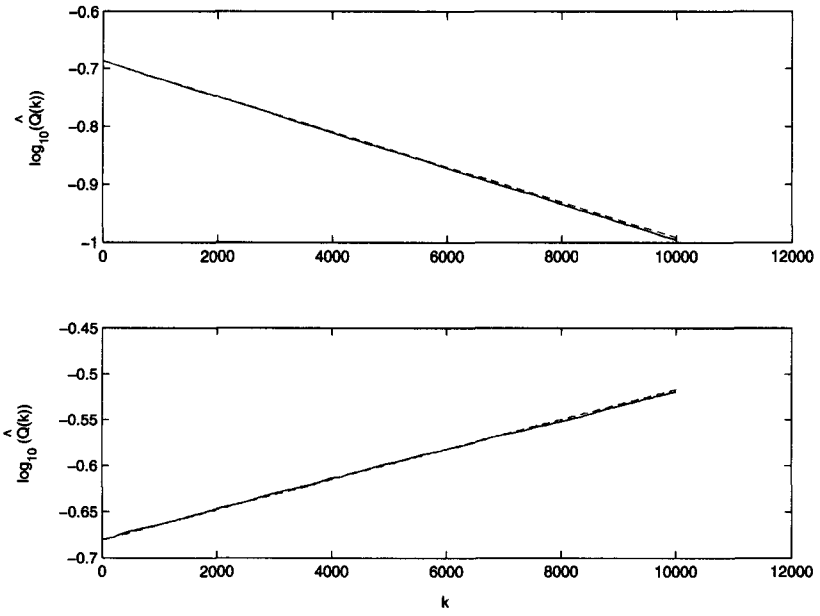

Figure 4.9: The Monte Carlo generated statistic $log_{10}(\hat{Q})$ for the Recovery Example 1 when the input is second-order Markovian with $\Pi_{E|E,E} = 0.7$ and 0.8. The dotted line is $k\ log_{10}(\rho(\mathcal{A}_2)) = k\ log_{10}(\rho(\mathcal{B}_3))$.

mode from the release mode only a fixed number of times N_{Rs} after which it enters an abort mode for a possible emergency action by an external supervisor to bring the system back to its normal operation. The corresponding state diagram is shown in Figure 4.3.3 when $N_{Rd} = N_{Rs} = N_{Ab} = 1$. The example will demonstrate the effect of varying N_{Ab} along with the persistence of upsets when $N_{Ab} = 1$, 2 and 5. Again, based on the dimensionality issues in Section 3.9, for the various values of N_{Ab} considered, $dim(\mathcal{A}_r) < dim(\mathcal{B}_{(r+1)})$. Hence, only the approach of using the jump-linear system driven by the cross-chain process $\boldsymbol{\rho}_1$ is considered here. The state transition matrices

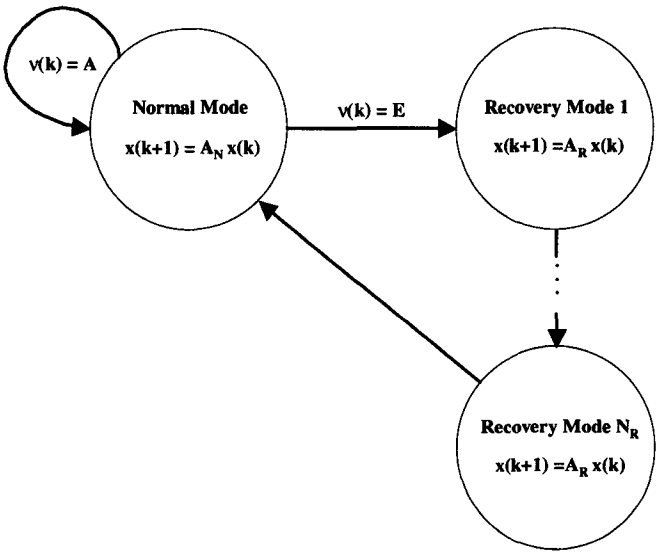

Figure 4.10: The state diagram for the algorithm in Recovery Example 2.

in the case of $N_{Ab} = 1$ are given by:

$$S_A = \begin{bmatrix} e_1 & e_3 & e_1 & e_5 & e_1 & e_1 \end{bmatrix}, S_E = \begin{bmatrix} e_2 & e_3 & e_4 & e_5 & e_6 & e_1 \end{bmatrix}.$$

The state transition matrices for $N_{Ab} = 2$ and $N_{Ab} = 5$, respectively are given by

$$S_A = \begin{bmatrix} e_1 & e_3 & e_1 & e_5 & e_1 & e_7 & e_1 \end{bmatrix}$$
$$S_E = \begin{bmatrix} e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_1 \end{bmatrix},$$

and

$$S_A = \begin{bmatrix} e_1 & e_3 & e_1 & e_5 & e_1 & e_7 & e_8 & e_9 & e_{10} & e_1 \end{bmatrix}$$
$$S_E = \begin{bmatrix} e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_1 \end{bmatrix}.$$

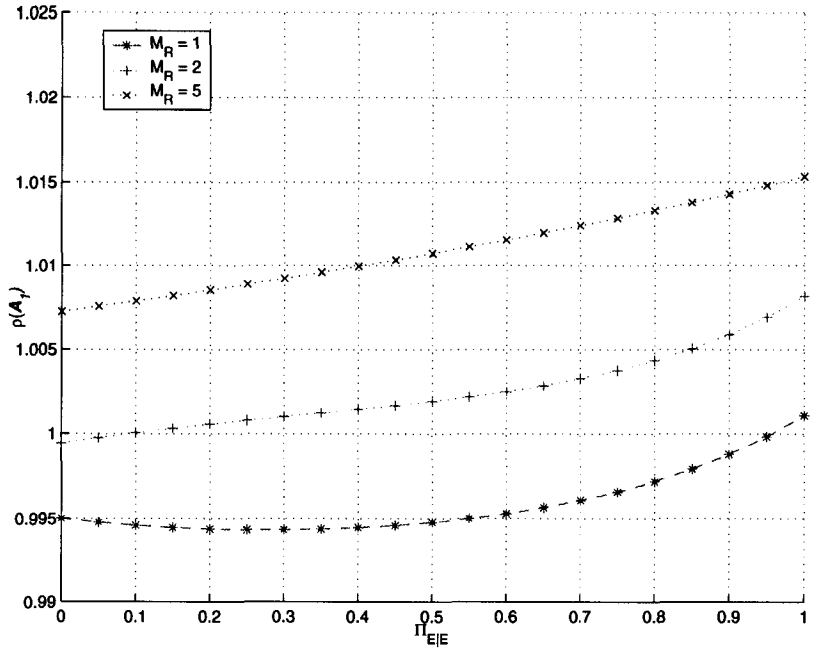

Figure 4.11: The spectral radius of $\mathcal{A}_1$ as a function of $\Pi_{E|E}$ for the Recovery Example 2.

The system and simulation parameters when $N_{Ab} = 1$,2 and 5 are given in Table A-5. The corresponding spectral radii plots of $\mathcal{A}_1$ versus transition probabilities $\Pi_{E|E}$ and stationary probabilities p_E are shown in Figures 4.3.3 and 4.3.3, respectively. The plots in these figures show that the lower the values of N_{Ab}, the more robust is the closed-loop system with respect to stability. The corresponding Monte Carlo simulation when $\Pi_{E|E} = 0.3$ and $N_{Ab} = 1$,2 and 5 are shown in Figure 4.3.3. The simulations verify that the closed-loop system is mean-square stable when $N_{Ab} = 1$, 2, but not mean-square stable when $N_{Ab} = 5$. It is in this manner that these stability criteria can be used as design tools.

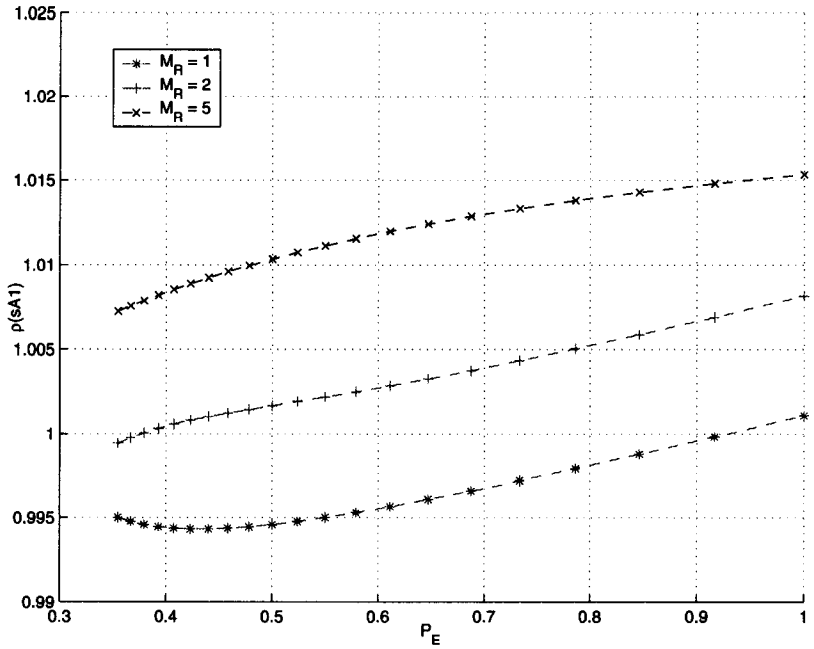

Figure 4.12: The spectral radius of $\mathcal{A}_1$ as a function of p_E for the Recovery Example 2 when $M_R = 1, 2, 5$.

4.4 Summary of Key Points

In this chapter, three recovery models were demonstrated using Monte Carlo simulation and compared against the theoretical predictions made by via the mean-square stability criteria developed in the previous chapter. All the simulations agreed with the theoretical predictions, thus demonstrating the the mean-square stability tools derived. The three examples demonstrated the effect of persistent upsets and recovery duration on the robustness of the stability of closed-loop system. In all three examples plots of Lyapunov exponents matched the plot of simulations of $log(\hat{Q})$ asymptotically as predicted in theory.

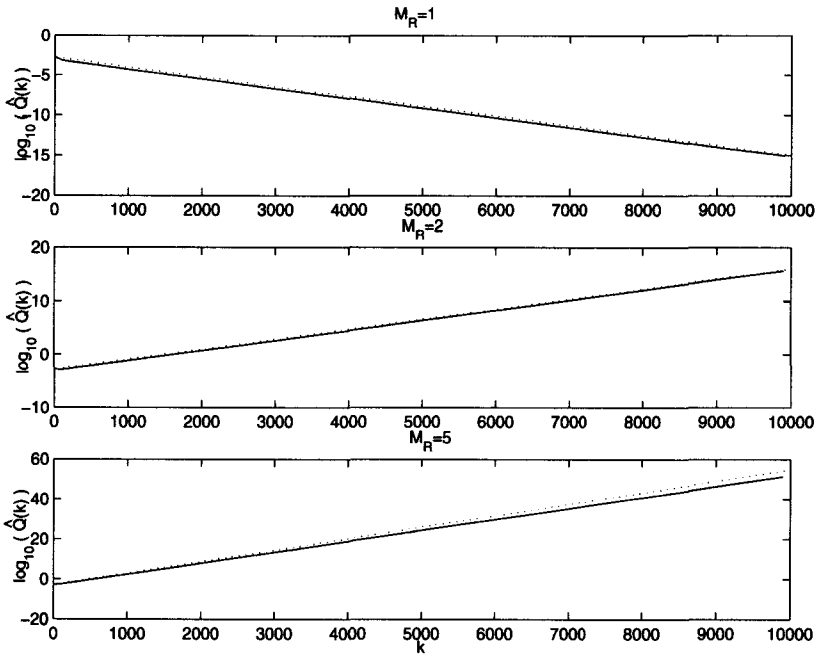

Figure 4.13: The Monte Carlo generated statistic $log_{10}(\hat{Q})$ for the Recovery Example 2 when $\Pi_{E|E} = 0.8$ and M_R is variable. The dotted line is $k\ log_{10}(\rho(\mathcal{A}_1))$.

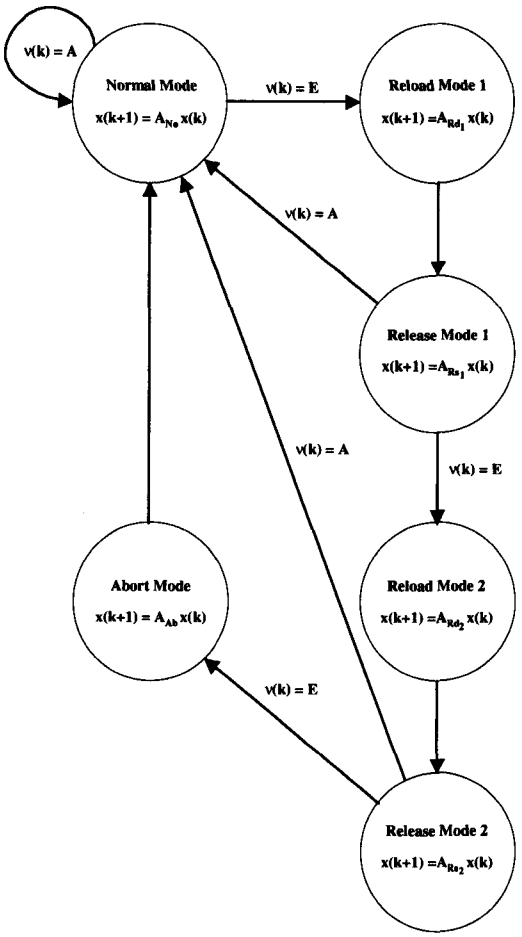

Figure 4.14: The state diagram for the Recovery Example 3 when $N_{Ab} = 1$.

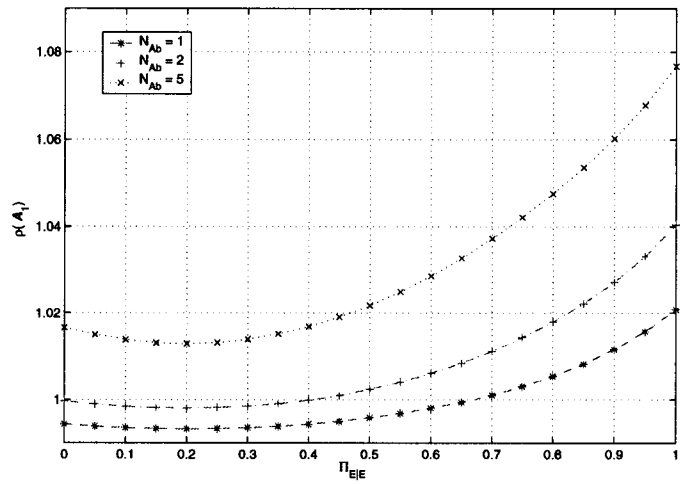

Figure 4.15: The spectral radius of $\mathcal{A}_1$ as a function of $\Pi_{E|E}$ for the Recovery Example 3 when $N_{Ab} = 1, 2, 5$.

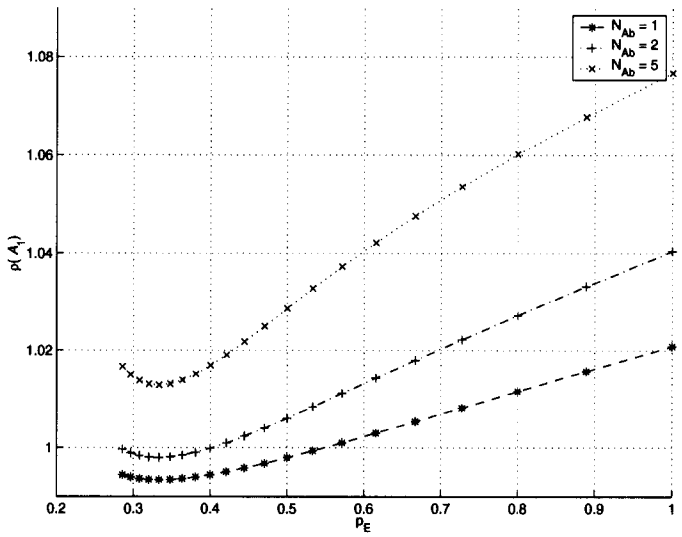

Figure 4.16: The spectral radius of $\mathcal{A}_1$ as a function of p_E for the Recovery Example 3 when $N_{Ab} = 1, 2, 5$.

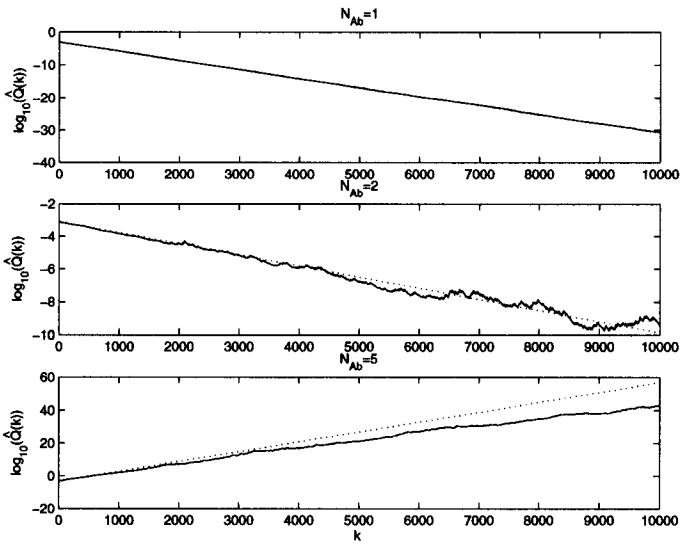

Figure 4.17: The Monte Carlo generated statistic $log_{10}(\hat{Q})$ for the Recovery Example 3, when $\Pi_{E|E} = 0.3$ and N_{Ab} is variable. The dotted line is $k\ log_{10}(\rho(\mathcal{A}_1))$.

CHAPTER V

FUTURE RESEARCH AND CONCLUSIONS

In this chapter, the main contributions of this thesis are summarized and prospects for future research are discussed.

5.1 Summary of Main Contributions

To summarize the main contributions of this research, as outlined in the problem statement of Chapter 1, two random processes generated by a finite-state machine when driven by Markov inputs of order $r \geq 0$ were characterized. It was shown that the random process $(\boldsymbol{\nu}, \boldsymbol{z})$ resulting from the cross-product of input process and state process is a first-order Markov process when the input is an i.i.d. process, and it is a Markov process of order r when input is a Markov process of order r. The state process of the finite-state machine $\boldsymbol{z}$ was shown to be first-order Markovian when input is i.i.d., and the state process of the unifilar type finite-state machine $\tilde{\boldsymbol{z}}$ was shown to be Markovian of order $r+1$ when input is Markovian of any order r. Based on these characterization, mean-square stability conditions were developed for the system $(\boldsymbol{\nu}, \mathcal{M}, A, \boldsymbol{\theta})$ comprising of a jump-linear system driven by an arbitrary finite-state machine with the i.i.d. and higher order Markov processes as inputs. As a corollary, it was also shown that those mean-square stability conditions can also be used to test for exponentially second moment stability and stochastic second moment stability of the system. Mean-square stability was further characterized using the concept of Lyapunov exponent. Finally, three recovery examples were simulated

using Monte Carlo methods that demonstrated the mean-square stability conditions developed.

5.2 Directions for Future Research

In this concluding section, some discussion about the directions for future research is provided. The directions discussed here are based on the advantages and limitations of the methodology presented in this thesis.

5.2.1 Curse of Dimensionality: Some Possible Solutions

When recovery algorithms, especially those with counters for providing a fixed amount of delay in each mode, are modeled as finite-state machines with non-isomorphic mapping, it needs to be investigated whether numerically more efficient testable stability conditions can be derived. Perhaps the answer lies in determining under what conditions the output of such a finite-state machine or some related process is Markovian. Some conditions for Markovianness of such output process are given in [5,7,41], which may be helpful in producing numerically efficient tests. However the conditions are known to be very restrictive.

5.2.2 Jump-Linear Systems Driven by Cascaded Finite-State Machines with Markovian Inputs

A more general situation shown in Figure 5.2.2 a is recovery algorithm modeled as two or more finite-state machines in cascade. Since the cross-chain formed by the

Markov input process ν and the state process z is known to be Markovian of the same order as the input, if there are c number of finite-state machines in cascade, it is expected that the process $(\nu, z_1, \ldots, z_c)$ will also be Markovian of the same order. And hence, it is conjectured that the tools presented here can also be used to infer the mean-square stability property for the jump-linear systems driven by cascaded finite-state machines with Markovian inputs.

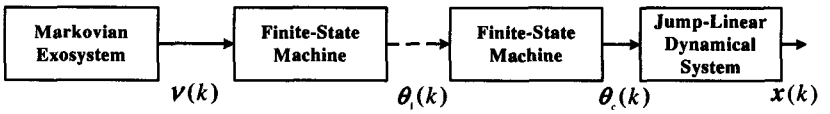

Figure 5.1: A jump-linear system driven by cascaded finite-state machines with a Markovian input.

5.2.3 Systems with Stochastic Recovery Algorithms

Another general situation shown in Figure 5.2.4 is when state transitions in the recovery algorithm are probabilistic. Such algorithms can be modeled by probabilistic finite-state machines (stochastic automata). It is known that the cross-chain process formed by the input and state process is again Markovian of the same order as that of Markovian input. Hence it is conjectured that the stability of hybrid models for recovery algorithms with probabilistic finite-state machines can be analyzed by extending the tools presented in this thesis in an appropriate manner.

Recovery algorithms with counters having random delays with known distributions also need to be investigated. A reasonable hypothesis is: the resulting output process

is semi-Markov. This would lead to a study of jump-linear systems driven by semi-Markov processes.

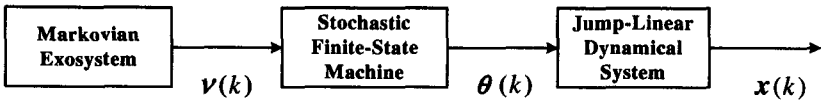

Figure 5.2: A jump-linear system driven by a stochastic finite-state machine with a Markovian input.

5.2.4 Recovery System Identification

A related problem arises as shown in Figure 5.2.4 when the exact nature of the recovery algorithm is unknown. By measuring the output sequence probabilities, it may be possible to identify an approximation for the underlying finite-state machine [4, 12, 47]. Then, using the state estimates, the stability criteria presented here can be applied.

Figure 5.3: A jump-linear system driven by an unknown finite-state machine with a Markovian input.

REFERENCES

[1] B. Adimi-Sadjadi, "Stability of networked control systems in the presence of packet losses," in *Proc. 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, 2003, pp. 676–681.

[2] F. B. Bastani, "Relational programs: an architecture for robust real-time safety-critical process-control systems," *Annals of Software Engineering*, vol. 7, no. 1, pp. 5–24, 1999.

[3] K. Benjelloun, E. K. Boukas, and P. Shi, "Robust stochastic stability of discrete-time linear systems with Markovian jumping parameters," in *Proc. 36th IEEE Conference on Decision and Control*, San Diego, California, 1997, pp. 559–564.

[4] D. Blackwell and L. Koopmans, "On the identifiability problem for functions of Markov chains," *Annals of Mathematical Statistics*, vol. 28, no. 4, pp. 1011–1015, 1957.

[5] T. L. Booth, *Sequential Machines and Automata Theory.* John Wiley & Sons, Inc., New York, 1967.

[6] M. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Trans. Automatic Control*, vol. 43, pp. 475–482, 1998.

[7] C. J. Burke and M. Rosenblatt, "A Markovian function of a Markov chain," *The Annals of Mathematical Statistics*, vol. 29, no. 4, pp. 1112–1122, 1958.

[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, no. 1, pp. 154–178, 1993.

[9] A. M. Davis, "Markov chains as random input automata," *American Mathematical Monthly*, vol. 68, no. 3, pp. 264–267, 1961.

[10] Y. Fang, "Stability analysis of linear control systems with uncertain parameters," Ph.D. dissertation, Case Western University, Cleveland, Ohio, 1994.

[11] Y. Fang and K. A. Loparo, "Stochastic stability of jump linear systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 7, pp. 1204–1208, 2002.

[12] E. J. Gilbert, "On the identifiability problem for functions of finite Markov chains," *Annals of Mathematical Statistics*, vol. 30, no. 3, pp. 688–697, 1959.

[13] O. R. González, W. S. Gray, A. Tejada, and S. Patilkulkarni, "Stability analysis of electromagnetic interference upset recovery methods," in *Proc. 40th IEEE Conference on Decision and Control*, Orlando, Florida, 2001, pp. 4134–4139.

[14] O. R. González, A. Tejada, and W. S. Gray, "Analytical tools for the design and verification of safety critical control systems," in *Proc. 2001 International Conference on Lightning and Static Electricity*, Seattle, Washington, 2001, pp. 2938–1–10.

[15] O. R. González, A. Tejada, and W. S. Gray, "Analysis of design trade-offs in rollback recovery methods for fault tolerant digital control systems," in *Proc. 2002 American Control Conference*, Anchorage, Alaska, 2002, pp. 4801–4806.

[16] W. S. Gray, S. Patilkulkarni, and O. R. González, "Stochastic stability of a recoverable computer control system modeled as a finite-state machine," in *Proc. 2003 American Control Conference*, Denver, Colorado, 2003, pp. 2240–2245.

[17] W. S. Gray, H. Zhang, and O. R. González, "Closed-loop performance measures for flight controllers subject to neutron-induced upsets," in *Proc. 42nd Conference on Decision and Control*, Maui, Hawaii, 2003, pp. 2465–2470.

[18] K. Hagbae and K. G. Shin, "Modeling of externally-induced/common-cause faults in fault-tolerant systems," in *Proc. 13th Digital Avionics Systems Conference*, Phoenix, Arizona, 1994, pp. 402–407.

[19] A. Hassibi, S. P. Boyd, and J. How, "Control of asynchronous dynamical systems with rate constraints on events," in *Proc. 38th IEEE Conference on Decision and Control*, Orlando, Florida, 1999, pp. 1345–1351.

[20] J. Hespanha, "Hybrid stochastic systems: application to communication networks," in *Lecture Notes in Computer Science*, R. Alur and G. Pappas, Eds. Springer-Verlag, New York, 2004, vol. 2993, no. 12, pp. 387–401.

[21] R. Hess, "Computing platform architectures for robust operation in the presence of lightning and other electromagnetic threats," in *Proc. 16th Digital Avionics Systems Conference*, vol. 1, New York, 1997, pp. 4.3–9–16.

[22] R. Hess, "Options for aircraft function preservation in the presence of lightning," in *Proc. International Conference on Lightning and Static Electricity*, Toulouse, France, Paper No. 106, 1999.

[23] W. M. L. Holkombe, *Algebraic Automata Theory.* Cambridge University Press, New York, 1982.

[24] J. Hu, J. Lygeros, and S. Sastry, "Towards hybrid stochastic systems," in *Lecture Notes in Computer Science*, Pittsburgh, Pennsylvania, 2000, vol. 1790, pp. 160–173.

[25] W. Huang and E. McCluskey, "Transient errors and rollback recovery in LZ compression," in *International Symposium on Dependable Computing*, Los Angels, California, 2000, pp. 128–135.

[26] W. H. Jermann, "Measurement of linearly dependent processes," Ph.D. dissertation, University of Connecticut, Connecticut, 1967.

[27] 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.

[28] R. E. Kalman, P. L. Falb, and M. A. Arbib, *Mathematical System Theory.* McGraw-Hill Inc., New York, 1969.

[29] S. Karlin and H. M. Taylor, *Second Course in Stochastic Processes.* Springer-Verlag, New York, 1981.

[30] G. Kemeny and L. J. Snell, *Finite Markov Chains.* Springer-Verlag, New York, 1976.

[31] Z. Kohavi, *Switching And Finite Automata Theory.* McGraw-Hill Inc., New York, 1978.

[32] J. Lala and R. Harper, "Architectural principles for safety-critical real-time applications," vol. 82, no. 1, pp. 25–40, 1994.

[33] Q. Ling and M. D. Lemmon, "Optimal dropout compensation in networked control systems," in *Proc. 42nd Conference on Decision and Control*, Maui, Hawaii, 2003, pp. 670–675.

[34] A. M. Lyapunov, *Stability of Motion: General Problem.* Taylor and Francis, Co., London, UK, 1992.

[35] M. Malekpour and W. Torres, "Characterization of a recoverable flight control computer system," in *Proc. 1999 IEEE International Conference on Control Applications*, Kohala Coast, Hawaii, 1999, pp. 1519–1524.

[36] M. Malekpour and W. Torres, "Characterization of a flight control computer with rollback recovery," in *Proc. 19th Digital Avionics Systems Conference*, Philadelphia, Pennsylvania, 2000, pp. 3.C.4–1–8.

[37] D. Marculescu and R. Marculescu, "Information-theoretic bounds for switching activity analysis in finite-state machines under temporally correlated inputs," in *Proc. 33rd IEEE Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, California, 1999.

[38] D. Marculescu, R. Marculescu, and M. Pedram, "Steady-state probability estimation in finite-state machines considering higher-order temporal effects," University of Southern California, Tech. Rep. CENG97-19, 1997.

[39] M. Mariton, *Jump-Linear Systems in Automatic Control.* Marcel-Decker, New York, 1990.

[40] P. S. Miner, V. A. Carreno, M. Malekpour, and W. Torres, "A case-study application of rtca do-254: Design assurance guidance for airborne electronic hardware," in *Proc. 19th Digital Avionics Systems Conference*, vol. 1, Philedelphia, Pennsylvania, 2000, pp. 1.A.1–1–8.

[41] D. Moorehead, "The analysis and synthesis of probability transformers," Electronics Research Laboratory, University of California, Berkely, California, Tech. Rep. 65-06, 1965.

[42] T. Morozan, "Optimal stationary control for dynamic systems with perturbations," *Stochastic Analysis and Applications*, vol. 1, no. 2, pp. 299–325, 1983.

[43] R. Narasimhan, D. J. Rosenkrantz, and S. S. Ravi, "Early comparison and decision strategies for datapaths that recover from transient faults," *IEEE Transactions on Circuits & Systems I-Fundamental Theory & Applications*, vol. 44, no. 5, pp. 435–438, 1997.

[44] S. Patilkulkarni, H. Herencia-Zapana, W. S. Gray, and O. R. González, "On the stability of jump-linear systems driven by finite-state machines with Markovian

inputs," in *Proc. 2004 American Control Conference*, Boston, Massachusettes, 2004, pp. 2534–2539.

[45] D. K. Pradhan, *Fault-Tolerant Computer System Design.* Prentice Hall, Inc., New-Jersey, 1995.

[46] A. Ranganathan and S. Upadhyaya, "Performance evaluation of rollback-recovery techniques in computer programs," *IEEE Transactions on Reliability*, vol. 42, no. 2, pp. 220–226, 1993.

[47] S. Rudich, "Inferring the structure of a markov chain from its output," in *Proc. 26th Annual Symposium on Foundations of Computer Science*, Portland, Oregon, 1985, pp. 321–325.

[48] A. Schaft and H. Schumacher, "An introduction to hybrid dynamical systems," in *Lecture Notes in Control and Informational Sciences.* Springer-Verlag, Heidelberg, New York, 2000, vol. 251.

[49] P. Seiler and R. Sengupta, "Analysis of communication losses in vehicle control problems," in *Proc. 2001 American Control Conference*, Arlington, Virginia, 2001, pp. 1491–1496.

[50] E. Seneta, *Non-Negative Matrices and Markov Chains.* Springer Series in Statistics, New York, 1981.

[51] E. D. Sontag, "Interconnected automata and linear systems: A theoretical framework in discrete-time," in *Hybrid Systems: 3, Verification and Control.* Springer-Verlag, New York, 1995, pp. 436–448.

[52] A. Tejada, "Analysis of error recovery effects on digital flight control systems," Master's thesis, Old Dominion University, Norfolk, Virginia, 2002.

[53] A. Tejada, O. R. González, and W. S. Gray, "Towards analysis of jump linear systems with state-dependent and stochastic switching," in *Proc. 2004 American Control Conference*, Boston, MA, 2004, pp. 1893–1898.

[54] G. N. Tsertsvadze, "Certain properties of stochastic automata and certain methods for synthesizing them," *Automation and Remote Control*, vol. 24, no. 3, pp. 341–352, 1963.

[55] A. S. Willsky and B. C. Levy, "Stochastic stability research for complex power systems," Messachusetts Institute of Technology, Boston, Massachusetts, Tech. Rep. ET-76-C-01-2295, 1979.

[56] L. Xiao, A. Hassibi, and J. How, "Control with random communication delays via a discrete-time jump system approach," in *Proc. 2000 American Control Conference*, vol. 3, Chicago, Illinois, 2000, pp. 28–30.

[57] H. Ye, A. Michel, and L. Hou, "Stability theory for hybrid dynamical systems," *IEEE Trans. Automatic Control*, vol. 43, pp. 461–474, 1998.

[58] W. Zhang, M. S. Branicky, and S. Phillips, "Stability of networked control systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, pp. 84–99, 2001.

APPENDIX A: TABLES OF SYSTEM AND SIMULATION PARAMETERS

Table A-1: The system and simulation parameters for the Recovery Example 1 when the input is i.i.d.

Parameter	Value
Σ_I	$\{A, E\}$
Σ_S	$\{e_1, e_2, \ldots, e_N\}$
Σ_O	$\{N_0, R_1, R_2\}$
$\{A_{N_0}, A_{R_1}, A_{R_2}\}$	$\{0.999, 1.001001, 1.001001\}$
Π_A	$\{0.45, 0.55\}$
Π_E	$\{0.55, 0.45\}$
$\boldsymbol{\nu}(0)$	$P(\boldsymbol{\nu}(0) = A) = P(\boldsymbol{\nu}(0) = E)$
$\boldsymbol{z}(0)$	e_1
$\boldsymbol{\theta}(0)$	N_0
$\boldsymbol{x}(0)$	uniform on $[-0.05, 0.05]$
Monte Carlo runs	1000
number of samples	10001

Table A-2: The system and simulation parameters for the Recovery Example 1 when the input is first-order Markov.

Parameter	**Value**
Σ_I	$\{A, E\}$
Σ_S	$\{e_1, e_2, \ldots, e_N\}$
Σ_O	$\{N_0, R_1, R_2\}$
$\{A_{N_0}, A_{R_1}, A_{R_2}\}$	$\{0.999, 1.001001, 1.001001\}$
Π_{AA}	0.45
Π_{EE}	$\{0.3, 0.4\}$
$\boldsymbol{\nu}(0)$	$P(\boldsymbol{\nu}(0) = A) = P(\boldsymbol{\nu}(0) = E)$
$\boldsymbol{z}(0)$	e_1
$\boldsymbol{\theta}(0)$	No
$\boldsymbol{x}(0)$	uniform on $[-0.05, 0.05]$
Monte Carlo runs	1000
number of samples	10001

Table A-3: The system and simulation parameters for the Recovery Example 1 when the input is second-order Markov.

Parameter	Value
Σ_I	$\{A, E\}$
Σ_S	$\{e_1, e_2, \ldots, e_N\}$
Σ_O	$\{N_0, R_1, R_2\}$
$\{A_{N_0}, A_{R_1}, A_{R_2}\}$	$\{0.999, 1.001001, 1.001001\}$
$\pi_{A\|AA}$	0.7
$\pi_{A\|AE}$	0.4
$\pi_{A\|EA}$	0.45
$\pi_{E\|EE}$	$\{0.7, 0.8\}$
$\boldsymbol{\nu}(0)$	$P(\boldsymbol{\nu}(0) = A) = P(\boldsymbol{\nu}(0) = E)$
$\boldsymbol{z}(0)$	e_1
$\boldsymbol{\theta}(0)$	N_0
$\boldsymbol{x}(0)$	uniform on $[-0.05, 0.05]$
Monte Carlo runs	1000
number of samples	10001

Table A-4: The system and simulation parameters for the Recovery Example 2.

Parameter	Value
Σ_I	$\{A, E\}$
Σ_S	$\{e_1, e_2, \ldots, e_N\}$
Σ_O	$\{N_0, R_1, \ldots, R_{M_R}\}$
M_R	variable: $1, 2, 5$
N	$M_R + 1$
$\{A_N, A_{R_i}\}$	$\{0.99, 1.0112\}$
Π_{AA}	0.45
Π_{EE}	0.8
$\boldsymbol{\nu}(0)$	$P(\boldsymbol{\nu}(0) = E) = P(\boldsymbol{\nu}(0) = A)$
$\boldsymbol{z}(0)$	e_1
$\boldsymbol{\theta}(0)$	N_0
$\boldsymbol{x}(0)$	uniform on $[-0.05, 0.05]$
Monte Carlo runs	1000
number of samples	10001

Table A-5: The system and simulation parameters for the Recovery Example 3.

Parameter	Value
Σ_I	$\{A, E\}$
Σ_S	$\{e_1, e_2, \ldots, e_N\}$
Σ_O	$\{No, Rd_1, Rs_1, Rd_2, Rs_2,$ $Ab_1 \ldots, Ab_{N_{Ab}}\}$
$\{A_{N_0}, A_{Rd_i}, A_{Rs_i}, A_{Ab_i}\}$	$\{0.99, 1.039, 0.96, 1.08\}$
N_{Rd}, N_{Rs}	$1, 2$
N_{Ab}	variable: $1, 2, 5$
N	$N_{Ab} + 5$
Π_{AA}	0.6
Π_{EE}	0.3
$\boldsymbol{\nu}(0)$	$P(\boldsymbol{\nu}(0) = E) = P(\boldsymbol{\nu}(0) = A)$
$\boldsymbol{z}(0)$	e_1
$\boldsymbol{\theta}(0)$	N_0
$\boldsymbol{x}(0)$	uniform on $[-0.05, 0.05]$
Monte Carlo runs	1000
number of samples	10001

APPENDIX B: SIMULATION SOFTWARE

This appendix is a compilation of the the MATLAB® source code used for generating Markov chains, the theoretical prediction of stability boundary, as well and the simulation results for the three recovery examples in Chapter 4.

B-1 Generating Markov Chains

B-1.1 Independent Identically Distributed Process

Gen_disc_iid: This program generates independent identically distributed process for the given distribution Π_E and for the given number of samples Numsam.

```
%Here \Pi_E=pe

function N=Gen_disc_iid(pe,Numsam)

N(1,Numsam+1)=0;

rand('state',sum(100*clock));

for jj=1:Numsam+1

if unifrnd(0,1)<(1-pe)
```

```
    N(jj)=0;
else
    N(jj)=1;
end;


end;
```

B-1.2 First-Order Markov Process

Gen_disc_Markov_first: This function generates a first-order two state Markov process with two states (A,E) for the given transition probability matrix characterized by transition probabilities $\Pi_{A|A}$, $\Pi_{E|E}$ and for the given number of samples Numsam. The Markov chain is initialized at steady state proabilities.

```
%Here transition probabilities \Pi_{A|A}=paa, \Pi_{E|E}=pee, stationary probabi
function N=Gen_disc_markov_first(paa,pee,Numsam)


N(1,Numsam+1)=0;


pa=((1-pee)/((1-paa)+(1-pee)));
pe=((1-paa)/((1-paa)+(1-pee)));


rand('state',sum(100*clock));
```

```
if unifrnd(0,1)<pa
    N(1)=0;
else
    N(1)=1;
end;

for jj=2:1:Numsam+1

   if N(jj-1)==0
       if unifrnd(0,1)<paa
        N(jj)=0;
        else N(jj)=1;
        end;
    else
        if unifrnd(0,1)<pee
            N(jj)=1;
        else N(jj)=0;
        end;
    end;
end;
```

B-1.3 Second-Order Markov Process

Gen_disc_Markov_second: This function generates a second-order two state Markov process of states (A,E) for the given transition probability matrix characterized by transition probabilities $\Pi_{A|AA}$, $\Pi_{A|AE}$, $\Pi_{A|EA}$ and $\Pi_{A|EE}$ and for the given number of samples Numsam. The Markov chain is initialized at steady-state probabilities.

```
%Here $\Pi_{A|AA}=paaa$, $\Pi_{A|AE}=paae$, $\Pi_{A|EA}=paea$ and $\Pi_{A|EE}=p
function N=Gen_disc_markov_second(paaa,paae,paea,paee,Numsam)

N(1,Numsam+1)=0;

Pi_I=[paaa paae 0 0;
    0 0 paea paee;
    1-paaa 1-paae 0 0;
    0   0    1-paea 1-paee;];

p=null(Pi_I-eye(4));
p=p/sum(p);

rand('state',sum(100*clock));

Uinit=unifrnd(0,1);
```

```
if Uinit<p(1)
     N(2)=0; N(1)=0;
elseif Uinit <p(1)+p(2)
    N(2)=0; N(1)=1;
elseif Uinit < p(1)+p(2)+p(3)
    N(2)=1 ; N(1)=0;
else
    N(2)=1; N(1)=1;
end;



for jj=3:1:Numsam+1


if N(jj-1)==0 & N(jj-2)==0
       if unifrnd(0,1)<paaa
            N(jj)=0;
        else N(jj)=1;
        end;
    elseif N(jj-1)==0 & N(jj-2)==1
        if unifrnd(0,1)<paae
            N(jj)=0;
        else N(jj)=1;
```

```
        end;


    elseif N(jj-1)==1 & N(jj-2)==0
        if unifrnd(0,1)<paea
            N(jj)=0;
        else N(jj)=1;
        end;
    else
        if unifrnd(0,1)<paee
            N(jj)=0;
        else N(jj)=1;
        end;


    end;
end;
```

B-2 Recovery Example 1

B-2.1 Program Code for Theoretical Prediction

plot_srscriptA0: This program produces plots for the Recovery Example 1 when $M_R = 2$, Π_E v/s spectral radius of $\mathcal{A}_0$ for the i.i.d. input.

```
%System parameter in the normal mode

A_0=0.999;


%System parameter in the recovery mode

A_1=1/A_0;


%Transition matrices for the finite-state machine


S_1=[1 0 1;

    0 0 0;

    0 1 0];


S_2=[0 0 1;

    1 0 0;

    0 1 0];


%Varying paramaeter \Pi_{E|E}


pi_r_1=0.05:0.05:1;


sz=size(pi_r_11,2);

for jj=1:sz;
```

```
Pi_I=[1-pi_r_1(jj) pi_r_1(jj); 1-pi_r_1(jj) pi_r_1(jj)];

Pi_IO=kron(Pi_I, eye(3))*blkdiag(S_1,S_2);

sA0=kron(Pi_IO,eye(1))*blkdiag(A_0^2,A_1^2,A_1^2,A_0^2,A_1^2,A_1^2);

sr_scriptA1(jj)=max(abs(eig(sA0)));

end;

%Plot \Pi_{E|E) v/s spectralradius(scriptA1);

plot(pi_r_11,sr_scriptA1, 'r*');
title(strcat(' Recovery Example 1', ' A_0= ', num2str(A_0),
' A_1= ', num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));
```

plot_srscriptB1: This program plots variation in Π_E v/s spectral radius of $\mathcal{B}_1$, when input is i.i.d..

```
%Fix \Pi_{A|A}=0.45

%System parameter in the normal mode
A_0=0.999;
```

```
%System parameter in the recovery mode

A_1=1/A_0;


%Transition matrices for the finite-state machine


S_1=[1 0 1;

    0 0 0;

    0 1 0];


S_2=[0 0 1;

    1 0 0;

    0 1 0];


%Varying paramaeter \Pi_{E|E}


pi_r_1=0.05:0.05:1;

pi_r_0=1-pi_r_11;


sz=size(pi_r_1,2);

for jj=1:sz;


Pi_I=[pi_r_0  pi_r_1(jj); pi_r_0 pi_r_1(jj)];
```

```
Pi_0=(1-pi_r_1(jj))*S_1+pi_r_11(jj)*S_2;
```

plot_srscriptA1: This program produces plots for the Recovery Example 1, $\Pi_{E|E}$ v/s spectral radius of $\mathcal{A}_1$, corresponding to jump-linear system driven by cross-chain process $\rho = (\nu, \theta)$, when input is a first-order Markov process.

```
%Fix \Pi_{A|A A}=0.45
pi_r_00=0.45;

%System parameter in the normal mode
A_0=0.999;

%System parameter in the recovery mode
A_1=1/A_0;

%State transition matrices
S_1=[1 0 1; 0 0 0; 0 1 0]; S_2=[0 0 1; 1 0 0; 0 1 0];

%Vary probability \Pi{A|E E}, \Pi{E|E E}
pi_r_11=0.05:0.05:1;
```

```
sz=size(pi_r_11,2);


for jj=1:sz;


    Pi_I=[pi_r_00 1-pi_r_11(jj); 1-pi_r_00 pi_r_11(jj)];


    Pi_IO=kron(Pi_I, eye(3))*blkdiag(S_1,S_2);
    sA1=(kron(Pi_IO, eye(1)))*blkdiag(A_0^2, A_1^2, A_1^2, A_0^2, A_1^2, A_1^2)


    sr_scriptA1(jj)=max(abs(eig(sA1)));


end;
%Plot pi_{E|E) v/s spectralradius(scriptA1);
plot(pi_r_11,sr_scriptA1, 'r*');
title(strcat('Simplified Recovery Example', ' A_0= ',
num2str(A_0), ' A_1= ', num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));
```

plot_srscriptB2: This program produces plots for the Recovery Example 1 when the input is a first-order Markovian, Π_{EE} v/s spectral radius of $\mathcal{B}_2$ for fixed $\Pi_{A,A} = 0.45$, when input is first-order Markov process.

```
%Fix \Pi_{A|A}=0.45
```

```
pi_r_00=0.45;

A_0=0.999; A_1=1/A_0;

%Vary  \Pi_{E|E}=0.45
pi_r_11=0.05:0.05:1;

sz=size(pi_r_11,2);

for jj=1:sz;

    Pi_I=[pi_r_00 1-pi_r_11(jj);
    1-pi_r_00 pi_r_11(jj)];

    Pi_02_11=zeros(5,5);
    Pi_02_12=zeros(5,5);
    Pi_02_13=zeros(5,5);
    Pi_02_14=zeros(5,5);
    Pi_02_15=zeros(5,5);
    Pi_02_21=zeros(5,5);
    Pi_02_22=zeros(5,5);
    Pi_02_23=zeros(5,5);
```

```
Pi_02_24=zeros(5,5);

Pi_02_25=zeros(5,5);

Pi_02_31=zeros(5,5);

Pi_02_32=zeros(5,5);

Pi_02_33=zeros(5,5);

Pi_02_34=zeros(5,5);

Pi_02_35=zeros(5,5);

Pi_02_41=zeros(5,5);

Pi_02_42=zeros(5,5);

Pi_02_43=zeros(5,5);

Pi_02_44=zeros(5,5);

Pi_02_45=zeros(5,5);

Pi_02_51=zeros(5,5);

Pi_02_52=zeros(5,5);

Pi_02_53=zeros(5,5);

Pi_02_54=zeros(5,5);

Pi_02_55=zeros(5,5);


%Pi_02_11(1,:)=[P(x1|x1,x1) P(x1|x1,x2) P(x1|x1,x3)

    %P(x1|x1,x4) P(x1|x1,x5)] ;


Pi_02_11(1,:)=[pi_r_00 0 pi_r_00 pi_r_00 0] ;
```

```
%Pi_02_21(2,:)=[P(x2|x1,x1) P(x2|x1,x2) P(x2|x1,x3)
    %P(x2|x1,x4) P(x2|x1,x5)] ;

Pi_02_21(1,:)=[1-pi_r_00 0 1-pi_r_00 1-pi_r_00 0] ;

%Pi_02_31(3,:)=[P(x3|x1,x1) P(x3|x1,x2) P(x3|x1,x3)
        %P(x3|x1,x4) P(x3|x1,x5)] ;

Pi_02_31(1,:)=[0 0 0 0 0] ;

%Pi_02_41(1,:)=[P(x4|x1,x1) P(x4|x1,x2) P(x4|x1,x3)
            %P(x4|x1,x4) P(x4|x1,x5)] ;

Pi_02_41(1,:)=[0 0 0 0 0 ] ;

%Pi_02_51(1,:)=[P(x5|x1,x1) P(x5|x1,x2) P(x5|x1,x3)
        %P(x5|x1,x4) P(x5|x1,x5)] ;

Pi_02_51(1,:)=[0 0 0 0 0 ] ;

%Pi_02_12(2,:)=[P(x1|x2,x1) P(x1|x2,x2) P(x1|x2,x3)
        %P(x1|x2,x4) P(x1|x2,x5)] ;
```

```
Pi_02_12(2,:)= [ 0 0 0 0 0 ];

%Pi_02_22(2,:)=[P(x2|x2,x1) P(x2|x2,x2) P(x2|x2,x3)
    %P(x2|x2,x4) P(x2|x2,x5)];

Pi_02_22(2,:)=[0 0 0 0 0];

%Pi_02_32(2,:)=[P(x3|x2,x1) P(x3|x2,x2) P(x3|x2,x3)
%P(x3|x2,x4) P(x3|x2,x5)];

Pi_02_32(2,:)=[ 1-pi_r_11(jj) 0 0 0 1-pi_r_11(jj)];

%Pi_02_42(2,:)=[P(x4|x2,x1) P(x4|x2,x2) P(x4|x2,x3)
%P(x4|x2,x4) P(x4|x2,x5)];

Pi_02_42(2,:)=[pi_r_11(jj) 0 0 0 pi_r_11(jj)];

%Pi_02_52(2,:)=[P(x5|x2,x1) P(x5|x2,x2) P(x5|x2,x3)
%P(x5|x2,x4) P(x5|x2,x5)];

Pi_02_52(2,:)=[0 0 0 0 0];

%Pi_02_13(3,:)=[P(x1|x3,x1) P(x1|x3,x2) P(x1|x3,x3)
```

```
%P(x1|x3,x4) P(x1|x3,x5)];


Pi_02_13(3,:)=[0 pi_r_00  0 0 0];


%Pi_02_23(3,:)=[P(x2|x3,x1) P(x2|x3,x2) P(x2|x3,x3)
%P(x2|x3,x4) P(x2|x3,x5)];


Pi_02_23(3,:)=[0 0 0 0 0];


%Pi_02_33(3,:)=[P(x3|x3,x1) P(x3|x3,x2) P(x3|x3,x3)
%P(x3|x3,x4) P(x3|x3,x5)];


Pi_02_33(3,:)=[0 0 0 0 0];


%Pi_02_43(3,:)=[P(x4|x3,x1) P(x4|x3,x2) P(x4|x3,x3)
%P(x4|x3,x4) P(x4|x3,x5)];


Pi_02_43(3,:)=[0 0 0 0 0];


%Pi_02_53(3,:)=[P(x5|x3,x1) P(x5|x3,x2) P(x5|x3,x3)
%P(x5|x3,x4) P(x5|x3,x5)];


Pi_02_53(3,:)=[0 1-pi_r_00 0 0 0];
```

```
%Pi_02_14(4,:)=[P(x1|x4,x1) P(x1|x4,x2) P(x1|x4,x3)
%P(x1|x4,x4) P(x1|x4,x5)];

Pi_02_14(4,:)=[0 1-pi_r_11(jj) 0 0 0];

%Pi_02_24(4,:)=[P(x2|x4,x1) P(x2|x4,x2) P(x2|x4,x3)
%P(x2|x4,x4) P(x2|x4,x5)];

Pi_02_24(4,:)=[0 0 0 0 0];

%Pi_02_34(4,:)=[P(x3|x4,x1) P(x3|x4,x2) P(x3|x4,x3)
%P(x3|x4,x4) P(x3|x4,x5)];

Pi_02_34(4,:)=[0 0 0 0 0];

%Pi_02_44(4,:)=[P(x4|x4,x1) P(x4|x4,x2) P(x4|x4,x3)
%P(x4|x4,x4) P(x4|x4,x5)];

Pi_02_44(4,:)=[0 0 0 0 0];

%Pi_02_54(4,:)=[P(x5|x4,x1) P(x5|x4,x2) P(x5|x4,x3)
%P(x5|x4,x4) P(x5|x4,x5)];
```

```
Pi_02_54(4,:)=[0 pi_r_11(jj) 0 0 0];

%Pi_02_15(5,:)=[P(x1|x5,x1) P(x1|x5,x2) P(x1|x5,x3)
%P(x1|x5,x4) P(x1|x5,x5)];

Pi_02_15(5,:)=[0 0 0 0 0];

%Pi_02_25(5,:)=[P(x2|x5,x1) P(x2|x5,x2) P(x2|x5,x3)
%P(x2|x5,x4) P(x2|x5,x5)];

Pi_02_25(5,:)=[0 0 pi_r_11(jj) pi_r_11(jj) 1-pi_r_00];

%Pi_02_35(5,:)=[P(x3|x5,x1) P(x3|x5,x2) P(x3|x5,x3)
%P(x3|x5,x4) P(x3|x5,x5)];

Pi_02_35(5,:)=[0 0 0 0 0];

%Pi_02_45(5,:)=[P(x4|x5,x1) P(x4|x5,x2) P(x4|x5,x3)
%P(x4|x5,x4) P(x4|x5,x5)];

Pi_02_45(5,:)=[0 0 0 0 0];
```

```
%Pi_02_55(,:)=[P(x5|x5,x1) P(x5|x5,x2) P(x5|x5,x3)
%P(x5|x5,x4) P(x5|x5,x5)];

Pi_02_55(5,:)=[0 0 1-pi_r_11(jj) 1-pi_r_11(jj) pi_r_00];

Pi_02=[Pi_02_11 Pi_02_12 Pi_02_13 Pi_02_14 Pi_02_15;
Pi_02_21 Pi_02_22 Pi_02_23 Pi_02_24 Pi_02_25;
Pi_02_31 Pi_02_32 Pi_02_33 Pi_02_34 Pi_02_35;
Pi_02_41 Pi_02_42 Pi_02_43 Pi_02_44 Pi_02_45;
Pi_02_51 Pi_02_52 Pi_02_53 Pi_02_54 Pi_02_55];

sB2=(kron(Pi_02, eye(1)))*blkdiag(eye(5)*A_0^2, eye(5)*A_1^2,
    eye(5)*A_1^2, eye(5)*A_1^2, eye(5)*A_0^2);

veceig=abs(eig(sB2));
veceig(find(abs(eig(sB2))>2e-8))
sr_scriptB2(jj)=max(abs(eig(sB2)));

end;
```

```
%Plot pi_{eta_2,eta_2) v/s spectralradius(scriptB2);

plot(pi_r_11,sr_scriptB2, '+');
```

plot_srscriptA2: This program takes second-order transition probabilities as input, in the form of its first-order representation (as a stochastic matrix) and computes spectral radius of $\mathcal{A}_2$ for the Recovery Example 1.

```
paaa=0.7;

paae=0.4;

%%Fix p_{\eta2|\eta2,\eta1}=0.45

a=0.45;

%Vary probability \Pi{A|E E}, \Pi{E|E E}
b=0.05:0.05:1;
```

```
sz=size(b,2);



%System parameter in the normal mode

A_0=0.999;


%System parameter in the recovery mode

A_1=1/A_0;


%Transition matrices for the finite-state machine

S_1=[1 0 1; 0 0 0; 0 1 0];



S_2=[0 0 1; 1 0 0; 0 1 0];



for jj=1:sz;


%Transition probability matrix for the second-order Markov process

Pi_I=[paaa paae 0   0;

      0    0  a  1-b(jj);

      1-paaa 1-paae  0    0;

      0  0     1-a  b(jj);]
```

```
    Pi_I0=kron(Pi_I, eye(3))*blkdiag(S_1,S_2,S_1,S_2);
    sA2=(kron(Pi_I0, eye(1)))*blkdiag(A_0^2, A_1^2, A_1^2, A_0^2,
    A_1^2, A_1^2,A_0^2, A_1^2, A_1^2,A_0^2, A_1^2, A_1^2);

    sr_scriptA2(jj)=max(abs(eig(sA2)));

end;

plot(b,sr_scriptA2, 'r*');
title(strcat('Recovery Example', ' A_0= ',
 num2str(A_0), ' A_1= ', num2str(A_1), '\Pi_{aa}= ',num2str(a)));
```

plot_srscriptB3: This program produces plots for the Recovery Example 1, when the input is a second-order Markov process, for $\Pi_{E|EE}$ v/s spectral radius of $\mathcal{B}_3$, given $pi_{A|AA} = 0.7$,$pi_{A|AE} = 0.4$ and $pi_{A|EA} = 0.45$.

```
pi_000=0.7;
pi_001=0.4;
pi_010=0.45;
pi_100=0.3;
```

```
pi_101=0.6;

pi_110=0.55;


A_0=0.999; A_1=1/A_0;


b=0.05:0.05:1;

sz=size(b,2);


for jj=1:sz



    pi_011=1-b(jj);

    pi_111=b(jj);


    Pi_I=[pi_000 pi_001 0 0;

        0   0 pi_010 pi_011;

        pi_100 pi_101 0 0;

        0   0 pi_110 pi_111];


    Pi_03_111=zeros(5,5);

    Pi_03_112=zeros(5,5);

    Pi_03_113=zeros(5,5);

    Pi_03_114=zeros(5,5);
```

```
Pi_03_115=zeros(5,5);



Pi_03_121=zeros(5,5);

Pi_03_122=zeros(5,5);

Pi_03_123=zeros(5,5);

Pi_03_124=zeros(5,5);

Pi_03_125=zeros(5,5);



Pi_03_131=zeros(5,5);

Pi_03_132=zeros(5,5);

Pi_03_133=zeros(5,5);

Pi_03_134=zeros(5,5);

Pi_03_135=zeros(5,5);



Pi_03_141=zeros(5,5);

Pi_03_142=zeros(5,5);

Pi_03_143=zeros(5,5);

Pi_03_144=zeros(5,5);

Pi_03_145=zeros(5,5);
```

```
Pi_03_151=zeros(5,5);

Pi_03_152=zeros(5,5);

Pi_03_153=zeros(5,5);

Pi_03_154=zeros(5,5);

Pi_03_155=zeros(5,5);


Pi_03_211=zeros(5,5);

Pi_03_212=zeros(5,5);

Pi_03_213=zeros(5,5);

Pi_03_214=zeros(5,5);

Pi_03_215=zeros(5,5);



Pi_03_221=zeros(5,5);

Pi_03_222=zeros(5,5);

Pi_03_223=zeros(5,5);

Pi_03_224=zeros(5,5);

Pi_03_225=zeros(5,5);



Pi_03_231=zeros(5,5);

Pi_03_232=zeros(5,5);
```

```
Pi_03_233=zeros(5,5);

Pi_03_234=zeros(5,5);

Pi_03_235=zeros(5,5);



Pi_03_241=zeros(5,5);

Pi_03_242=zeros(5,5);

Pi_03_243=zeros(5,5);

Pi_03_244=zeros(5,5);

Pi_03_245=zeros(5,5);



Pi_03_251=zeros(5,5);

Pi_03_252=zeros(5,5);

Pi_03_253=zeros(5,5);

Pi_03_254=zeros(5,5);

Pi_03_255=zeros(5,5);

Pi_03_311=zeros(5,5);

Pi_03_312=zeros(5,5);

Pi_03_313=zeros(5,5);

Pi_03_314=zeros(5,5);

Pi_03_315=zeros(5,5);
```

```
Pi_03_321=zeros(5,5);

Pi_03_322=zeros(5,5);

Pi_03_323=zeros(5,5);

Pi_03_324=zeros(5,5);

Pi_03_325=zeros(5,5);



Pi_03_331=zeros(5,5);

Pi_03_332=zeros(5,5);

Pi_03_333=zeros(5,5);

Pi_03_334=zeros(5,5);

Pi_03_335=zeros(5,5);



Pi_03_341=zeros(5,5);

Pi_03_342=zeros(5,5);

Pi_03_343=zeros(5,5);

Pi_03_344=zeros(5,5);

Pi_03_345=zeros(5,5);
```

```
Pi_03_351=zeros(5,5);

Pi_03_352=zeros(5,5);

Pi_03_353=zeros(5,5);

Pi_03_354=zeros(5,5);

Pi_03_355=zeros(5,5);


Pi_03_411=zeros(5,5);

Pi_03_412=zeros(5,5);

Pi_03_413=zeros(5,5);

Pi_03_414=zeros(5,5);

Pi_03_415=zeros(5,5);



Pi_03_421=zeros(5,5);

Pi_03_422=zeros(5,5);

Pi_03_423=zeros(5,5);

Pi_03_424=zeros(5,5);

Pi_03_425=zeros(5,5);



Pi_03_431=zeros(5,5);

Pi_03_432=zeros(5,5);

Pi_03_433=zeros(5,5);
```

```
Pi_03_434=zeros(5,5);

Pi_03_435=zeros(5,5);



Pi_03_441=zeros(5,5);

Pi_03_442=zeros(5,5);

Pi_03_443=zeros(5,5);

Pi_03_444=zeros(5,5);

Pi_03_445=zeros(5,5);



Pi_03_451=zeros(5,5);

Pi_03_452=zeros(5,5);

Pi_03_453=zeros(5,5);

Pi_03_454=zeros(5,5);

Pi_03_455=zeros(5,5);

Pi_03_511=zeros(5,5);

Pi_03_512=zeros(5,5);

Pi_03_513=zeros(5,5);

Pi_03_514=zeros(5,5);

Pi_03_515=zeros(5,5);
```

```
Pi_03_521=zeros(5,5);

Pi_03_522=zeros(5,5);

Pi_03_523=zeros(5,5);

Pi_03_524=zeros(5,5);

Pi_03_525=zeros(5,5);



Pi_03_531=zeros(5,5);

Pi_03_532=zeros(5,5);

Pi_03_533=zeros(5,5);

Pi_03_534=zeros(5,5);

Pi_03_535=zeros(5,5);



Pi_03_541=zeros(5,5);

Pi_03_542=zeros(5,5);

Pi_03_543=zeros(5,5);

Pi_03_544=zeros(5,5);

Pi_03_545=zeros(5,5);



Pi_03_551=zeros(5,5);
```

```
    Pi_03_552=zeros(5,5);

    Pi_03_553=zeros(5,5);

    Pi_03_554=zeros(5,5);

    Pi_03_555=zeros(5,5);


% Pi_03_111(1,:)=[P(x1|x1,x1,x1) P(x1|x1,x1,x2) P(x1|x1,x1,x3)
% P(x1|x1,x1,x4) P(x1|x1,x1,x5)] ;


Pi_03_111(1,:)=[pi_000 0 pi_000 pi_000 0] ;


%Pi_03_112(2,:)=[P(x1|x1,x2,x1) P(x1|x1,x2,x2) P(x1|x1,x2,x3)
%P(x1|x1,x2,x4) P(x1|x1,x2,x5)] ;


Pi_03_112(2,:)=[0 0 0 0 0] ;


%Pi_03_111(1,11:15)=[P(x1|x1,x3,x1) P(x1|x1,x3,x2) P(x1|x1,x3,x3)
%P(x1|x1,x3,x4) P(x1|x1,x3,x5)] ;


Pi_03_113(3,:)=[0 pi_000 0 0 0] ;


%Pi_03_111(1,16:20)=[P(x1|x1,x4,x1) P(x1|x1,x4,x2) P(x1|x1,x4,x3)
%P(x1|x1,x4,x4) P(x1|x1,x4,x5)] ;
```

```
    Pi_03_114(4,:)=[0 pi_001 0 0 0 ] ;

%Pi_03_111(1,21:25)=[P(x1|x1,x5,x1) P(x1|x1,x5,x2) P(x1|x1,x5,x3)
%P(x1|x1,x5,x4) P(x1|x1,x5,x5)] ;

    Pi_03_115(5,:)=[0 0 0 0 0] ;

%Pi_03_121(2,26:30)=[P(x1|x2,x1,x1) P(x1|x2,x1,x2) P(x1|x2,x1,x3)
%P(x1|x2,x1,x4) P(x1|x2,x1,x5)] ;

     Pi_03_121(1,:)=[0 0 0 0 0] ;

%Pi_03_121(2,:)=[P(x1|x2,x2,x1) P(x1|x2,x2,x2) P(x1|x2,x2,x3)
%P(x1|x2,x2,x4) P(x1|x2,x2,x5)] ;
Pi_03_122(2,:)=[0 0 0 0 0] ;

%Pi_03_121(3,:)=[P(x1|x2,x3,x1) P(x1|x2,x3,x2) P(x1|x2,x3,x3)
%P(x1|x2,x3,x4) P(x1|x2,x3,x5)] ;
Pi_03_123(3,:)=[0 0 0 0 0] ;

%Pi_03_121(4,:)=[P(x1|x2,x4,x1) P(x1|x2,x4,x2) P(x1|x2,x4,x3)
%P(x1|x2,x4,x4) P(x1|x2,x4,x5)] ;
Pi_03_124(4,:)=[0 0 0 0 0] ;
```

```
%Pi_03_121(5,:)=[P(x1|x2,x5,x1) P(x1|x2,x5,x2) P(x1|x2,x5,x3)
%P(x1|x2,x5,x4) P(x1|x2,x5,x5)] ;
 Pi_03_125(5,:)=[0 0 0 0 0] ;

%Pi_03_131(1,:)=[P(x1|x3,x1,x1) P(x1|x3,x1,x2) P(x1|x3,x1,x3)
%P(x1|x3,x1,x4) P(x1|x3,x1,x5)] ;
    Pi_03_131(1,:)=[0 0 0 0 0] ;

%  Pi_03_132(2,:)=[P(x1|x3,x2,x1) P(x1|x3,x2,x2) P(x1|x3,x2,x3)
%P(x1|x3,x2,x4) P(x1|x3,x2,x5)] ;
   Pi_03_132(2,:)=[pi_001 0 0 0 pi_001] ;

%  Pi_03_133(3,:)=[P(x1|x3,x3,x1) P(x1|x3,x3,x2) P(x1|x3,x3,x3)
%P(x1|x3,x3,x4) P(x1|x3,x3,x5)] ;
   Pi_03_133(3,:)=[0 0 0 0 0] ;

%  Pi_03_134(4,:)=[P(x1|x3,x4,x1) P(x3|x3,x4,x2) P(x3|x3,x4,x3)
%P(x3|x3,x4,x4) P(x3|x3,x4,x5)] ;
   Pi_03_134(4,:)=[0 0 0 0 0] ;

%  Pi_03_135(5,:)=[P(x1|x3,x5,x1) P(x3|x3,x5,x2) P(x3|x3,x5,x3)
%P(x3|x3,x5,x4) P(x3|x3,x5,x5)] ;
```

```
   Pi_03_135(5,:)=[0 0 0 0 0] ;

%  Pi_03_141(1,:)=[P(x1|x4,x1,x1) P(x1|x4,x1,x2) P(x1|x4,x1,x3)
%P(x1|x4,x1,x4) P(x1|x4,x1,x5)] ;
   Pi_03_141(1,:)=[0 0 0 0 0] ;

%  Pi_03_142(2,:)=[P(x1|x4,x2,x1) P(x1|x4,x2,x2) P(x1|x4,x2,x3)
%P(x1|x4,x2,x4) P(x1|x4,x2,x5)] ;
   Pi_03_142(2,:)=[pi_011 0 0 0 pi_011] ;

%  Pi_03_143(3,:)=[P(x1|x4,x3,x1) P(x1|x4,x3,x2) P(x1|x4,x3,x3)
%P(x1|x4,x3,x4) P(x1|x4,x1,x5)] ;
   Pi_03_143(3,:)=[0 0 0 0 0] ;

%  Pi_03_144(4,:)=[P(x1|x4,x4,x1) P(x1|x4,x4,x2) P(x1|x4,x4,x3)
%P(x1|x4,x4,x4) P(x1|x4,x4,x5)] ;
   Pi_03_144(4,:)=[0 0 0 0 0] ;

%  Pi_03_145(5,:)=[P(x1|x4,x5,x1) P(x1|x4,x5,x2) P(x1|x4,x5,x3)
%P(x1|x4,x5,x4) P(x1|x4,x5,x5)] ;
   Pi_03_145(5,:)=[0 0 0 0 0] ;

%  Pi_03_151(1,:)=[P(x1|x5,x1,x1) P(x1|x5,x1,x2) P(x1|x5,x1,x3)
```

```
%P(x1|x5,x1,x4) P(x1|x5,x1,x5)] ;
   Pi_03_151(1,:)=[0 0 0 0 0] ;

%  Pi_03_152(2,:)=[P(x1|x5,x2,x1) P(x1|x5,x2,x2) P(x1|x5,x2,x3)
%P(x1|x5,x2,x4) P(x1|x5,x2,x5)] ;
   Pi_03_152(2,:)=[0 0 0 0 0] ;

%  Pi_03_153(3,:)=[P(x1|x5,x3,x1) P(x1|x5,x3,x2) P(x1|x5,x3,x3)
%P(x1|x5,x3,x4) P(x1|x5,x3,x5)] ;
   Pi_03_153(3,:)=[0 0 0 0 0] ;

%  Pi_03_154(4,:)=[P(x1|x5,x4,x1) P(x1|x5,x4,x2) P(x1|x5,x4,x3)
%P(x1|x5,x4,x4) P(x1|x5,x4,x5)] ;
   Pi_03_154(4,:)=[0 0 0 0 0] ;

%  Pi_03_155(5,:)=[P(x1|x5,x5,x1) P(x1|x5,x5,x2) P(x1|x5,x5,x3)
%P(x1|x5,x5,x4) P(x1|x5,x5,x5)] ;
   Pi_03_155(5,:)=[0 0 0 0 0] ;

%  Pi_03_211(1,:)= [P(x2|x1,x1,x1) P(x2|x1,x1,x2) P(x2|x1,x1,x3)
%P(x2|x1,x1,x4) P(x2|x1,x1,x5)] ;
   Pi_03_211(1,:)=[pi_100 0 pi_100 pi_100 0] ;
```

```
%  Pi_03_212(2,:)= [P(x2|x1,x2,x1) P(x2|x1,x2,x2) P(x2|x1,x2,x3)
%P(x2|x1,x2,x4) P(x2|x1,x2,x5)] ;
   Pi_03_212(2,:)=[0 0 0 0 0] ;

%  Pi_03_213(3,:)= [P(x2|x1,x3,x1) P(x2|x1,x3,x2) P(x2|x1,x3,x3)
%P(x2|x1,x3,x4) P(x2|x1,x3,x5)] ;
   Pi_03_213(3,:)=[0 pi_100 0 0 0] ;

%  Pi_03_214(4,:)= [P(x2|x1,x4,x1) P(x2|x1,x4,x2) P(x2|x1,x4,x3)
%P(x2|x1,x4,x4) P(x2|x1,x4,x5)] ;
   Pi_03_214(4,:)=[0 pi_101 0 0 0] ;

%  Pi_03_215(5,:)= [P(x2|x1,x5,x1) P(x2|x1,x5,x2) P(x2|x1,x5,x3)
%P(x2|x1,x5,x4) P(x2|x1,x5,x5)] ;
   Pi_03_215(5,:)=[0 0 0 0 0] ;

%  Pi_03_253(3,:)= [P(x2|x5,x3,x1) P(x2|x5,x3,x2) P(x2|x5,x3,x3)
%P(x2|x5,x3,x4) P(x2|x5,x3,x5)] ;

   Pi_03_253(3,:)=[0 pi_110 0 0 0];

%  Pi_03_254(4,:)= [P(x2|x5,x4,x1) P(x2|x5,x4,x2) P(x2|x5,x4,x3)
%P(x2|x5,x4,x4) P(x2|x5,x4,x5)] ;
```

```
   Pi_03_254(4,:)=[0 pi_111 0 0 0];

%  Pi_03_253(3,:)= [P(x2|x5,x5,x1) P(x2|x5,x5,x2) P(x2|x5,x5,x3)
%P(x2|x5,x5,x4) P(x2|x5,x5,x5)] ;
   Pi_03_255(5,:)=[0 0 pi_101 pi_101 pi_100];

%  Pi_03_321(1,:)= [P(x3|x2,x1,x1) P(x3|x2,x1,x2) P(x3|x2,x1,x3)
%P(x3|x2,x1,x4) P(x3|x2,x1,x5)] ;
   Pi_03_321(1,:)=[pi_010 0 pi_010 pi_010 0];

%  Pi_03_325(1,:)= [P(x3|x2,x5,x1) P(x3|x2,x5,x2) P(x3|x2,x5,x3)
%P(x3|x2,x5,x4) P(x3|x2,x5,x5)] ;
   Pi_03_325(5,:)=[0 0 pi_011 pi_011 pi_010];

%  Pi_03_421(1,:)= [P(x4|x2,x1,x1) P(x4|x2,x1,x2) P(x4|x2,x1,x3)
%P(x4|x2,x1,x4) P(x4|x2,x1,x5)] ;
   Pi_03_421(1,:)=[pi_110 0 pi_110 pi_110 0];

%  Pi_03_425(1,:)= [P(x4|x2,x1,x1) P(x4|x2,x1,x2) P(x4|x2,x1,x3)
%P(x4|x2,x1,x4) P(x4|x2,x1,x5)] ;
   Pi_03_425(5,:)=[0 0 pi_111 pi_111 pi_110];
```

```
%  Pi_03_532(1,:)= [P(x5|x3,x2,x1) P(x5|x3,x2,x2) P(x35|x3,x2,x3)
%P(x5|x3,x2,x4) P(x5|x3,x2,x5)] ;
   Pi_03_532(2,:)=[pi_101 0 0 0 pi_101];

%  Pi_03_542(1,:)= [P(x5|x4,x2,x1) P(x5|x4,x2,x2) P(x5|x4,x2,x3)
%P(x5|x4,x2,x4) P(x5|x4,x2,x5)] ;
   Pi_03_542(2,:)=[pi_111 0 0 0 pi_111];

%  Pi_03_553(3,:)= [P(x5|x5,x3,x1) P(x5|x5,x3,x2) P(x5|x5,x3,x3)
%P(x5|x5,x3,x4) P(x5|x5,x3,x5)] ;
   Pi_03_553(3,:)=[0 pi_010 0 0 0];

%  Pi_03_554(1,:)= [P(x5|x5,x4,x1) P(x5|x5,x4,x2) P(x5|x5,x4,x3)
%P(x5|x5,x4,x4) P(x5|x5,x4,x5)] ;
   Pi_03_554(4,:)=[0 pi_011 0 0 0];

%  Pi_03_555(1,:)= [P(x5|x5,x5,x1) P(x5|x5,x5,x2) P(x5|x5,x5,x3)
%P(x5|x5,x5,x4) P(x5|x5,x5,x5)] ;
   Pi_03_555(5,:)=[0 0 pi_001 pi_001 pi_000];


   Pi_03_11=[Pi_03_111,Pi_03_112,Pi_03_113,Pi_03_114,Pi_03_115];
   Pi_03_12=[Pi_03_121,Pi_03_122,Pi_03_123,Pi_03_124,Pi_03_125];
```

```
Pi_03_13=[Pi_03_131,Pi_03_132,Pi_03_133,Pi_03_134,Pi_03_135];

Pi_03_14=[Pi_03_141,Pi_03_142,Pi_03_143,Pi_03_144,Pi_03_145];

Pi_03_15=[Pi_03_151,Pi_03_152,Pi_03_153,Pi_03_154,Pi_03_155];



Pi_03_21=[Pi_03_211,Pi_03_212,Pi_03_213,Pi_03_214,Pi_03_215];

Pi_03_22=[Pi_03_221,Pi_03_222,Pi_03_223,Pi_03_224,Pi_03_225];

Pi_03_23=[Pi_03_231,Pi_03_232,Pi_03_233,Pi_03_234,Pi_03_235];

Pi_03_24=[Pi_03_241,Pi_03_242,Pi_03_243,Pi_03_244,Pi_03_245];

Pi_03_25=[Pi_03_251,Pi_03_252,Pi_03_253,Pi_03_254,Pi_03_255];

Pi_03_31=[Pi_03_311,Pi_03_312,Pi_03_313,Pi_03_314,Pi_03_315];

Pi_03_32=[Pi_03_321,Pi_03_322,Pi_03_323,Pi_03_324,Pi_03_325];

Pi_03_33=[Pi_03_331,Pi_03_332,Pi_03_333,Pi_03_334,Pi_03_335];

Pi_03_34=[Pi_03_341,Pi_03_342,Pi_03_343,Pi_03_344,Pi_03_345];

Pi_03_35=[Pi_03_351,Pi_03_352,Pi_03_353,Pi_03_354,Pi_03_355];

Pi_03_41=[Pi_03_411,Pi_03_412,Pi_03_413,Pi_03_414,Pi_03_415];

Pi_03_42=[Pi_03_421,Pi_03_422,Pi_03_423,Pi_03_424,Pi_03_425];

Pi_03_43=[Pi_03_431,Pi_03_432,Pi_03_433,Pi_03_434,Pi_03_435];

Pi_03_44=[Pi_03_441,Pi_03_442,Pi_03_443,Pi_03_444,Pi_03_445];

Pi_03_45=[Pi_03_451,Pi_03_452,Pi_03_453,Pi_03_454,Pi_03_455];
```

```
Pi_03_51=[Pi_03_511,Pi_03_512,Pi_03_513,Pi_03_514,Pi_03_515];

Pi_03_52=[Pi_03_521,Pi_03_522,Pi_03_523,Pi_03_524,Pi_03_525];

Pi_03_53=[Pi_03_531,Pi_03_532,Pi_03_533,Pi_03_534,Pi_03_535];

Pi_03_54=[Pi_03_541,Pi_03_542,Pi_03_543,Pi_03_544,Pi_03_545];

Pi_03_55=[Pi_03_551,Pi_03_552,Pi_03_553,Pi_03_554,Pi_03_555];



Pi_03=[blkdiag(Pi_03_11, Pi_03_12, Pi_03_13, Pi_03_14, Pi_03_15);
    blkdiag(Pi_03_21, Pi_03_22, Pi_03_23, Pi_03_24, Pi_03_25);
    blkdiag(Pi_03_31, Pi_03_32, Pi_03_33, Pi_03_34, Pi_03_35);
    blkdiag(Pi_03_41, Pi_03_42, Pi_03_43, Pi_03_44, Pi_03_45);
    blkdiag(Pi_03_51, Pi_03_52, Pi_03_53, Pi_03_54, Pi_03_55)];


    m(jj)=max(max(Pi_03))
    M(jj)= max(max(abs(eig(Pi_03))));
    nzs(jj)= size(find(Pi_03),1);


 sB3=(kron(Pi_03, eye(1)))*blkdiag(eye(25)*A_0^2, eye(25)*A_1^2,
 eye(25)*A_1^2, eye(25)*A_1^2, eye(25)*A_0^2);


 sr_scriptB3(jj)=max(abs(eig(sB3,'nobalance')));


end;
```

```
plot(b,sr_scriptB3, 'k+');
hold on
```

B-2.2 Program Code for Monte Carlo Simulation

sim_recovery_ex1: This code simulates Recovery Example 1, when the input is an i.i.d. process, a first-order Markov process or a second-order Markov process. Number of Monte Carlo runs M=1000, Number of samples=10001.

```
sim_recovery_ex1(outputfname)

M=1000;

A_0=0.999; A_1=1/A_0;

%%Number of Samples

Numsam=10001;

S_1=[1 0 1; 0 0 0; 0 1 0];
```

```
S_2=[0 0 1; 1 0 0; 0 1 0];


%i.i.d.

order=0;

Pi_I=[0.55 0.55; 0.45 0.45];


runsim_ex1(order,Pi_I);


%first-order Markov

order=1;

Pi_I=[0.45 0.7; 0.55 0.3];


runsim_ex1(order,Pi_I);


paaa=0.7;

paae=0.4;


paea=0.45;

paee=0.2;


%second-order Markov

%order=2;
```

```
Pi_I=[paaa paae 0 0;
    0 0 paea paee;
    1-paaa 1-paae 0 0;
    0   0  1-paea 1-paee];

runsim_ex1(order,Pi_I);

function runsim_ex1(order,Pi_I)
  %% Time counters
tick1 = 0;

tick2 = 0;

q1=zeros(1,Numsam+1);
Q1=zeros(1,Numsam+1);
x=zeros(1,Numsam+1);

x0=0.3;

A_0=0.999; A_1=1/A_0;

for i=1:M
```

```
p_A1(i)=0;

p_A2(i)=0;

p_A3(i)=0;

p_E1(i)=0;

p_E2(i)=0;

p_E3(i)=0;


p_A(i)=0;

p_E(i)=0;



tic;


rand('state',sum(100*clock));


% Generate state vector initial conditions

x(1)=unifrnd(1*norm(x0),2*norm(x0),size(x0,1),size(x0,2));

% Accumulate first-order moment

q1(1) = q1(1) + norm(x(1));


Q1(1) = Q1(1) + norm(x(1)'*x(1));
```

```
theta=zeros(1,Numsam+1);

%Generate i.i.d. process for given distribution $\Pi$

If order==0


N=Gen_disc_iid(Pi_I(2,2),Numsam);


elseif order==1


N=Gen_disc_Markov_first(Pi_I(1,1),Pi_I(2,2),Numsam);


else


N=Gen_disc_Markov_second(Pi_I(1,1),Pi_I(1,2),Pi_I(2,3),Pi_I(2,4),Numsam);


end;


U=unifrnd(0,3);




if U<1
    theta(1)=1;
    A=A_0;
elseif 1<U<2
```

```
    theta(1)=2;
    A=A_1;
elseif 2<U<3
        theta(1)=3;
        A=A_1;

end;



rand('state',sum(100*clock));
    for k=1:Numsam
        %Simulate closed-loop dynamics
         x(k+1)=A*x(k);

        if N(k)==0
            p_A(i)=p_A(i)+1;
        else
            p_E(i)=p_E(i)+1;
        end;

        if N(k)==0 & theta(k)==1
            theta(k+1)=1;
```

```
        A=A_0;

        p_A1(i)=p_A1(i)+1;

    elseif N(k)==0 & theta(k)==2
        theta(k+1)=3;
        A=A_1;

        p_A2(i)=p_A2(i)+1;

    elseif N(k)==0 & theta(k)==3
        theta(k+1)=1;
        A=A_0;

        p_A3(i)=p_A3(i)+1;
    end;



if N(k)==1 & theta(k)==1
        theta(k+1)=2;
        A=A_1;
```

```
            p_E1(i)=p_E1(i)+1;


        elseif N(k)==1 & theta(k)==2
            theta(k+1)=3;
            A=A_1;


            p_E2(i)=p_E2(i)+1;
         elseif N(k)==1 & theta(k)==3
            theta(k+1)=1;
            A=A_0;


            p_E3(i)=p_E3(i)+1;


         end;
    %% Accumulate first moment
    q1(k+1)= q1(k+1) + norm(x(k+1));
    %% Accumulate second moment
    Q1(k+1)= Q1(k+1) + norm(x(k+1)'*x(k+1));



end;


%% Mean First and Second Moments
```

```
q=q1/M;

Q=Q1/M;


tick1=toc;

tick2=tick2+tick1;

fprintf('toc = %8.2f total time = %8.2f\n',tick1, tick2);


end;



p_A1m=mean(p_A1)/Numsam;

p_A2m=mean(p_A2)/Numsam;

p_A3m=mean(p_A3)/Numsam;

p_E1m=mean(p_E1)/Numsam;

p_E2m=mean(p_E2)/Numsam;

p_E3m=mean(p_E3)/Numsam;


p_Am=mean(p_A)/Numsam;

p_Em=mean(p_E)/Numsam;

 %% Mean First and Second Moments

q = q1/M;


Q=  Q1/M;
```

```
if order==0;

Pi_I=[1-pe(jj) pe(jj)]'*[1 1];
Pi_IO=(kron(Pi_I,eye(3)))*blkdiag(S_1,S_2);

sA1=(kron(Pi_IO,eye(1)))*blkdiag(A_0^2,A_1^2,A_1^2,A_0^2,A_1^2,A_1^2);

elseif order==1;
Pi_IO=(kron(Pi_I,eye(3)))*blkdiag(S_1,S_2);

sA1=(kron(Pi_IO,eye(1)))*blkdiag(A_0^2,A_1^2,A_1^2,A_0^2,A_1^2,A_1^2);

else

    Pi_IO=kron(Pi_I, eye(3))*blkdiag(S_1,S_2,S_1,S_2);
    sA2=(kron(Pi_IO, eye(1)))*blkdiag(A_0^2, A_1^2, A_1^2, A_0^2,
    A_1^2, A_1^2,A_0^2, A_1^2, A_1^2,A_0^2, A_1^2, A_1^2);


end;

sr_sA1=max(abs(eig(sA1)));
```

```
subplot(2,1,2);

hold on;

%plot the second Lyapunov exponent

n=1:Numsam+1;

plot(n*log10(sr_sA1),'r:');


fname=strcat('RE1_',outputfname,num2str(pe*100),'_M_',num2str(M),'_',num2str(Nu

save (fname);
```

B-3 Recovery Example 2

B-3.1 Program Code for Theoretical Prediction

plot_sr_scriptA1_Mr2: This program produces plots for the Recovery Example 2, $pi_{E|E}$ v/s spectral radius of $\mathcal{A}_1$ (i.e., using cross-chain process $\boldsymbol{\rho} = (\boldsymbol{\nu}, \boldsymbol{\theta})$ when $M_R = 2$.

```
%Fix \Pi_{A|A A}=0.45

pi_r_00=0.45;


%System parameter in the normal mode

A_0=0.99;
```

```
%System parameter in the recovery mode
A_1=1.0112;

%State transition matrices for the finite-state machines
S_1=[1 0 1;
    0 0 0;
    0 1 0];

S_2=[0 0 1;
    1 0 0;
    0 1 0];

%Vary probability \Pi{A|E E}, \Pi{E|E E}
pi_r_11=0.05:0.05:1;

sz=size(pi_r_11,2);

for jj=1:sz;

    PA(jj)=(1-pi_r_11(jj))/(1-pi_r_11(jj)+1-pi_r_00(jj));

    PE(jj)=(1-pi_r_00(jj))/(1-pi_r_11(jj)+1-pi_r_00(jj));
```

```
    Pi_I=[pi_r_00 1-pi_r_11(jj); 1-pi_r_00 pi_r_11(jj)];

    Pi_I0=kron(Pi_I, eye(3))*blkdiag(S_1,S_2);
    sA1=(kron(Pi_I0, eye(1)))*blkdiag(A_0^2, A_1^2, A_1^2, A_0^2, A_1^2, A_1^2)

    sr_scriptA1(jj)=max(abs(eig(sA1)));

end;
%Plot pi_{eta_2,eta_2) v/s spectralradius(scriptA1);
figure(1);
plot(pi_r_11,sr_scriptA1, 'r*');
title(strcat('Simplified Recovery Example', ' A_0= ', num2str(A_0),
 ' A_1= ', num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));
 figure(2);
plot(PE,sr_scriptA1, 'r*');
title(strcat(' Recovery Example 2', ' A_0= ', num2str(A_0),
' A_1= ', num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));
```

plot_sr_scriptA1_Mr3: This program produces plots for the Recovery Example 2, $\Pi_{E|E}$ v/s spectral radius of scriptA1 (i.e., using cross-chain process $\rho = (\boldsymbol{\nu}, \boldsymbol{\theta})$) when

$M_R = 3.$

```
%Fix \Pi_{A|A A}=0.45

pi_r_00=0.45;


%System parameter in the normal mode

A_0=0.99;


%System parameter in the recovery mode

A_1=1.0112;


%State transition matrices for the finite-state machines

S_1=[1 0 0 1;

     0 0 0 0;

     0 1 0 0;

     0 0 1 0];


S_2=[0 0 0 1;

    1 0 0 0;

    0 1 0 0;

    0 0 1 0];
```

```
%Vary probability \Pi{A|E E}, \Pi{E|E E}

pi_r_11=0.05:0.05:1;


sz=size(pi_r_11,2);


for jj=1:sz;


    Pi_I=[pi_r_00 1-pi_r_11(jj); 1-pi_r_00 pi_r_11(jj)];

    PA(jj)=(1-pi_r_11(jj))/(1-pi_r_11(jj)+1-pi_r_00(jj));

    PE(jj)=(1-pi_r_00(jj))/(1-pi_r_11(jj)+1-pi_r_00(jj));


    Pi_IO=kron(Pi_I, eye(3))*blkdiag(S_1,S_2);


    sA1=(kron(Pi_IO, eye(1)))*blkdiag(A_0^2, A_1^2, A_1^2, A_1^2,

     A_0^2, A_1^2, A_1^2,A_1^2);


    sr_scriptA1(jj)=max(abs(eig(sA1)));


end;

%Plot pi_{eta_2,eta_2) v/s spectralradius(scriptA1);

figur(1);

plot(pi_r_11,sr_scriptA1, 'r*');
```

```
title(strcat(' Recovery Example 2', ' A_0= ', num2str(A_0),
' A_1= ', num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));

figure(2);
plot(PE,sr_scriptA1, 'r*');
title(strcat(' Recovery Example 2', ' A_0= ', num2str(A_0),
' A_1= ', num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));
```

plot_sr_scriptA1_Mr5: This program produces plots for the Recovery Example 2, $\Pi_{E|E}$ v/s spectral radius of scriptA1 (i.e., using cross-chain process $\rho = (\boldsymbol{\nu}, \boldsymbol{\theta})$) when $M_R = 5$

```
%Fix \Pi_{A|A A}=0.45
pi_r_00=0.45;

%System parameter in the normal mode
A_0=0.99;

%System parameter in the recovery mode
A_1=1.0112;

%State transition matrices for the finite-state machines
S_1=[1 0 0 0 0 1;
     0 0 0 0 0 0;
```

```
     0 1 0 0 0 0;
     0 0 1 0 0 0;
     0 0 0 1 0 0;
     0 0 0 0 1 0];

S_2=[0 0 0 0 0 1;
     1 0 0 0 0 0;
     0 1 0 0 0 0;
     0 0 1 0 0 0;
     0 0 0 1 0 0;
     0 0 0 0 1 0];

%Vary probability \Pi{A|E E}, \Pi{E|E E}
pi_r_11=0.05:0.05:1;

sz=size(pi_r_11,2);

for jj=1:sz;

    Pi_I=[pi_r_00 1-pi_r_11(jj); 1-pi_r_00 pi_r_11(jj)];
    PA(jj)=(1-pi_r_11(jj))/(1-pi_r_11(jj)+1-pi_r_00(jj));
    PE(jj)=(1-pi_r_00(jj))/(1-pi_r_11(jj)+1-pi_r_00(jj));
```

```
    Pi_I0=kron(Pi_I, eye(3))*blkdiag(S_1,S_2);
    sA1=(kron(Pi_I0, eye(1)))*blkdiag(A_0^2, A_1^2, A_1^2, A_1^2, A_1^2,
    A_1^2, A_0^2, A_1^2, A_1^2,A_1^2, A_1^2);

    sr_scriptA1(jj)=max(abs(eig(sA1)));

end;
%Plot pi_{eta_2,eta_2) v/s spectralradius(scriptA1);
figure(1);
plot(pi_r_11,sr_scriptA1, 'r*');
title(strcat(' Recovery Example 2', ' A_0= ', num2str(A_0), ' A_1= ',
 num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));

figure(2);
plot(PE,sr_scriptA1, 'r*');
title(strcat(' Recovery Example 2', ' A_0= ', num2str(A_0), ' A_1= ',
num2str(A_1), '\Pi_{aa}= ',num2str(pi_r_00)));
```

B-3.2 Program Code for Monte Carlo Simulation

sim_recovery_ex2: This program simulates Recovery Example 2 for the given number of recovery states $M_R = 3$, $\Pi_{A|A} = 0.45$ and $\Pi_{E|E} = 0.8$.

```
sim_recovery_ex2(outputfname)

M=1000, Numsam=10000;
paa=0.45 ; pee=0.8;

M_R=3;

A_0=0.99;

A_1=1.0112;
    %% Vector of the chain's states
    state=zeros(1,Numsam+1);
    %% Vector of the chain's transitions
    numbers=state;
    %% Vector of the first-order moment.
    q1=state;
    %% Vector of the second-order moment.
    Q1=state;
```

```
    %% Vector of the mean first-order moment.
    q=state;
    %% Vector of the mean second-order moment.
    Q=state;
    %% Time counters
    tick1 = 0;
     %% Number of zero states in the simulated chain
    tick2 = 0;


    z     = zeros(1,M);
%% Number of one states in the simulated chain
    o     = zeros(1,M);
%% Number of zero to zero transferences in the sim. chain
    z2z   = zeros(1,M);
%% Number of one to one transferences in the sim. chain
    o2o   = zeros(1,M);
%% Number of zero to one transferences in the sim chain
    z2o   = zeros(1,M);
%% Number of one to zero transferences in the sim chain
    o2t   = zeros(1,M);
%% Probability of goingt from 0 to 0
    pi00  = zeros(1,M);
%% Probability of goingt from 0 to 1
```

```
    pi01  = zeros(1,M);
%% Probability of goingt from 1 to 0
    pi10  = zeros(1,M);
%% Probability of goingt from 1 to 1
    pi11  = zeros(1,M);


    Nom   = zeros(1,M);


    Rld   = zeros(1,M);


    N2N   = zeros(1,M);


    N2R   = zeros(1,M);


    R2N   = zeros(1,M);


    R2R   = zeros(1,M);


    piNN  = zeros(1,M);


    piNR  = zeros(1,M);


    piRN  = zeros(1,M);
```

```
    piRR  = zeros(1,M);

for k=1:M

fprintf('loop M = %3d, tic...',k);

N=Gen_disc_markov(paa,pee,Numsam);
%N=N';

Rec_state=zeros(1, Numsam+1);
% Equilibrium statistics of N: pk,PA,PE

zs=N==zeros(1,Numsam+1);

% The occurrence of N=0
z(k)=dot(zs,zs);


nzs=N~=zeros(1,Numsam+1);

% The occurrence of N>0
nz(k)=dot(nzs,nzs);
```

```
os=N==ones(1,Numsam+1);

% The occurrence of N=1

o(k)=dot(os,os);

% Equilibrium state probabilities

pkx(k,:)=[z(k) o(k)]/(Numsam+1);

% Probability of disturbance absent(N=0)

PAx(k)=z(k)/(Numsam+1);

% Probability of disturbance exists(N>0)

PEx(k)=1-PAx(k);

% Transitional statistics of N: pikkp1, pikkm1, PAE, PEA

lszs=[zs(2:Numsam+1) 0];

lsnzs=[nzs(2:Numsam+1) 0];
```

```
lsos=[os(2:Numsam+1) 0];

% Count the number of transition occurrences

% From mode N=0 to mode N=0

z2z(k)=dot(zs.*lszs,zs.*lszs);

% From mode N=1 to mode N=1

o2o(k)=dot(os.*lsos,os.*lsos);

% From mode N=0 to mode N=1

z2o(k)=dot(zs.*lsos,zs.*lsos);

% From mode N=1 to mode N=0

o2z(k)=dot(os.*lszs,os.*lszs);


% Calculate the corresponding transition rates

if z(k)==0


pi11(k)=nan;

pi00(k)=nan;

pi01(k)=nan;

pi10(k)=nan;


else


pi00(k)=z2z(k)/z(k);

pi01(k)=z2o(k)/z(k);
```

```
pi10(k)=o2z(k)/o(k);

pi11(k)=o2o(k)/o(k);


end


piaam=nanmean(pi00);

piaem=nanmean(pi01);

pieam=nanmean(pi10);

pieem=nanmean(pi11);

piaasd=nanstd(pi00);

pieesd=nanstd(pi11);

piaesd=nanstd(pi01);

pieasd=nanstd(pi10);



PAm=nanmean(PAx);

PEm=nanmean(PEx);

PAsd=nanstd(PAx);

PEsd=nanstd(PEx);


%% State vector of the plant

x=zeros(size(x0));

%% Consider the initial chain state and the initial conditions:
```

```
% Generate state vector initial conditions
x=unifrnd(-1*norm(x0),norm(x0),size(x0,1),size(x0,2));
% accumulate first-order moment
q1(1) = q1(1) + norm(x);
% accumulate second-order moment
Q1(1) = Q1(1) + norm(x*x');



    c=0;


    tic
    for i=1:Numsam


        if Rec_state(i)==0


            x=A0*x;
            if N(i)==0
                Rec_state(i+1)=0;
            else
                Rec_state(i+1)=1;
            end;
        else
            c=c+1;
```

```
            x=A1*x;
            if c < N_R
            Rec_state(i+1)=1;
            else
            Rec_state(i+1)=0;
            c=0;
            end;

        end;

    %% Accumulate first moment
    q1(i)= q1(i) + norm(x);
    %% Accumulate second moment
    Q1(i)= Q1(i) + norm(x'*x);

    end;
    % Equilibrium statistics of N: pk,PA,PE

    tick1=toc;
    tick2=tick2+toc;
  Noms=Rec_state==zeros(1,Numsam+1);

% The occurrence of Rec_state=Nominal
```

```
  Nom(k)=dot(Noms,Noms);



  Rlds=Rec_state~=zeros(1,Numsam+1);

% The occurrence of Rec_state >0

  Rld(k)=dot(Rlds,Rlds);

% Equilibrium state probabilities

pkx(k,:)=[Nom(k) Rld(k)]/(Numsam+1);

% Probability of disturbance absent(N=0)



PNx(k)=Nom(k)/(Numsam+1);

% Probability of disturbance exists(N>0)

PRx(k)=Rld(k)/(Numsam+1);

% Transitional statistics of N: pikkp1, pikkm1, PAE, PEA
```

```
fprintf('toc = %8.2f total time = %8.2f \n',tick1, tick2);


if (mod(k,100)==0)


q = q1/k;


Q=  Q1/k;


%% Mean First and Second Moments

N=   N/k;


fname=strcat('RE2_',outputfname,'_',

num2str(k/100),'_','N_R','_',num2str(N_R),'_',

num2str(A0),'_',num2str(A1),'paa_',num2str(paa),

'_pee_',num2str(pee),'_M_',num2str(M),'_',num2str(Numsam),'.mat');


save (fname);

 %% End of Montecarlo Runs

        end



%% Mean First and Second Moments

    q = q1/k;
```

```
    Q=  Q1/k;

    N=   N/k;

if k<100

    fname=strcat('Recovery_Example_2_',outputfname,
    '_','N_R','_',num2str(Mrd),'_',num2str(A0),
    '_',num2str(A1),'paa_',num2str(paa),'_pee_',num2str(pee),
    '_M_',num2str(M),'_',num2str(Numsam),'.mat');

        save (fname);

    end

end;

figure;
subplot(2,1,1);

plot(log10(q(1:Numsam)));
xlabel('Samples');
```

```
ylabel('log_{10} (q)');
title(strcat('Recovery Example 3_', ' A0= ', num2str(A0), ',
 A1= ', num2str(A1),
' N_R= ', num2str(N_R),' p_{aa}= ', num2str(paa), ' p_{ee}= ',
num2str(pee), ' M = ',num2str(M), 'Num of Samples =',
 num2str(Numsam)));

subplot(2,1,2);

plot(log10(Q(1:Numsam)));
xlabel('Samples');
ylabel('log_{10} trace(Q)');

S_1=[1 0 0 1;
     0 0 0 0;
     0 1 0 0;
     0 0 1 0];

S_2=[0 0 0 1;
    1 0 0 0;
    0 1 0 0;
    0 0 1 0];
```

```
Pi_IO=kron(Pi_I, eye(3))*blkdiag(S_1,S_2);

    sA1=(kron(Pi_IO, eye(1)))*blkdiag(A_0^2, A_1^2, A_1^2, A_1^2,

     A_0^2, A_1^2, A_1^2,A_1^2);


%plot the second Lyapunov exponent

hold on;

n=1:Numsam+1;

plot(n*log10(max(abs(eig(sA1)))),'r:');


title(strcat('RE2_',outputfname,'_',' A0= ', num2str(A0), ',

A1= ', num2str(A1), ' N_R= ', num2str(N_R),' p_{aa}= ', num2str(paa),

' p_{ee}= ',num2str(pee), ' M = ',num2str(M), 'Number of Samples =',

 num2str(Numsam)));
```

B-4 Recovery Example 3

B-4.1 Program Code for Theoretical Prediction

plot_sr_scriptA1_NAb: This program produces theoretical plots for $\Pi_{E|E}$ v/s spectral radius of $\mathcal{A}_1$ and P_E versus spectral radius $\mathcal{A}_1$ for Recovery Example 3 when $N_{Ab} = 1, 2, 5$.

```
%Fix P_{AA}

pi_r_00=0.6;


pi_r_11=0:0.05:1;


A0=0.99; A1=1.039; A2=0.96; A3=1.08;


N_Ab=[1 2 5];

for l=1:3


if N_Ab(l)==1;


N=6; pl='r*';


S_0=[1 0 1 0 1 1;

    0 0 0 0 0 0;

    0 1 0 0 0 0;

    0 0 0 0 0 0;

    0 0 0 1 0 0;

    0 0 0 0 0 0];


S_1=[0 0 0 0 0 1;

    1 0 0 0 0 0;
```

```
    0 1 0 0 0 0;

    0 0 1 0 0 0;

    0 0 0 1 0 0;

    0 0 0 0 1 0];


elseif N_Ab(1)==2


N=7; pl='g+'


S_0=[1 0 1 0 1 0 1 ;

    0 0 0 0 0 0 0;

    0 1 0 0 0 0 0;

    0 0 0 0 0 0 0;

    0 0 0 1 0 0 0;

    0 0 0 0 0 0 0;

    0 0 0 0 0 1 0];


S_1=[0 0 0 0 0 0 1;

    1 0 0 0 0 0 0;

    0 1 0 0 0 0 0;

    0 0 1 0 0 0 0;

    0 0 0 1 0 0 0;

    0 0 0 0 1 0 0;
```

```
    0 0 0 0 0 1 0];

elseif N_Ab(1)==5

N=10; pl='kx'

S_0=[1 0 1 0 1 0 1 0 0 1;
     0 0 0 0 0 0 0 0 0 0;
     0 1 0 0 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0;
     0 0 0 1 0 0 0 0 0 0;
     0 0 0 0 0 0 0 0 0 0;
     0 0 0 0 0 1 0 0 0 0;
     0 0 0 0 0 0 1 0 0 0;
     0 0 0 0 0 0 0 1 0 0;
     0 0 0 0 0 0 0 0 1 0];

S_1=[0 0 0 0 0 0 0 0 0 1;
    1 0 0 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0 0 0;
    0 0 1 0 0 0 0 0 0 0;
    0 0 0 1 0 0 0 0 0 0;
    0 0 0 0 1 0 0 0 0 0;
```

```
    0 0 0 0 0 1 0 0 0 0

    0 0 0 0 0 0 1 0 0 0;

    0 0 0 0 0 0 0 1 0 0;

    0 0 0 0 0 0 0 0 1 0];

end;



sz1=size(pi_r_00,2);

sz2=size(pi_r_11,2);



PA=zeros(sz1*sz2,1);

PE=zeros(sz1*sz2,1);

k=1;

for i=1:sz1

for jj=1:sz2
```

```
Pi_I=[pi_r_00(i) 1-pi_r_00(i); 1-pi_r_11(jj) pi_r_11(jj)]';

PA(k)=(1-pi_r_11(jj))/(1-pi_r_00(i)+1-pi_r_11(jj));

PE(k)=(1-pi_r_00(i))/(1-pi_r_00(i)+1-pi_r_11(jj));

if N_Ab(l)==1

N=6; pl='r*';

Pi_I0=kron(Pi_I, eye(N))*blkdiag(S_0,S_1);


sA1=(kron(Pi_I0,eye(size(A0,1)^2)))*(blkdiag(kron(A0,A0),kron(A1,A1),
kron(A2,A2), kron(A1,A1), kron(A2,A2), kron(A3,A3),kron(A0,A0),
kron(A1,A1),kron(A2,A2), kron(A1,A1), kron(A2,A2), kron(A3,A3)));


elseif N_Ab(l)==2
N=7; pl='g+' ;

Pi_I0=kron(Pi_I, eye(N))*blkdiag(S_0,S_1);
sA1=(kron(Pi_I0,eye(size(A0,1)^2)))*(blkdiag(kron(A0,A0),kron(A1,A1),
```

```
kron(A2,A2), kron(A1,A1),kron(A2,A2), kron(A3,A3), kron(A3,A3),
kron(A0,A0),kron(A1,A1),kron(A2,A2), kron(A1,A1),kron(A2,A2),
kron(A3,A3), kron(A3,A3)));



elseif N_Ab(1)==5
    N=10; pl='kx';
    Pi_I0=kron(Pi_I,eye(N))*blkdiag(S_0,S_1);



    sA1=(kron(Pi_I0,eye(size(A0,1)^2)))*(blkdiag(kron(A0,A0),
    kron(A1,A1),kron(A2,A2), kron(A1,A1),kron(A2,A2),
    kron(A3,A3), kron(A3,A3), kron(A3,A3), kron(A3,A3),
kron(A1,A1),kron(A2,A2), kron(A3,A3), kron(A3,A3),
kron(A3,A3), kron(A3,A3), kron(A3,A3)));


end;


spectralrad(k)=max(abs(eig(sA1)));


k=k+1;


end;
```

```
end;



figure(1);
hold on;

plot(pi_r_11,spectralrad,pl);
xlabel('\Pi_{E|E}');
ylabel('\rho(sA1)');
title(strcat('Recovery Example 3', ' A0= ', num2str(A0), ' A1= ',
num2str(A1), ', A2= ', num2str(A2),', A3= ', num2str(A3),'
N_{ab}= ', num2str(N_Ab(1)),' p_{aa}= ', num2str(pi_r_00)));

grid on;

figure(2);
hold on;
plot(PE, spectralrad,pl);

xlabel('P_{E}');
ylabel('\rho(sA1)');
```

```
title(strcat('Recovery Example 3', ' A0=', num2str(A0), '
A1=', num2str(A1), ', A2= ', num2str(A2),', A3= ', num2str(A3),'
N_{ab}= ', num2str(N_Ab(1)),' p_{aa}= ', num2str(pi_r_00)));


grid on;
end;
end;
```

B-4.2 Program Code for Monte Carlo Simulation

sim_recovery_ex3: This program simulates Recovery Example 3 when $N_R = 1$, $N_{Rs} = 2$, $N_{Ab} = 1$, $M = 1000$, $Numsam = 10000$ for transition probabilities of the Markov input $\Pi_{A|A} = 0.6$ and $\Pi E|E = 0.3$.

```
sim_recovery_ex3(outputfname)


A0=0.99;


A1=1.039;


A2=0.96;


A3=1.08;
```

```
N_Rd=1;

N_Rs=2;

N_Ab=1;

paa=0.6;

pee=0.3;



M=1000;

Numsam=10000;

    %% Vector of the chain's states
    state=zeros(1,Numsam+1);
 %% Vector of the chain's transitions
    numbers=state;
%% Vector of the first-order moment.
    q1=state;
%% Vector of the second-order moment.
```

```
    Q1=state;
%% Vector of the mean first-order moment.
    q=state;
%% Vector of the mean second-order moment.
    Q=state;
 %% Time counters
    tick1 = 0;


    tick2 = 0;



%% Number of zero states in the simulated chain
    z     = zeros(1,M);
%% Number of one states in the simulated chain
    o     = zeros(1,M);
 %% Number of zero to zero transferences in the sim. chain
    z2z   = zeros(1,M);
 %% Number of one to one transferences in the sim. chain
    o2o   = zeros(1,M);
%% Number of zero to one transferences in the sim chain
    z2o   = zeros(1,M);
 %% Number of one to zero transferences in the sim chain
    o2t   = zeros(1,M);
```

```
%% Probability of goingt from 0 to 0
    pi00  = zeros(1,M);
%% Probability of goingt from 0 to 1
    pi01  = zeros(1,M);
%% Probability of goingt from 1 to 0
    pi10  = zeros(1,M);
%% Probability of goingt from 1 to 1
    pi11  = zeros(1,M);

    Nom   = zeros(1,M);

    Rld   = zeros(1,M);

    Rel   = zeros(1,M);

    Abrt  = zeros(1,M);



for k=1:M

fprintf('loop M = %3d, tic...',k);
```

```
N=Gen_disc_markov(paa,pee,Numsam);

%N=N';


Rec_state=zeros(1, Numsam+1);

    % Equilibrium statistics of N: pk,PA,PE


  zs=N==zeros(1,Numsam+1);


  z(k)=dot(zs,zs);              % The occurrence of N=0


  nzs=N~=zeros(1,Numsam+1);


  nz(k)=dot(nzs,nzs);           % The occurrence of N>0


  os=N==ones(1,Numsam+1);
```

```
  o(k)=dot(os,os);              % The occurrence of N=1

  % Equilibrium state probabilities

  pkx(k,:)=[z(k) o(k)]/(Numsam+1);

  % Probability of disturbance absent(N=0)
    PAx(k)=z(k)/(Numsam+1);

  % Probability of disturbance exists(N>0)

  PEx(k)=1-PAx(k);

% Transitional statistics of N: pikkp1, pikkm1, PAE, PEA

  lszs=[zs(2:Numsam+1) 0];
  lsnzs=[nzs(2:Numsam+1) 0];
  lsos=[os(2:Numsam+1) 0];

% Count the number of transition occurrences
% From mode N=0 to mode N=0
z2z(k)=dot(zs.*lszs,zs.*lszs);
% From mode N=1 to mode N=1
```

```
o2o(k)=dot(os.*lsos,os.*lsos);
% From mode N=0 to mode N=1
 z2o(k)=dot(zs.*lsos,zs.*lsos);
 % From mode N=1 to mode N=0
 o2z(k)=dot(os.*lszs,os.*lszs);


% Calculate the corresponding transition rates
if z(k)==0
    pi11(k)=nan;
    pi00(k)=nan;
    pi01(k)=nan;
    pi10(k)=nan;
else
    pi00(k)=z2z(k)/z(k);
    pi01(k)=z2o(k)/z(k);
    pi10(k)=o2z(k)/o(k);
    pi11(k)=o2o(k)/o(k);
             end


     piaam=nanmean(pi00);
     piaem=nanmean(pi01);
     pieam=nanmean(pi10);
     pieem=nanmean(pi11);
```

```
piaasd=nanstd(pi00);

pieesd=nanstd(pi11);

piaesd=nanstd(pi01);

pieasd=nanstd(pi10);

PAm=nanmean(PAx);

PEm=nanmean(PEx);

PAsd=nanstd(PAx);

PEsd=nanstd(PEx);


   %% State vector of the plant

   x=zeros(size(x0));

   %% Consider the initial chain state and the initial conditions:

   % Generate state vector initial conditions

   x=unifrnd(-1*norm(x0),norm(x0),size(x0,1),size(x0,2));


    % accumulate first-order moment

   q1(1) = q1(1) + norm(x);


   % accumulate second-order moment

   Q1(1) = Q1(1) + norm(x*x');



c_2=0;
```

```
c_3=0;


tic
for i=1:Numsam


    if Rec_state(i)==0


        x=A0*x;
        if N(i)==0
            Rec_state(i+1)=0;
        else
            Rec_state(i+1)=1;
        end;


    elseif Rec_state(i)==1


        x=A1*x;


        Rec_state(i+1)=2;


    elseif Rec_state(i)==2
        c_2=c_2+1;
        x=A2*x;
```

```
        if c_2 < Mrs
            if N(i)==0
                Rec_state(i+1)=2;
            else
                Rec_state(i+1)=1;
            end;
        else

        if N(i)==0
            Rec_state(i+1)=0;
        else
            Rec_state(i+1)=3;
        end;
        c_2=0;
    end;

    else
        c_3=c_3+1;
        x=A3*x;
        if c_3 < Mab
        Rec_state(i+1)=3;
        else
        Rec_state(i+1)=0;
```

```
            c_3=0;
            end;

        end;

  %% Accumulate first moment
  q1(i)= q1(i) + norm(x);
  %% Accumulate second moment
  Q1(i)= Q1(i) + norm(x'*x);

  end;
  tick1=toc;
  tick2=tick2+toc;
Noms=Rec_state==zeros(1,Numsam+1);

 % The occurrence of Rec_state=Nominal
Nom(k)=dot(Noms,Noms);


Rlds=Rec_state==ones(1,Numsam+1);

  % The occurrence of Rec_state Reload
Rld(k)=dot(Rlds,Rlds);
```

```
      % The occurrence of Rec_state Release
  Rels= (Rec_state==2*ones(1,Numsam+1));

  Rel(k)=dot(Rels,Rels);

  % The occurrence of Rec_state Abort
  Abrts=Rec_state==3*ones(1,Numsam+1);

  Abrt(k)=dot(Abrts,Abrts);

% Equilibrium state probabilities

% Probability of disturbance absent(N=0)
PNx(k)=Nom(k)/(Numsam+1);

% Probability of disturbance exists(N>0)
PRx(k)=Rld(k)/(Numsam+1);
PRex(k)=Rel(k)/(Numsam+1);
PAbx(k)=Abrt(k)/ (Numsam+1);

  % Transitional statistics of N: pikkp1, pikkm1, PAE, PEA
end;
```

```
PNm=nanmean(PNx);

PRm=nanmean(PRx);

PRem=nanmean(PRex);

PAbm=nanmean(PAbx);

PNsd=nanstd(PNx);

PRsd=nanstd(PRx);

PResd=nanstd(PRex);

PAbsd=nanstd(PAbx);

fprintf('toc = %8.2f total time = %8.2f \n',tick1, tick2);


        if (mod(k,100)==0)

            %% Mean First and Second Moments


            q = q1/k;

            Q=  Q1/k;

            N=   N/k;


fname=strcat('Recovery_Example3,'_',num2str(k/100),'_','N_{Rs}',

'_',num2str(N_Rs),'_','Mab','_',num2str(Mab),'_',num2str(A0),'_',

num2str(A1),'_',num2str(A2),'_',num2str(A3),'_paa_',num2str(paa),

'_pee_',num2str(pee),'_M_',num2str(M),'_',num2str(Numsam),'.mat');
```

```
        save (fname);
        end


 %% End of Montecarlo Runs
 %% Mean First and Second Moments
    q = q1/k;
    Q=  Q1/k;
    N=   N/k;
    if k<100
 fname=strcat('Recovery Example 3',outputfname,'_',
'N_R','_',num2str(N_Rs),
'_','N_Ab','_',num2str(N_Ab),'_',
num2str(A0),'_',num2str(A1),'_',
num2str(A2),'_',num2str(A3),'_paa_',num2str(paa),
'_pee_',num2str(pee),'_M_',
num2str(M),'_',num2str(Numsam),'.mat');


        save (fname);
    end
end;


figure;
subplot(2,1,1);
```

```
plot(log10(q(1:Numsam)));
xlabel('Samples');
ylabel('log_{10} (q)');
title(strcat('Recovery Example 3', ' A0= ', num2str(A0), ',
A1= ', num2str(A1), 'A2= ', num2str(A2), 'A3= ', num2str(A3),
' N_{Ab}= ', num2str(N_Ab),' p_{aa}= ', num2str(paa), '
p_{ee}= ',num2str(pee), ' M = ',num2str(M),
'Num of Samples =', num2str(Numsam)));

subplot(2,1,2);

plot(log10(Q(1:Numsam)));
xlabel('Samples');
ylabel('log_{10} trace(Q)');
title(strcat('Recovery Example 3', ' A0= ', num2str(A0), ',
 A1= ', num2str(A1), ' N_{Ab}= ', num2str(N_Ab),' p_{aa}= ',
 num2str(paa), ' p_{ee}= ',num2str(pee), ' M = ',num2str(M), '
 Num of Samples =', num2str(Numsam)));

subplot(2,1,2);
hold on;

S_0=[1 0 1 0 1 1;
```

```
    0 0 0 0 0 0;
    0 1 0 0 0 0;
    0 0 0 0 0 0;
    0 0 0 1 0 0;
    0 0 0 0 0 0];

S_1=[0 0 0 0 0 1;
    1 0 0 0 0 0;
    0 1 0 0 0 0;
    0 0 1 0 0 0;
    0 0 0 1 0 0;
    0 0 0 0 1 0];

Pi_IO=kron(Pi_I, eye(6))*blkdiag(S_0,S_1);

sA1=(kron(Pi_IO,eye(size(A0,1)^2)))*(blkdiag(kron(A0,A0),kron(A1,A1),
kron(A2,A2), kron(A1,A1), kron(A2,A2), kron(A3,A3),kron(A0,A0),
kron(A1,A1),kron(A2,A2), kron(A1,A1), kron(A2,A2), kron(A3,A3)));

%plot the second Lyapunov exponent
n=1:Numsam+1;
```

```
plot(n*log10(max(abs(eig(sA1)))),'r:');
```

CURRICULUM VITA

for

SUDARSHAN S. PATILKULKARNI

DEGREES:

-Masters of Electrical Engineering, Old Dominion University, Norfolk, Virginia, USA, December 2000.

-Bachelors of Engineering (Electronics & Communication), Karnataka University, Dharwad, Karnataka, India, July 1996.

ACADEMIC AWARDS:

-2000, Old Dominion University Fellowship from the College of Engineering.

PROFESSIONAL CHRONOLOGY:

-January 2000 to Present: Graduate Research Assistant in Systems Research Lab,
Department of Electrical & Computer Engineering,
Old Dominion University
Director: Dr. W. Steven Gray
Main field: Stochastic System Theory, Hybrid Systems

-January 1999 to December 1999: Graduate Research Assistant in Speech Lab,
Department of Electrical & Computer Engineering,
Old Dominion University
Director: Dr. Stephan Zahorian
Main field: Signal Processing, Speech Recognition

PUBLICATIONS:

1. S. Patilkulkarni, H. Herencia-Zapana, W. S. Gray and O. R. González, "On the Stability of Jump-Linear Systems Driven by Finite-State Machines with Markovian Inputs,' *Proc. 2004 American Control Conference*, Boston, MA, 2004, pp. 2534-2539.
2. W. S. Gray, S. Patilkulkarni and O. R. González, "Stochastic Stability of a Recoverable Computer Control System Modeled as a Finite-State Machine," *Proc. 2003 American Control Conference*, Denver, CO, 2003, pp. 2240-2245.
3. W. S. Gray, S. Patilkulkarni and O. R. González, "Towards Hybrid Models of Recoverable Computer Control Systems," *Proc. 21st Digital Avionics Systems Conference*, Irvine, CA, 2002, pp. 13.C.2.1-13.C.2.8.
4. 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, FL, 2001, pp. 4134-4139.
5. O. R. González, W. S. Gray, A. Tejada and S. Patilkulkarni, "Stability Analysis For Upset Recovery Methods For Electromagnetic Interference," *Proc. 20th Digital Avionics Systems Conference*, 2001, Daytona, FL, pp. 1.C.4.1-1.C.4.9.
6. O. R. González, W. S. Gray and S. Patilkulkarni,"Analysis of Memory Bit Errors Induced by Electromagnetic Interference in Closed Loop Digital Flight Control Systems," *Proc. 19th Digital Avionics Systems Conference*, 2000, Philadelphia, PA, pp. 3.C.5.1-3.C.5.9.

7. W. S. Gray, O. R. González and S. Patilkulkarni, "Stability of Control Systems Subject to Jump Linear Random Perturbations," *Proc. 39th IEEE Conference on Decision and Control*, 2000, Sydney, Australia, pp. 1154-1159.

8. S. A. Zahorian, S. Patilkulkarni, M. Karnjanadecha and C. Brewton, "Speech-to-Text Translation for Indexing and Searching of Audio/Visual Materials for a Digital Library," *Proc. 2000 World Multiconference on Systemics, Cybernetics and Informatics, Image, Acoustic, Speech and Signal Processing*, Part II, SCI 2000/ISAS 2000, Vol. VI., Orlando, FL.

9. M. D. Lee, S. Patilkulkarni, G. D. Capabianco, "Effects of Age and Consistency on Performance Audio Search," CAC 2000, Poster Session, Georgia Tech, Atlanta, GA, 2000.