ABSTRACT

Stability of Non-Diagonalizable Switched Linear Systems on Time Scales

John E. Miller, M.S.E.C.E.

Advisor: Ian A. Gravagne, Ph.D.

This thesis investigates the stability of switched linear systems on time scales using Lyapunov stability theory. First, we focus on the most general case, non-diagonalizable systems with arbitrary switching. Subsequently, a constrained switching case is investigated. Several examples are given for both cases.

Switched linear systems are often found wherever a dynamical system is coupled with supervisory control logic that can abruptly change the system's operating mode, such as in the transmission of a vehicle or on computer-controlled real-time networks. This coupling of a dynamical system with discrete logic is difficult to model on standard time domains, especially if the switching events are non-uniformly spaced. Time scales mathematics allows for these non-uniform time domains.

Stability of Non-Diagonalizable Switched Linear Systems on Time Scales

by

John E. Miller, B.S.E.C.E.

A Thesis

Approved by the Department of Electrical and Computer Engineering

____
Kwang Y. Lee, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Masters of Science in Electrical and Computer Engineering

Approved by the Thesis Committee

____
Ian A. Gravagne, Ph.D., Chairperson

____
John M. Davis, Ph.D.

____
Robert J. Marks II, Ph.D.

Accepted by the Graduate School
August 2009

____
J. Larry Lyon, Ph.D., Dean

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would first like to thank Dr. Gravagne for being an excellent mentor. This thesis would certainly not be possibly without his guidance and leadership. Second, I want to thank the other members of the Baylor Time Scales Research Group, Dr. Davis, Dr. Marks, and Alice Ramos, who have worked right along side me in this research. I am also grateful for the support of the Baylor Engineering program, both for the instruction I have received at the undergraduate and graduate levels as well as for the financial support while being a graduate student. Finally, I want to thank my wife Sarah for her support, especially during the many hours of work I have put into preparing this thesis.

CHAPTER ONE

Background

LTI Systems

Linear systems are ubiquitous in engineering, and much theory has been developed to describe, analyze, and design them [2, 5, 19]. We focus on *causal, lumped-parameter, linear time-invariant* (LTI) systems. All physical systems are *causal* systems, which means that their output depends only on current or previous inputs. *Lumped*, as opposed to *distributed*, systems have a finite number of *state variables.* Throughout this thesis, we will assume all systems are causal and lumped. We can mathematically describe a (causal and lumped) linear system with a state variable model of the form

$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \quad t \in \mathbb{R}, \tag{1.1}$$

where $x : \mathbb{R} \to \mathbb{R}^n$ is the state vector, $u : \mathbb{R} \to \mathbb{R}^p$ is the input, $\dot{x} := \frac{dx(t)}{dt}$, and $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times p}$. In (1.1), we have explicitly noted that the system A and input B matrices are *time-varying. Time-invariant* means that the output of our system is the same for a given initial state and input, regardless of the time that the input was applied. This requires that our system and input matrices be constant, which yields a system of the form

$$\dot{x}(t) = Ax(t) + Bu(t). \tag{1.2}$$

In most of the following discussion, we will assume the system (1.1) is *autonomous* and *closed-loop* (i.e. u is a linear function of x) and can be generally written as

$$\dot{x}(t) = Ax(t). \tag{1.3}$$

One important note about LTI system theory is that continuous-time and discrete-time systems have to be treated as separate cases. Equation (1.2) is the appropriate equation for the continuous-time case but can be discretized by a sample-and-hold method to get the discrete-time case. This means the input to our system will come from a discrete controller, pass through a digital-to-analog converter, and will be piecewise constant. In other words,

$$u[k] := u(kh) = u(t), \qquad \text{for } kh \leq t < (k+1)h, \tag{1.4}$$

where $h > 0$ is the sample period and $k \in \mathbb{N}_0 = \{0, 1, 2, \ldots\}$. Then, if we also compute the response of the system only at $t = kh$, it can be shown [5] that the system takes the form

$$\begin{aligned} x[k+1] &= e^{At}x[k] + \left(\int_0^h e^{A\tau} d\tau\right) Bu[k] \\ &= A_d x[k] + B_d u[k] \end{aligned} \tag{1.5}$$

where the *matrix exponential function*, e^{At}, is defined as

$$e^{At} := I + tA + \frac{t^2}{2} + \ldots = \sum_{k=0}^{\infty} \frac{1}{k} t^k A^k. \tag{1.6}$$

I is the standard identity matrix. In order to rewrite B_d, we also define the *exponent cardinal* as

$$\begin{aligned} \operatorname{expc}(hA) &:= hI + \frac{h^2}{2}A + \frac{h^3}{3}A^2 + \cdots \\ &= \int_0^h e^{A\tau} d\tau \\ &= (e^{hA} - I)A^{-1} \qquad \text{when } A^{-1} \text{ exists.} \end{aligned} \tag{1.7}$$

Thus, $B_d = \operatorname{expc}(hA)B$. One useful property of the expc function is that $\operatorname{expc}(X) \to I$ as $X \to 0$.

We next define the *state transition matrix*. The state transition matrix describes the system's transition from one state to the next. For a linear system of the form (1.1), the state transition matrix, $\Phi(t, t_0)$, is the unique solution of the initial-value problem

$$\dot{X}(t) = A(t)X(t), \quad X(t_0) = I, \tag{1.8}$$

where, in general, A can be time-varying and we have used X to denote an $n \times n$ matrix. Since A is constant for an LTI system, we get the familiar result $\Phi(t, t_0) = e^{A(t-t_0)}$. A discrete version also exists and is $\Phi[k, k_0] = A_d^{(k-k_0)}$, which can be readily deduced from (1.5) with $u[k] \equiv 0$.

Stability

One of the important topics in dynamic systems and, in particular, control theory, is that of *stability.* Stability deals with how "well-behaved" a system is. There are several methods to determine stability of a system [2, 5, 19]. We will first quantify the definition of stability in terms of standard linear systems theory. Then, we will define stability in the sense of Lyapunov and use his second (or direct) method [21] to determine the stability of our system.

We say that the state of the system is *bounded* if there exists some constant x_b such that

$$\|x(t)\| \leq x_b < \infty \quad \forall\, t \geq 0 \tag{1.9}$$

(i.e. the system has a finite response). For some given initial condition, $x(0) = x_0$, the solution of (1.2) is $x(t) = e^{At}x_0$. By applying a similarity transformation to A (and thus to e^{At}), we can show that, for (1.9) to be true, all of the eigenvalues of A must have negative real parts. If A satisfies this condition, it is called a *Hurwitz* matrix. In the discrete case, the solution to (1.5) is $x[k+1] = A_d^k x[0]$ for $u[k] \equiv 0$, which means that all of the eigenvalues of A_d must have magnitude less than one (i.e. inside the unit circle in the z-plane). These are common results in standard control theory. We now move on to Lyapunov theory.

We say that the system (1.2) is *stable* if for every $\varepsilon > 0$ there exists a $\delta > 0$ such that

$$\|x(0)\| \leq \delta \quad \Rightarrow \quad \|x(t)\| \leq \varepsilon \quad \forall\, t \geq 0. \tag{1.10}$$

The system is *asymptotically stable* if it is stable and δ can be chosen so that

$$\|x(0)\| \leq \delta \quad \Rightarrow \quad x(t) \to 0 \text{ as } t \to \infty. \tag{1.11}$$

According to Lyapunov's stability theorem, if there exists a positive definite *Lyapunov function* $V : \mathbb{R}^n \to \mathbb{R}$ and it satisfies

$$\dot{V}(x) < 0, \quad \forall\ x \neq 0, \tag{1.12}$$

then the LTI system (1.2) is asymptotically stable. For LTI systems, the typical method used is to construct a Lyapunov function candidate in the *quadratic* form

$$V(x) = x^T P x \tag{1.13}$$

where, if P is a positive definite symmetric matrix, then V is positive definite. $\dot{V}$ then becomes

$$\begin{aligned} \dot{V}(x) &= x^T P \dot{x} + \dot{x}^T P x \\ &= x^T (PA + A^T P) x \\ &= -x^T Q x \end{aligned} \tag{1.14}$$

where $Q := -(A^T P + PA)$ must be a positive definite symmetric matrix in order for the system to be stable.

Switched Systems

Many dynamical systems feature a combination of continuous and discrete dynamics. Such systems are often called *hybrid systems*. One of the sub-categories of hybrid systems is *switched systems*, where the overall dynamics are described using a number of sub-systems (modeled by continuous- or discrete-time dynamics) coupled with discrete switching events. These systems are the focus of [20], which gives a good introduction and presents much of the theory that has been developed for switched systems.

We define a *switched linear system* as

$$\dot{x}(t) = A_{c(t)}x(t), \quad t \in \mathbb{R}^+ \tag{1.15}$$

where $c : \mathbb{R}^+ \rightarrow \{1, 2, \ldots, m\}$ is a switching signal which chooses one of a finite set (or family) of matrices $A_i \in \mathbb{R}^{n \times n}$ with $i \in \{1, 2, \ldots, m\}$. If each individual system is stable (i.e. each A_i is Hurwitz), then we are guaranteed the existence of a quadratic Lyapunov function for each system. And if we can find a single *common* quadratic Lyapunov function which satisfies

$$A_i^T P + P A_i \leq -Q, \quad \forall\ i = 1, \ldots, m \tag{1.16}$$

for Q a positive definite symmetric matrix, then we say that the switched system (1.15) is asymptotically stable.

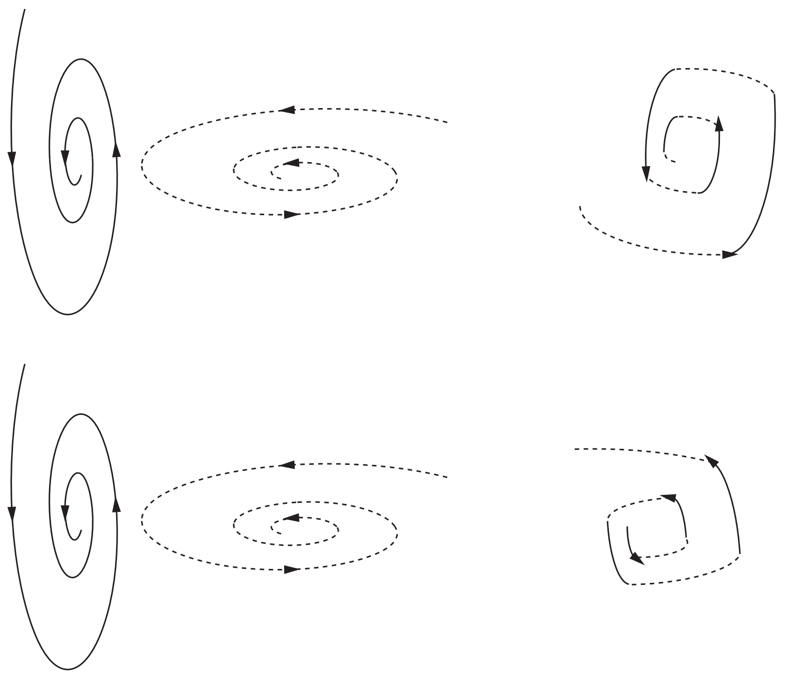


Figure 1.1. Phase plots showing that switching can produce either stability (top) or instability (bottom) from the same sub-systems (which are both stable in this case).

Switched systems are time-varying because of the switching mechanism. Figure 1.1 shows how switching can introduce either stability or instability into a switched system, even when its constituent sub-systems are stable. (Note that this is not as simple as these plots may make it seem.) While the system can no longer be expressed in the form of (1.1), meaning the Hurwitz criteria from the section on Stability does not apply, Lyapunov's stability criteria applies to time-varying systems.

Time Scales

As discussed in the previous two sections, modeling dynamical systems may involve both continuous- and discrete-time mathematics. This presents a significant difficulty when dealing with hybrid systems. *Time scales* is a branch of mathematics, first introduced by Stefan Hilger in [15], which seeks to both unify and extend the theory from continuous and discrete mathematics. Accordingly, one of the major areas of study in time scales is that of *dynamic equations on time scales*, which includes this thesis and a number of other publications [1, 4, 8, 9, 10, 12, 18, 22]. Bohner and Peterson present the basics of time scales theory in their introductory text [3]. We will present several definitions in this section to develop the theory necessary for the rest of the discussion.

A *time scale*, denoted as $\mathbb{T}$, is any closed subset of the real numbers, $\mathbb{R}$. Figure 1.2 shows several examples of common time scales.

The *forward jump operator*, $\sigma(t)$, and the *backward jump operator*, $\rho(t)$, are defined by

$$\sigma(t) := \inf\{s \in \mathbb{T} : s > t\} \tag{1.17}$$

$$\rho(t) := \sup\{s \in \mathbb{T} : s < t\}. \tag{1.18}$$

These return the next and previous points in the time scale, respectively. We say that a point $t \in \mathbb{T}$ is *left-dense*, *right-dense*, *left-scattered*, or *right-scattered* if $\rho(t) = t$, $\sigma(t) = t$, $\rho(t) < t$, or $\sigma(t) > t$, respectively. In this thesis, we call a contiguous set of

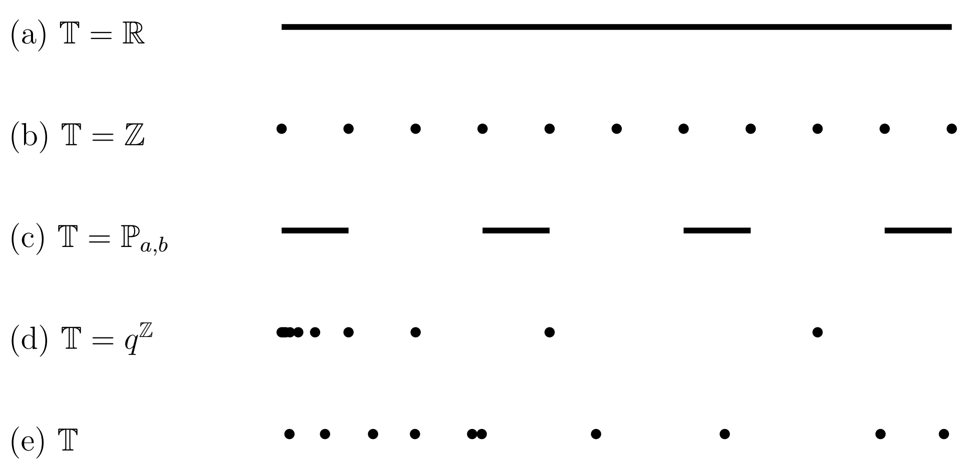

(a) $\mathbb{T} = \mathbb{R}$
(b) $\mathbb{T} = \mathbb{Z}$
(c) $\mathbb{T} = \mathbb{P}_{a,b}$
(d) $\mathbb{T} = q^{\mathbb{Z}}$
(e) $\mathbb{T}$

Figure 1.2. Time scale examples: (a) Continuous-time (real numbers) (b) Uniform discrete-time (integers) (c) Periodic continuous intervals (d) "Quantum" time scale with $q = 2$ (e) Random discrete.

points which are both left- and right-dense, a *continuous interval* (or just *continuous*), and we call a contiguous set of points which are both left- and right-scattered a *discrete interval* (or just *discrete*).

The *graininess*, $\mu(t)$, is equivalent to the distance between the current and next points (i.e. the step size) and is defined by

$$\mu(t) := \sigma(t) - t. \tag{1.19}$$

In this thesis, we will also use the idea of a bounded graininess on the time scale $\mathbb{T}$, defined by

$$\mu_{\max}(t) := \sup_{t\in\mathbb{T}} \mu(t). \tag{1.20}$$

The graininess will play a major role in the rest of the discussion.

The graininess and forward jump operators can also operate on each other or themselves. For instance, $\sigma(\sigma(t))$ will give us the "next, next" point and $\mu(\sigma(t))$ will give us the graininess of the next point. An alternate notation, which we will use from now on, is $\mu^\sigma(t) := \mu(\sigma(t))$.

We define a *delta* (or *Hilger*) *derivative* by

$$f^{\Delta}(t) := \frac{f(\sigma(t)) - f(t)}{\mu(t)} \tag{1.21}$$

if t is right-scattered and f is *differentiable* at t (see [3] for a formal definition of differentiable). Note that if $\mathbb{T}$ has a left-scattered maximum M, then we can only talk about the derivative on $\mathbb{T}^{\kappa} = \mathbb{T} - \{M\}$. All subsequent references to the delta derivative on time scales with discrete intervals will be subject to this condition, even if not shown explicitly. If t is right-dense and f is *differentiable* at t, then it will take the form

$$f^{\Delta}(t) = \lim_{s \to t} \frac{f(t) - f(s)}{t - s}. \tag{1.22}$$

We next define the *Hilger complex numbers* as

$$\mathbb{C}_h := \left\{ z \in \mathbb{C} : z \neq -\frac{1}{h} \right\} \tag{1.23}$$

for $h > 0$. Along with this, the *Hilger imaginary circle* (or just *Hilger circle*) is defined as

$$\mathbb{I}_h := \left\{ z \in \mathbb{C}_h : \left| z + \frac{1}{h} \right| = \frac{1}{h} \right\} \tag{1.24}$$

These two ideas are easiest to see graphically, as shown in Figure 1.3.

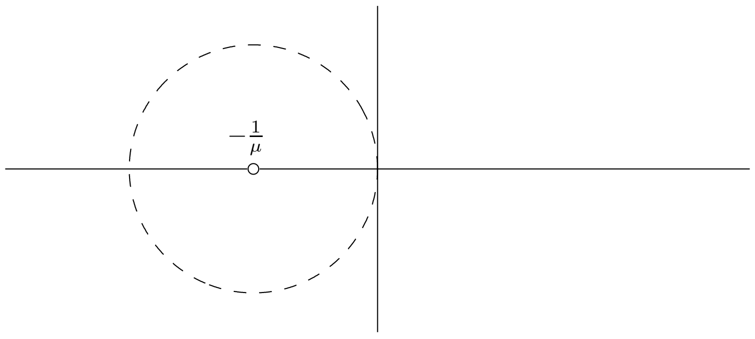


Figure 1.3. The Hilger complex plane ($\mathbb{C}_\mu$) and Hilger circle ($\mathbb{I}_\mu$).

We also introduce “circle plus” addition, defined by

$$z \oplus w := z + w + zwh \tag{1.25}$$

for $z, w \in \mathbb{C}_h$. Note that a “circle minus” operator also exists, but we will not make use of it in this thesis. The function $p : \mathbb{T} \to \mathbb{R}$ is *regressive* if

$$1 + \mu(t)p(t) \neq 0, \quad \forall\, t \in \mathbb{T}^\kappa. \tag{1.26}$$

An $n \times n$ matrix-valued function $A(t)$ on a time scale $\mathbb{T}$ is *regressive* if

$$I + \mu(t)A(t) \quad \text{is invertible for all } t \in \mathbb{T}^\kappa. \tag{1.27}$$

Note that A is regressive if and only if the eigenvalues, $\lambda_i(t)$, of A are regressive for all $1 \leq i \leq n$.

The general, first-order *time scales dynamic equation* is of the form

$$x^\Delta(t) = p(t)x(t), \quad x(t_0) = x_0 \tag{1.28}$$

where p is regressive. Similar to the standard exponential function, the *time scale exponential function*, $e_p(t, t_0)$, is defined in [3] and is the unique solution to (1.28) with $x_0 = 1$. Again, similar to the case on $\mathbb{R}$, we can guarantee that a time scales LTI system (like (1.28), except p is constant) is stable if p is inside the Hilger circle for all t. In other words,

$$|1 + \mu(t)p| < 1, \quad \forall\, t \in \mathbb{T}. \tag{1.29}$$

The general, first-order *time scales linear dynamic equation* is of the form

$$x^\Delta(t) = A(t)x(t), \quad x(t_0) = x_0. \tag{1.30}$$

where A is regressive. The *time scale matrix exponential function*, $e_{A(t)}(t, t_0)$ is the unique solution to (1.30) with $x_0 = I$. For LTI systems, A is constant, so we have $e_A(t, t_0)$. For the case where $\mathbb{T} = \mathbb{R}$, this becomes the familiar matrix exponential function $e_A(t,t_0) = e^{A(t-t_0)}$. When $\mathbb{T} = h\mathbb{Z}$, we have $e_A(t,t_0) = (I+hA)^{\frac{(t-t_0)}{h}}$. We define the *time scale state transition matrix*, $\Phi(t,t_0)$, where $\Phi_{A(t)}(t,t_0) = e_{A(t)}(t,t_0)$ for the system (1.30). In accordance with the examples just given (for A constant), when $\mathbb{T} = \mathbb{R}$, $\Phi_A(t,t_0) = e^{A(t-t_0)}$, and when $\mathbb{T} = h\mathbb{Z}$, $\Phi_A(t,t_0) = (I+hA)^{\frac{(t-t_0)}{h}}$.

We can guarantee stability of the system (1.30) if all of the eigenvalues λ of A satisfy

$$|1+\mu(t)\lambda_i| < 1, \quad \forall\ 1 \le i \le n \text{ and } t \in \mathbb{T}. \tag{1.31}$$

This should seem fairly intuitive in light of the scalar time scale case and the matrix standard ($\mathbb{R}$) case. From [7], we have that exponential and asymptotic stability are equivalent for the system (1.30). Thus, the condition (1.31) implies both.

Table 1.1. Analogous properties of standard time domains.

$\mathbb{R}$	$\mathbb{Z}$	$\mathbb{T}$
Continuous ($\mu = 0$)	Discrete ($\mu = 1$)	Hybrid ($\mu(t)$)
Standard derivative $f'(t) = \lim_{h\to 0} \frac{f(t+h)-f(t)}{h}$	Forward difference $\Delta f(t) = f(t+1) - f(t)$	Delta (Hilger) derivative $f^{\Delta}(t) = \frac{f(\sigma(t))-f(t)}{\mu(t)}$
Ordinary differential equation $y' = \lambda y,\ y(t_0) = 1$ $\Downarrow$ $y(t) = e^{\lambda(t-t_0)}$	Difference equation $y[k+1] = \lambda y[k],\ y[k_0] = 1$ $\Downarrow$ $y[k] = \lambda^{t-t_0}$	Dynamic equation $y^{\Delta} = \lambda y,\ y(t_0) = 1$ $\Downarrow$ $y(t) = e_{\lambda}(t,t_0)$
Stability in the s plane $\text{Re}\ \lambda < 0$	Stability in the z plane $|\lambda| < 1$	Stability in Hilger's complex plane $|1+\mu\lambda| < 1$

Table 1 gives a summary of various properties of the standard time domains, $\mathbb{R}$ and $\mathbb{Z}$, compared with an arbitrary time scale $\mathbb{T}$.

CHAPTER TWO

General Temporal Region of Stability

Introduction

We now examine the stability of switched LTI systems on time scales. We begin by defining the system and making several assumptions. We then apply Lyapunov's theorem to the system and develop constraints on the time scale itself in order to maintain stability. These constraints form a two-dimensional plane we term the *temporal region of stability* (or TRoS).

Problem Setup

Consider the set of subsystem matrices $A_1, A_2, \ldots, A_m \in \mathbb{R}^{n\times n}$ with switching signal $c : \mathbb{T} \to \{1, 2, \ldots, m\}$, where

$$x^{\Delta}(t) = A_{c(t)}x(t), \quad t \in \mathbb{T}, t \geq 0 \text{ and } x(0) = x_0. \tag{2.1}$$

We make the following assumptions about this system and the underlying time scale:

A1 The switching signal c is arbitrary over $\mathbb{T}$.

A2 The eigenvalues of all of the A_i are strictly within the Hilger circle for all $t \in \mathbb{T}$. (This means each A_i is stable with respect to the time scale $\mathbb{T}$.)

A3 Each A_i is regressive. (Meaning that $(I + \mu(t)A_i)^{-1}$ exists $\forall\ t \in \mathbb{T}$.)

A4 The family $\{A_i\}$ is pairwise commutative, i.e. $A_iA_j = A_jA_i\ \forall\ i \neq j$.

A5 $\mathbb{T}$ has the following properties: (i) $0 \in \mathbb{T}$, (ii) $\mathbb{T}$ is unbounded above, (iii) $\mathbb{T}$ has graininess $0 < \mu_{\min} \leq \mu(t) \leq \mu_{\max}$ for all $t \in \mathbb{T}$.

The reasons for these assumptions are:

R1 We want to investigate the most general case, where the system can switch from any subsystem A_i to an other subsystem A_j at any time. (This also happens to yield the most conservative TRoS.)

R2 Required if we want arbitrary switching. For example, if we included an unstable subsystem, we could choose an "arbitrary" switching signal that never leaves that unstable subsystem and, therefore, makes the switched system unstable.

R3 We need to be able to invert this term, which is the definition of regressivity given in (1.27).

R4 This is a specific and commonly studied class of switched systems. We use this assumption to guarantee that the set of subsystem matrices is *simultaneously* Jordan-diagonalizable.

R5 The systems we are interested in evolve on discrete, non-uniform graininess time scales. Also, the stability requirement (A2) for the A_i's yields a $\mu_{\max}$ which corresponds to the smallest Hilger circle encompassing all the eigenvalues of all the A_i's.

Diagonal Case

The results for the simultaneously diagonalizable (or just diagonal) case have been derived previously in [12]. These results are analogous to the results presented in this thesis, and thus, we show only one development (the more general one) of them. In [12], we define a region $\mathcal{R}$ as the set of all $\{\mu(t), \mu^\sigma(t)\}$ such that

$$\prod_{k=1}^{m} K_k^{-1} K_k^\sigma > (I + \mu\Lambda_i^*)(I + \mu\Lambda_i), \quad \text{for } i = 1, \ldots, m \tag{2.2}$$

with $0 < \mu_{\min} \leq \mu(t) \leq \mu_{\max}$ for all $t \in \mathbb{T}$ and $A_i = S^{-1}\Lambda_i S$, where Λ_i is diagonal and $n \times n$. K_k is defined as

$$K_k := 2 \text{ Re } \Lambda_k + \mu\Lambda_k^*\Lambda_k \tag{2.3a}$$

$$K_k^\sigma := 2 \text{ Re } \Lambda_k + \mu^\sigma\Lambda_k^*\Lambda_k. \tag{2.3b}$$

It turns out that the inequalities in (2.2) yield a set of $2m$ polynomials in the variables μ and μ^σ, under which the system is stable. We will explore this further and give examples after the proof of the general (non-diagonal) theorem.

Jordan Epsilon Form

Apart from the above assumptions, we do not want to limit the cases we explore any further. Thus, we want to include all possible subsystem matrices in our investigation. This requires that the A_i's not be simply *diagonalizable* (as was the case for (2.2)), but, more generally, *Jordan-diagonalizable* as it is known that every square matrix is similar to a Jordan matrix [23]. Jordan-diagonalizable means that the A_i's have a similarity transformation $A_i = S^{-1}J_iS$ which yields a Jordan form matrix J_i. By assumption A4 and a theorem [17], the A_i's are *simultaneously* Jordan-diagonalizable, meaning that there exists a single similarity transform, S, for all of them.

When applying the similarity transform $A = S^{-1}JS$, J is typically put in Jordan canonical form whenever it cannot be completely diagonal. The Jordan canonical form is defined as

$$J = \begin{bmatrix} \lambda & 1 & & & \\ & \lambda & 1 & & \\ & & \lambda & 1 & \\ & & & & \ddots \end{bmatrix},$$

where we note two important observations. The first is that we have only a single eigenvalue. This is because distinct eigenvalues are not "coupled" through the matrix and can thus be separated into uncoupled Jordan blocks, which we can then treat independently. The second is that the choice of ones (1's) on the superdiagonal is arbitrary. Thus, we will define what we call the "Jordan epsilon form." This is a little known, but useful, technique for proving certain arguments [17, p.128]. Let $A = S_\varepsilon^{-1}J_\varepsilon S_\varepsilon$ with

$$J_\varepsilon = \begin{bmatrix} \lambda & \varepsilon & & & \\ & \lambda & \varepsilon & & \\ & & \lambda & \varepsilon & \\ & & & & \ddots \end{bmatrix},$$

where ε is arbitrary and $0 < \varepsilon < 1$. J_ε comes from the generalized eigenvector equations

$$\begin{aligned} A\bar{x}_1 = \lambda\bar{x}_1,\ A\bar{x}_2 = \lambda\bar{x}_2 + \varepsilon\bar{x}_1,\ A\bar{x}_3 = \lambda\bar{x}_3 + \varepsilon\bar{x}_2,\ \ldots \\ \Rightarrow (A-\lambda I)\bar{x}_1 = 0,\ \frac{(A-\lambda I)}{\varepsilon}\bar{x}_2 = \bar{x}_1,\ \frac{(A-\lambda I)}{\varepsilon}\bar{x}_3 = \bar{x}_2,\ \ldots \\ \Rightarrow (A-\lambda I)\bar{x}_1 = 0,\ \frac{(A-\lambda I)^2}{\varepsilon}\bar{x}_2 = 0,\ \frac{(A-\lambda I)^3}{\varepsilon^2}\bar{x}_3 = 0,\ \ldots \end{aligned}$$

Thus, let $S_\varepsilon = [\bar{x}_1\ \varepsilon\bar{x}_2\ \varepsilon^2\bar{x}_3\ \ldots]$ where the $\bar{x}_i$ are the eigenvectors of A. $S_\varepsilon^{-1}J_\varepsilon S_\varepsilon$ is a valid similarity transform (as multiplication by a scalar doesn't change the fact that the $\bar{x}_i$ are eigenvectors). We term J_ε the *Jordan epsilon form.* Henceforth, let $S = S_\varepsilon$ and $J = J_\varepsilon$.

It will be useful to look at J as the sum of two matrices (called an SN decomposition; see [16]), $J = \Lambda + N$, where

$$\Lambda = \begin{bmatrix} \lambda & 0 & & & \\ & \lambda & 0 & & \\ & & \lambda & 0 & \\ & & & & \ddots \end{bmatrix}, \quad N = \begin{bmatrix} 0 & \varepsilon & & & \\ & 0 & \varepsilon & & \\ & & 0 & \varepsilon & \\ & & & & \ddots \end{bmatrix}.$$

N is a nilpotent matrix with ε on the superdiagonal, and Λ is a diagonal matrix of the eigenvalues. Note that N being nilpotent of order n means that $N^n = [0]$, where N (and A, Λ, and J) is $n \times n$.

Lyapunov Analysis

As was pointed out in the section on Switched Systems in Chapter 1, stability of the individual subsystems (now given by assumption A2) is not enough to ensure stability of the switched system. This gives rise to the question that we seek to answer: how do we guarantee stability of the switched system as a whole? Just as we did on $\mathbb{R}$ in Chapter 1, we will apply Lyapunov's second (or direct) method on $\mathbb{T}$. Note that all quantities can be assumed to be time-varying except for the A_i, unless otherwise indicated. Without loss of generality, we restrict the following analysis to $m = 2$.

To investigate stability of (2.1), we define a Lyapunov candidate

$$V = x^T P x \tag{2.4}$$

where $P = P^T > 0$. To ensure stability, we need

$$\begin{aligned} & V^\Delta < 0 \\ \Rightarrow \quad & A_i^T P + P A_i + \mu A_i^T P A_i + (I + \mu A_i^T) P^\Delta (I + \mu A_i) < 0. \end{aligned} \tag{2.5}$$

We now set

$$A_i^T P + P A_i + \mu A_i^T P A_i = -M_i, \tag{2.6}$$

where $M_i = M_i^T > 0$ and solve for P. It can be shown [22] that P solves (2.6) for all i if

$$P(t) = \int_{\mathbb{S}_t} \Phi_{A_i}(s,0)^T M_i(t) \Phi_{A_i}(s,0) \Delta s \tag{2.7}$$

and

$$M_1(t) = \int_{\mathbb{S}_t} \Phi_{A_2}(s,0)^T Q(t) \Phi_{A_2}(s,0) \Delta s \tag{2.8a}$$

$$M_2(t) = \int_{\mathbb{S}_t} \Phi_{A_1}(s,0)^T Q(t) \Phi_{A_1}(s,0) \Delta s, \tag{2.8b}$$

where $\mathbb{S}_t = \mu(t)\mathbb{N}_0$, $Q = Q^T > 0$ is an arbitrary "seed" matrix, and $\Phi_{A_i}(s,0)$ is the transition matrix that solves $y^\Delta(s) = A_i y(s)$ with $s \in \mathbb{S}_t$ and an initial condition $y(0) = y_0$. For each fixed t, $\mathbb{S}_t$ is a constant-graininess time scale, so

$$\Phi_{A_i}(s,0) = e_{A_i}(s,0) = (I + \mu(t) A_i)^{\frac{s}{\mu(t)}}. \tag{2.9}$$

Substituting (2.6) into (2.5) yields

$$(I + \mu A_i)^T P^\Delta (I + \mu A_i) - M_i < 0, \qquad \text{for } t \in \mathbb{T} \tag{2.10}$$

where $P^\Delta = \frac{P^\sigma - P}{\mu}$. Note that the only terms in (2.10) which depend on t are μ and μ^σ.

We can now pose the following, more specific, question: given a time $t \in \mathbb{T}$, what is the region $\mathcal{R} \in \mathbb{R}^2$ such that $\{\mu(t), \mu^\sigma(t)\} \in \mathcal{R}$ satisfies (2.10) for all i? We need to develop a couple of tools before we can answer this question.

Transition Matrix

Triangular Transition Matrix

Remembering that $s \in \mathbb{S}_t := \mu(t)\mathbb{N}_0$, we may now rewrite (2.9) with $z = \frac{s}{\mu} \in \mathbb{N}_0$ (where we drop t to simplify notation) to obtain

$$\Phi_A(s,0) = e_A(s,0) = (I + \mu A)^{\frac{s}{\mu}} = (I + \mu A)^z. \tag{2.11}$$

Applying the Jordan epsilon similarity transform to A then gives

$$\begin{aligned}\Phi_A(s,0) &= (I + \mu A)^z \\ &= (I + \mu S^{-1} J S)^z \\ &= S^{-1}(SS^{-1} + \mu J)^z S \\ &= S^{-1}(I + \mu J)^z S \\ &= S^{-1}\Phi_J(s,0)S. \end{aligned} \tag{2.12}$$

Expanding Φ_J as a binomial series we get

$$\begin{aligned}\Phi_J(s,0) &= (I + \mu J)^z \\ &= (I + \mu(\Lambda + N))^z \\ &= ((1 + \mu\lambda)I + \mu N)^z \\ &= \sum_{k=0}^{z} \binom{z}{k} (1 + \mu\lambda)^{z-k} (\mu N)^k. \end{aligned} \tag{2.13}$$

If $z < n-1$, then this is the final form of the binomial series, and the series is finite. If $z \geq n-1$, then the series is truncated at $n-1$ because N is nilpotent, and we get

$$\begin{aligned} \Phi_J(s,0) &= (1+\mu\lambda)^z I + \binom{z}{1}(1+\mu\lambda)^{z-1}(\mu N) + \dots \\ &\qquad + \binom{z}{n-1}(1+\mu\lambda)^{z-n+1}(\mu N)^{n-1} + [0] + [0] + \dots \\ &= (I+\mu\Lambda)^z + \sum_{k=1}^{n-1}\binom{z}{k}(1+\mu\lambda)^{z-k}(\mu N)^k \end{aligned} \tag{2.14}$$

which is still finite for any z. Now, let

$$\Phi_J(s,0) = (I+\mu\Lambda)^z + E(s) = e_\Lambda(s,0) + E(s) \tag{2.15}$$

where

$$E(s) := \sum_{k=1}^{z}\binom{z}{k}(1+\mu\lambda)^{z-k}(\mu N)^k \qquad \left(\text{recall } z = \frac{s}{\mu}\right). \tag{2.16}$$

In other words, E is an error matrix that depends on ε, is upper-triangular with zeros on the diagonal, and is at most of order ε^{n-1}. Note that $\|E\| \to 0$ as $\varepsilon \to 0$.

Error Terms

In the following discussion, many such error terms will appear, so we list them here for the sake of clarity:

$$E_i(s) := \sum_{k=1}^{z}\binom{z}{k}[1+\mu\lambda_i]^{z-k}[\mu N]^k \tag{2.17a}$$

$$E1_i := \int_{\mathbb{S}_t}\left[e_{\bar{\Lambda}_i}(s,0)E_i(s) + E_i^*(s)e_{\Lambda_i}(s,0) + E_i^*(s)E_i(s)\right]\Delta s \tag{2.17b}$$

$$E2 := -E1_1[2\text{ Re }\Lambda_2 + \mu\Lambda_2^*\Lambda_2]^{-1} - E1_2[2\text{ Re }\Lambda_1 + \mu\Lambda_1^*\Lambda_1]^{-1} + E1_1E1_2 \tag{2.17c}$$

$$E3_i := \mu(I+\mu J_i)^{-*}E1_j(I+\mu J_i)^{-1} - E2^\sigma + E2 \tag{2.17d}$$

$$E4_i := \mu(1+\mu\lambda_i)N^* + \mu(1+\mu\bar{\lambda}_i)N + \mu^2 NN^* \tag{2.17e}$$

$$E5_i := E4_i[(I+\mu J_i)(I+\mu J_i^*)]^{-1} + K_1E3_iK_2 \tag{2.17f}$$

where $j = 2, 1$ and $*$ denotes conjugate transpose. Note: $E2$ is the same for $i = 1, 2$.

Transition Matrix Quadratic Integral

In proving the main theorem of this thesis, we will make use of the following lemma:

Lemma 2.1 (Transition Matrix Quadratic Integral). *Let $A = S^{-1}JS$, where $J \in \mathbb{C}^{n\times n}$ is a Jordan epsilon form matrix, and be stable on time scale $\mathbb{S} = h\mathbb{N}_0$ for $h > 0$. Then*

$$\int_{\mathbb{S}} \Phi_J(s,0)^* \Phi_J(s,0)\Delta s = -[2 \; Re \; \Lambda + h\Lambda^*\Lambda]^{-1} + E1, \tag{2.18}$$

where $J = \Lambda + N$, with Λ diagonal and N nilpotent, and $E1$ is defined as in (2.17b).

Proof. Using the definition of Φ_J from (2.15) gives

$$\int_{\mathbb{S}} \Phi_J(s,0)^* \Phi_J(s,0)\Delta s = \int_{\mathbb{S}} [e^*_{\Lambda}(s,0) + E^*(s)][e_{\Lambda_i}(s,0) + E(s)]\Delta s. \tag{2.19a}$$

By assumption A2, the left hand side of (2.19a) converges. We can expand it to see

$$\begin{aligned} &= \int_{\mathbb{S}} [e_{\bar{\Lambda}}(s,0)e_{\Lambda}(s,0) + e_{\bar{\Lambda}}(s,0)E(s) + E^*(s)e_{\Lambda_i}(s,0) + E^*(s)E(s)]\,\Delta s \\ &= \int_{\mathbb{S}} e_{\bar{\Lambda}}(s,0)e_{\Lambda}(s,0)\Delta s + E1, \end{aligned} \tag{2.19b}$$

where $E1$ is defined as in (2.17b). $E1$ must be finite because the integral on the left side of (2.19a) converges and the first term of (2.19b) is positive definite.

The first term of (2.19b) is an integral of a diagonal matrix, thus we only need to prove statements about it for the scalar case. The following result is repeated from [12]. Let $\lambda = \Lambda(1,1)$. As stated in (2.9), the transition matrix equates to the time scale exponential on the constant graininess time scale $\mathbb{S}$. Thus, $\Phi_\lambda(s,0) = e_\lambda(s,0)$. We then have

$$\begin{aligned} \int_{\mathbb{S}} \Phi^*_\lambda(s,0)\Phi_\lambda(s,0)\Delta s &= \int_{\mathbb{S}} e_{\bar{\lambda}}(s,0)e_\lambda(s,0)\Delta s \\ &= \int_{\mathbb{S}} e_{\bar{\lambda}\oplus\lambda}(s,0)\Delta s \end{aligned} \tag{2.20}$$

where $\oplus$ is the time scales "circle plus" operator defined in (1.25). The exponent evaluates to

$$\bar{\lambda} \oplus \lambda = 2 \text{ Re } (\lambda) + h|\lambda|^2 = \frac{|1 + h\lambda|^2 - 1}{h}. \tag{2.21}$$

By assumption A2, we have that $|1 + h\lambda| < 1$. Thus, $\bar{\lambda} \oplus \lambda$ is strictly within the Hilger circle (i.e. stable) on $\mathbb{S}$ and the integral becomes

$$\int_{\mathbb{S}} e_{\bar{\lambda}\oplus\lambda}(s,0)\Delta s = \frac{1}{-2 \text{ Re } (\lambda) - h|\lambda|^2}. \tag{2.22}$$

We can apply this scalar equation to our diagonal matrix to get

$$\int_{\mathbb{S}} e_{\bar{\Lambda}}(s,0)e_{\Lambda}(s,0)\Delta s = -[2 \text{ Re } \Lambda + h\Lambda^*\Lambda]^{-1}. \tag{2.23}$$

Substituting this into (2.19), we have

$$\int_{\mathbb{S}} \Phi_J(s,0)^*\Phi_J(s,0)\Delta s = -[2 \text{ Re } \Lambda + h\Lambda^*\Lambda]^{-1} + E1 \tag{2.24}$$

which is the lemma statement. $\square$

In order to simplify (2.18), we define the following quantities

$$K_i := 2 \text{ Re } \Lambda_i + \mu\Lambda_i^*\Lambda_i \tag{2.25a}$$

$$K_i^\sigma := 2 \text{ Re } \Lambda_i + \mu^\sigma\Lambda_i^*\Lambda_i. \tag{2.25b}$$

Note that K_i^{-1} always exists because K_i is diagonal and has non-zero eigenvalues as a result of assumption A2. We can now rewrite (2.18) in terms of (2.25), yielding

$$\int_{\mathbb{S}} \Phi_{J_i}(s,0)^*\Phi_{J_i}(s,0)\Delta s = -K_i^{-1} + E1_i. \tag{2.26}$$

Next, note

$$\lim_{\varepsilon\to 0}\int_{\mathbb{S}} \Phi_{J_i}(s,0)^*\Phi_{J_i}(s,0)\Delta s = -K_i^{-1}. \tag{2.27}$$

Using Lemma 2.1 (or more concisely, (2.26)), we can now answer the question we posed earlier: given a time $t \in \mathbb{T}$, what is the region $\mathcal{R} \in \mathbb{R}^2$ such that $\{\mu(t), \mu^\sigma(t)\} \in \mathcal{R}$ satisfies (2.10) for all i?

Theorem 2.1 (General TRoS). *Under assumptions A1-A5, given a set of matrices $A_i = S^{-1}J_iS$ for $1 \leq i \leq m$ where the J_i are Jordan epsilon form matrices with $J_i = \Lambda_i + N$ and S is a simultaneous similarity transform, there exists a region $\mathcal{R} \in \mathbb{R}^2$ consisting of pairs $\{\mu, \mu^\sigma\}$ such that*

$$\prod_{k=1}^{m} K_k^{-1}K_k^\sigma > (I + \mu\Lambda_i^*)(I + \mu\Lambda_i), \quad \text{for } i = 1, \ldots, m \tag{2.28}$$

with $0 < \mu_{\min} \leq \mu(t) \leq \mu_{\max}$ for all $t \in \mathbb{T}$.

Proof. As we have done thus far, we will investigate the $m = 2$ case, without loss of generality. Substituting (2.8a) into (2.7) for $i = 1$, we get

$$\begin{aligned} P &= \int_{\mathbb{S}_t} \Phi_{A_1}(s,0)^T M_1 \Phi_{A_1}(s,0)\Delta s \\ &= \int_{\mathbb{S}_t} \Phi_{A_1}(s,0)^T \left(\int_{\mathbb{S}_t} \Phi_{A_2}(r,0)^T Q \Phi_{A_2}(r,0)\Delta r \right) \Phi_{A_1}(s,0)\Delta s \\ &= \int_{\mathbb{S}_t}\int_{\mathbb{S}_t} \Phi_{A_1}(s,0)^T \Phi_{A_2}(r,0)^T Q \Phi_{A_2}(r,0)\Phi_{A_1}(s,0)\Delta r\Delta s. \end{aligned} \tag{2.29}$$

Since Q in (2.8) may be any arbitrary, positive definite matrix, we choose $Q = S^*S$. Substituting this and applying the Jordan epsilon similarity transform $A_i = S^{-1}J_iS$ to (2.29) gives

$$\begin{aligned} P &= \int_{\mathbb{S}_t}\int_{\mathbb{S}_t} \left(S^*\Phi_{J_1}(s,0)^*\Phi_{J_2}(r,0)^*S^{-T}\right) S^*S \left(S^{-1}\Phi_{J_2}(r,0)\Phi_{J_1}(s,0)S\right)\Delta r\Delta s \\ &= S^*\left[\int_{\mathbb{S}_t} \Phi_{J_1}(s,0)^*\Phi_{J_1}(s,0)\Delta s \int_{\mathbb{S}_t} \Phi_{J_2}(r,0)^*\Phi_{J_2}(r,0)\Delta r\right] S. \end{aligned} \tag{2.30}$$

We now use Lemma 2.1 and (2.25) to say

$$\begin{aligned} P &= S^*\left[\int_{\mathbb{S}_t} \Phi_{J_1}^*(s,0)\Phi_{J_1}(s,0)\Delta s \int_{\mathbb{S}_t} \Phi_{J_2}^*(r,0)\Phi_{J_2}(r,0)\Delta r\right] S \\ &= S^*\left[\left(-K_1^{-1} + E1_1\right)\left(-K_2^{-1} + E1_2\right)\right] S \\ &= S^*\left[K_1^{-1}K_2^{-1} + E2\right] S, \end{aligned} \tag{2.31}$$

where $E2$ is defined as in (2.17c). Similarly

$$P^\sigma = S^* \left[K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} + E2^\sigma \right] S. \tag{2.32}$$

Inserting P and P^σ from (2.31) and (2.32) into (2.10) for $i = 1$ and eliminating S gives

$$\frac{1}{\mu}(I + \mu J_1^*) \left[K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} + E2^\sigma - K_1^{-1} K_2^{-1} - E2 \right] (I + \mu J_1) + K_2^{-1} - E1_2 < 0. \tag{2.33}$$

We rearrange to obtain

$$\begin{aligned}
\frac{1}{\mu}(I + \mu J_1^*) \left[K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} - K_1^{-1} K_2^{-1} + E2^\sigma - E2 \right] (I + \mu J_1) &< -K_2^{-1} + E1_2 \\
K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} - K_1^{-1} K_2^{-1} + E2^\sigma - E2 &< \mu (I + \mu J_1)^{-*} \left(-K_2^{-1} + E1_2 \right) (I + \mu J_1)^{-1} \\
K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} - K_1^{-1} K_2^{-1} &< -\mu (I + \mu J_1)^{-*} (I + \mu J_1)^{-1} K_2^{-1} + E3_1,
\end{aligned} \tag{2.34}$$

where $E3$ is defined as in (2.17d). Continuing,

$$\begin{aligned}
K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} - K_1^{-1} K_2^{-1} &< -\mu [(I + \mu J_1)(I + \mu J_1^*)]^{-1} K_2^{-1} + E3_1 \\
K_1 K_2 K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} - I &< -\mu [(I + \mu J_1)(I + \mu J_1^*)]^{-1} K_1 + K_1 E3_1 K_2 \\
K_1 K_2 K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} &< I - \mu [(I + \mu J_1)(I + \mu J_1^*)]^{-1} K_1 + K_1 E3_1 K_2 \\
K_1 K_2 K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} &< [(I + \mu J_1)(I + \mu J_1^*) - \mu K_1][(I + \mu J_1)(I + \mu J_1^*)]^{-1} \\
&\quad + K_1 E3_1 K_2.
\end{aligned} \tag{2.35}$$

Multiplying $(I + \mu J_1)(I + \mu J_1^*)$ out gives

$$\begin{aligned}
(I + \mu J_1)(I + \mu J_1^*) &= I + 2\mu \text{ Re } \Lambda_1 + \mu^2 \Lambda_1 \Lambda_1^* + \mu(1 + \mu\lambda_1) N^* + \mu(1 + \mu\bar{\lambda}_1) N \\
&\quad + \mu^2 N N^* \\
&= I + \mu K_1 + E4_1,
\end{aligned} \tag{2.36}$$

where we define $E4$ as in (2.17e). Thus we have

$$K_1 K_2 K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} < [I + \mu K_1 + E4_1 - \mu K_1][(I + \mu J_1)(I + \mu J_1^*)]^{-1} + K_1 E3_1 K_2$$
$$K_1 K_2 K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} < [I + E4_1][(I + \mu J_1)(I + \mu J_1^*)]^{-1} + K_1 E3_1 K_2$$
$$K_1 K_2 K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} < [(I + \mu J_1)(I + \mu J_1^*)]^{-1} + E4_1[(I + \mu J_1)(I + \mu J_1^*)]^{-1} + K_1 E3_1 K_2. \tag{2.37}$$

Let $E5$ be defined as in (2.17f). Then

$$\begin{aligned} K_1 K_2 K_1^{\sigma^{-1}} K_2^{\sigma^{-1}} &< [(I + \mu J_1)(I + \mu J_1^*)]^{-1} + E5_1 \\ K_1^{-1} K_2^{-1} K_1^{\sigma} K_2^{\sigma} &> \left([(I + \mu J_1)(I + \mu J_1^*)]^{-1} + E5_1\right)^{-1} \\ K_1^{-1} K_2^{-1} K_1^{\sigma} K_2^{\sigma} &> \left([I + \mu K_1 + E4_1]^{-1} + E5_1\right)^{-1} \\ K_1^{-1} K_2^{-1} K_1^{\sigma} K_2^{\sigma} &> \left([(I + \mu \Lambda_1^*)(I + \mu \Lambda_1) + E4_1]^{-1} + E5_1\right)^{-1}. \end{aligned} \tag{2.38}$$

Similarly, $i = 2$ follows.

Recalling the diagonal case and the region $\mathcal{R}$ defined by (2.2), we define a region $\mathcal{R}_\varepsilon$ given by (2.38) for $i = 1, 2$. Then we note that

$$\lim_{\varepsilon \to 0} \mathcal{R}_\varepsilon = \mathcal{R} \tag{2.39}$$

because

$$\lim_{\varepsilon \to 0} \left([(I + \mu \Lambda_i^*)(I + \mu \Lambda_i) + E4_i]^{-1} + E5_i\right)^{-1} = (I + \mu \Lambda_i^*)(I + \mu \Lambda_i). \tag{2.40}$$

Applying this to (2.38) yields the set of inequalities for $m = 2$ in Theorem 2.1:

$$K_1^{-1} K_2^{-1} K_1^{\sigma} K_2^{\sigma} > (I + \mu \Lambda_i^*)(I + \mu \Lambda_i) \qquad \text{for} \;\; i = 1, 2. \tag{2.41}$$

Thus, to guarantee stability of the switched linear system (2.1), it is sufficient to show that $\{\mu(t), \mu^\sigma(t)\} \in \mathcal{R}$, regardless of whether the system is simultaneously diagonalizable or not, which is the essence of the theorem statement. □

This is a favorable result, because $\mathcal{R}$ is significantly easier to compute than $\mathcal{R}_\varepsilon$. We also note an interesting consequence of Theorem 2.1.

Corollary 2.1 (Unit Slope). *Under arbitrary switching, the system* (2.1) *will remain stable if $\mu^\sigma(t) \le \mu(t)$ for all $t \in \mathbb{T}$.*

Proof. We can derive this result by looking at the scalar representation of each side of (2.28) since everything is diagonal. The right-hand side has magnitude

$$|(I + \mu\lambda_k^*)(I + \mu\lambda_k)| \le |(I + \mu\lambda_k^*)||(I + \mu\lambda_k)| = |(I + \mu\lambda_k)|^2 < 1 \tag{2.42}$$

(where k is the index of *all* the eigenvalues of *all* the subsystems) because all the eigenvalues are strictly within the Hilger circle for all $t \in \mathbb{T}$ by assumption A2. The form of the left side is

$$\prod_{k=1}^{m} \frac{(2 \text{ Re } \lambda_k + \mu^\sigma |\lambda_k|^2)}{(2 \text{ Re } \lambda_k + \mu |\lambda_k|^2)}. \tag{2.43}$$

Recall that the Hilger circle is on the negative real axis, meaning each eigenvalue has negative real part. Thus, if $\mu^\sigma(t) \le \mu(t)$, then

$$-(2 \text{ Re } \lambda_k + \mu^\sigma |\lambda_k|^2) > -(2 \text{ Re } \lambda_k + \mu |\lambda_k|^2). \tag{2.44}$$

This implies

$$\left| \prod_{k=1}^{m} \frac{(2 \text{ Re } \lambda_k + \mu^\sigma |\lambda_k|^2)}{(2 \text{ Re } \lambda_k + \mu |\lambda_k|^2)} \right| \ge 1 > |(I + \mu\lambda_k)|^2. \tag{2.45}$$

So, $\mathcal{R}$ always incudes the area $\mu^\sigma(t) \le \mu(t)$ (i.e. below the 45° line where $\mu = \mu^\sigma$). □

Examples

We use MATLAB to generate several examples of the TRoS defined by both (2.28) ($\mathcal{R}$) and (2.10) (equivalent to $\mathcal{R}_\varepsilon$). The solution of the inequalities (2.28) as if they were equalities yields a set of $2m$ polynomials (for each distinct eigenvalue). m of them are outside the $\mu_{\max}$ limit imposed by A2. The minimum under the other m define the TRoS $\mathcal{R}$. The following figures are plots of μ vs. μ^σ containing the polynomial curves just described, a $\mu = \mu^\sigma$ line for reference, and either $\mathcal{R}$ or $\mathcal{R}_\varepsilon$.

The upper limits on the axes are $\mu_{\max}$, which is the maximum graininess allowed by assumption A2 (see R5).

$\mathcal{R}$ appears as a shaded region (under the curves) since it is defined *analytically* by (2.28). $\mathcal{R}_\varepsilon$ appears as an array of individual $\{\mu, \mu^\sigma\}$ points which satisfy (2.10) when checked *numerically.* Equations (2.10) and (2.38) are equivalent conditions (i.e. the same $\{\mu, \mu^\sigma\}$ pairs satisfy them). Note that $\mathcal{R}_\varepsilon$ is more conservative and difficult to compute than $\mathcal{R}$ because of the error terms on the right-hand side of (2.38), but as stated in the proof of Theorem 2.1, $\mathcal{R}_\varepsilon \to \mathcal{R}$ as $\varepsilon \to 0$.

Example 1

Let

$$A_1 = \begin{bmatrix} -1.6 & \varepsilon & 0 \\ 0 & -1.6 & \varepsilon \\ 0 & 0 & -1.6 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -0.4 & \varepsilon & 0 \\ 0 & -0.4 & \varepsilon \\ 0 & 0 & -0.4 \end{bmatrix},$$

which are already in irreducible, Jordan epsilon form. Figure 2.1 is a plot of $\mathcal{R}$, the area given by (2.28). Note that the region is upper-bounded by the minimum of the two curves. Each point in Figures 2.2-2.4 represents a $\{\mu, \mu^\sigma\}$ pair which satisfies (2.10). Notice that as $\varepsilon \to 0$, $\mathcal{R}_\varepsilon \to \mathcal{R}$, just as stated in the proof of Theorem 2.1. Figure 2.4 also demonstrates that (2.28) is equivalent to (2.10) in the limit as $\varepsilon \to 0$ and can be used to establish stability of the system (2.1).

Example 2

Let

$$A_1 = \begin{bmatrix} -0.7 & \varepsilon_1 & 0 \\ 0 & -0.7 & 0 \\ 0 & 0 & -1.1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -1.4 & \varepsilon_2 & 0 \\ 0 & -1.4 & 0 \\ 0 & 0 & -1.4 \end{bmatrix},$$

which is also in Jordan epsilon form. Figures 2.5-2.8 are similar to those in Example 1.

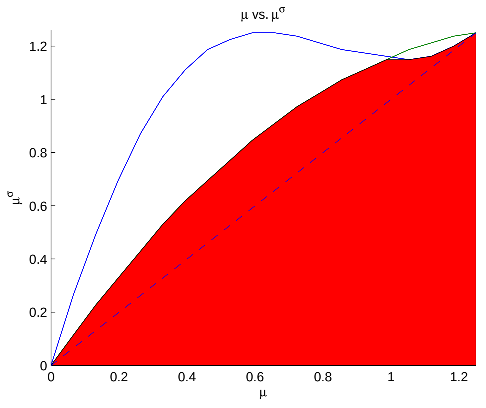


Figure 2.1. $\mathcal{R}$ (for $\varepsilon = 0.3$.)

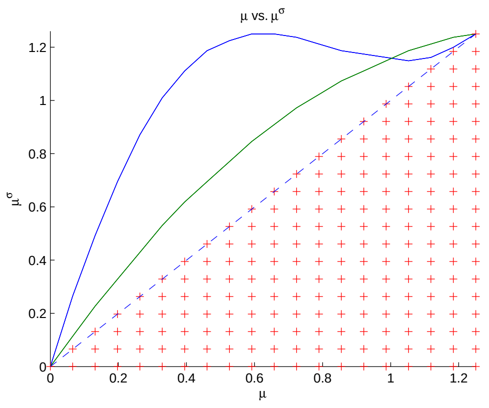


Figure 2.2. $\mathcal{R}_\varepsilon$ for $\varepsilon = 0.3$.

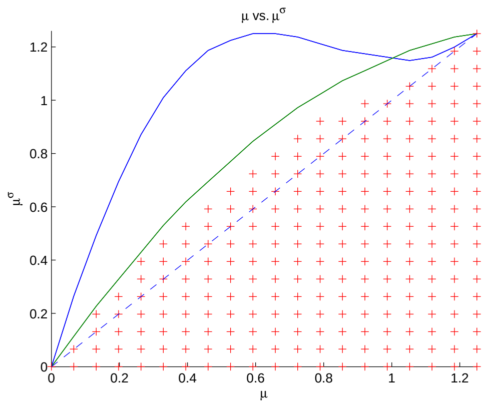


Figure 2.3. $\mathcal{R}_\varepsilon$ for $\varepsilon = 0.3 \times 10^{-5}$.

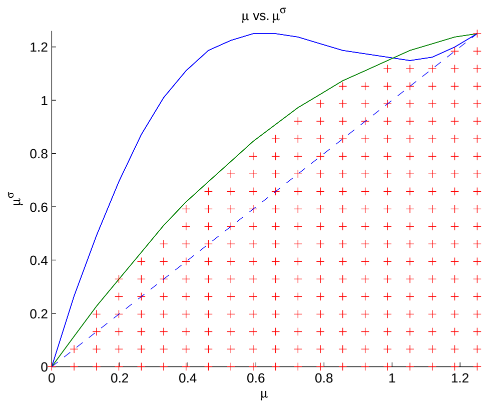


Figure 2.4. $\mathcal{R}_\varepsilon$ for $\varepsilon = 0.3 \times 10^{-15}$.

Example 3

Let

$$A_1 = \begin{bmatrix} -0.522421 & -0.444784 \\ 0.123072 & -1.3577 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -0.750658 & -0.169227 \\ 0.0468251 & -1.06846 \end{bmatrix},$$

which are randomly generated, commuting, *diagonalizable* matrices. Since they are diagonalizable, $\mathcal{R}_\varepsilon = \mathcal{R}$ and $\mathcal{R}$ is given by the diagonal case (2.2). Figure 2.9 is a plot of $\mathcal{R}$, while Figure 2.10 is a plot of the region given by (2.10). Theoretically, these regions should be the same. However, the regions in Figures 2.9 and 2.10 are not equivalent because the "infinite" sums required in (2.7) and (2.8) were truncated

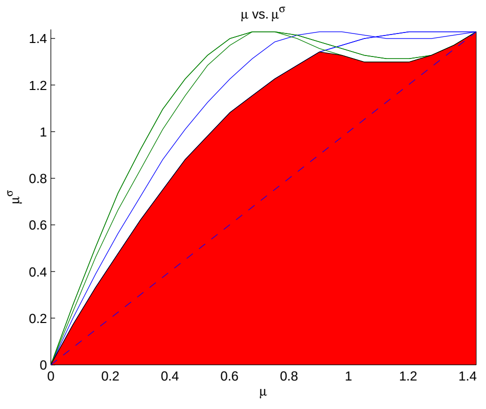


Figure 2.5. $\mathcal{R}$ (for $\varepsilon_1 = 0.2$ and $\varepsilon_2 = 0.1$.)

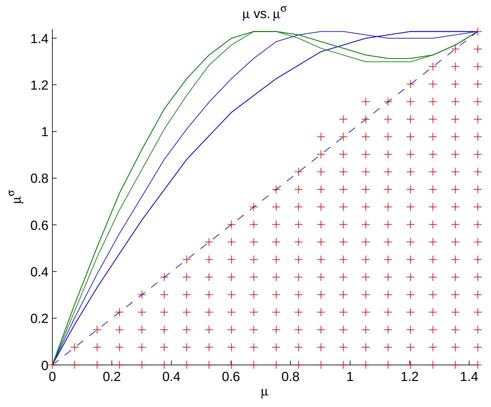


Figure 2.6. $\mathcal{R}_\varepsilon$ for $\varepsilon_1 = 0.2$ and $\varepsilon_2 = 0.1$.

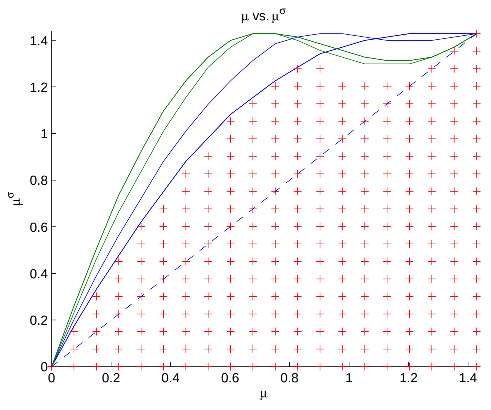


Figure 2.7. $\mathcal{R}_\varepsilon$ for $\varepsilon_1 = 0.2 \times 10^{-5}$ and $\varepsilon_2 = 0.1 \times 10^{-5}$.

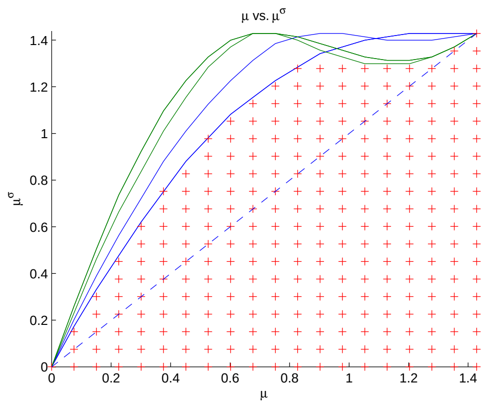


Figure 2.8. $\mathcal{R}_\varepsilon$ for $\varepsilon_1 = 0.2 \times 10^{-15}$ and $\varepsilon_2 = 0.1 \times 10^{-15}$.

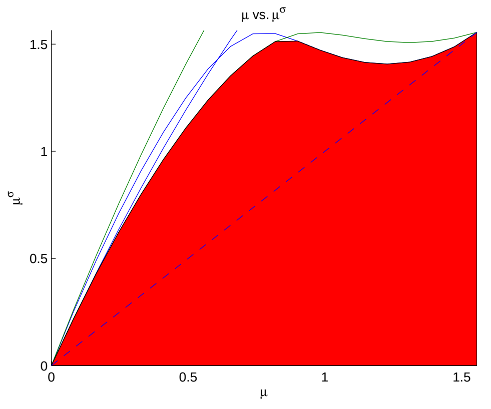


Figure 2.9. $\mathcal{R}$ given analytically by (2.2).

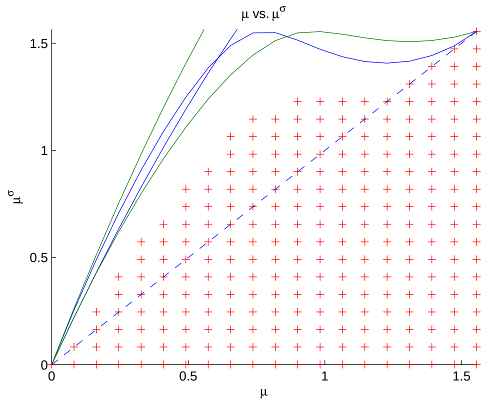


Figure 2.10. Region given by numerical computation of (2.10).

at 1000 terms for computational efficiency. This highlights the point that (2.28) is much easier to compute than (2.10).

CHAPTER THREE

Application to Controller Area Network

Introduction and CAN

While Chapter 2 focused on the general problem of system stability, this chapter will focus on a specific application of those results. We begin by noting that many industrial and automotive systems use discrete controllers which communicate with various sensors and actuators via a single real time network. One common protocol is the *Controller Area Network* (CAN). We will use the this as our motivation, although the ideas presented here are not limited to CAN systems.

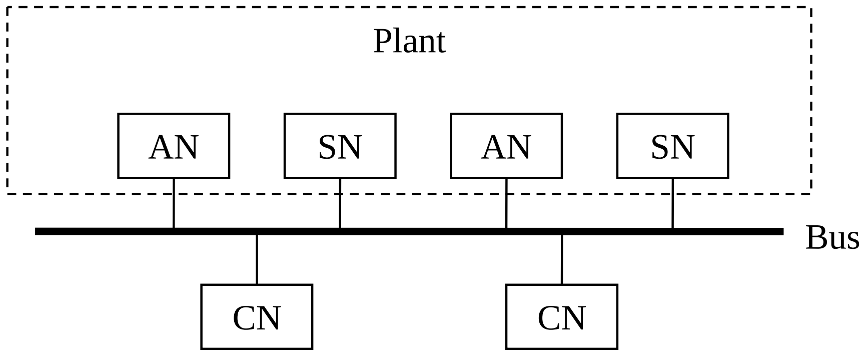


Figure 3.1. CAN network diagram with sensor nodes (SN), actuator nodes (AN), and controller nodes (CN).

Figure 3.1 shows a typical arrangement of nodes on a CAN bus. There are three types of nodes in the network: controllers, sensors, and actuators. Each node receives all of the messages that are transmitted on the bus and simply filters out the ones that are addressed to it. The messages are comprised of an identifier followed by the actual data. The identifier specifies which node the message is for, the type of data in the message, and the priority of the message. If a message collision (two nodes trying to transmit at the same time) occurs, the higher priority message will be allowed to transmit first.

A control loop can be implemented on the network, which at its most basic, involves a control node polling one or more sensors, the sensors responding, and the controller calculating new actuator values and transmitting these to one or more actuators. While this is a typical closed-loop feedback system with a specified sampling period, there are also sporadic *high-speed*, high-priority messages that need to be handled. These come from (often random) events which trigger a sensor, such as depressing the brake pedal in a car or a robot hand making contact with an object. They are termed *high-speed* because they have (possibly very tight) deadlines and must be processed within those deadlines.

One method of handling these sporadic messages is to allow more time (i.e. increase the sampling period) between the periodic messages. This deceases the robustness of the control-loop, however. An alternative tactic is to use *adaptive sampling*, meaning we can adjust the sampling period "on-the-fly." This allows for short periods of high-speed sporadic traffic but maintains system integrity. This type of adaptive control is the focus of [14].

We begin by describing (mathematically) the system using the discretization method from Chapter 1. We then discuss the role of commutativity under the switching constraint and present the main theorem, which is very similar (as should be expected) to Theorem 2.1. We conclude with several examples of systems and their corresponding TRoS's.

Problem Setup

Consider the linear system of the form

$$\dot{x} = Ax + Bu, \quad A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times p} \tag{3.1}$$

$$u = Kx, \quad K \in \mathbb{R}^{p\times n}. \tag{3.2}$$

(Note that K without a subscript always denotes the feedback gain matrix defined here.) Similar to Chapter 1, we can discretize this system (as was done in [14]) by

$$x^{\Delta}(t) = \mathcal{A}(\mu(t))x(t) \tag{3.3}$$

where

$$\mathcal{A}(\mu(t)) := \text{expc}(\mu(t)A)(A + BK). \tag{3.4}$$

Designating a finite number of choices of μ yields the set $\{\mu_1, \mu_2, \ldots, \mu_m\} \in \mathbb{R}^+$ (positive real numbers) and a corresponding set of matrices $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\} \in \mathbb{R}^{n\times n}$ with switching signal $c : \{\mu_i\} \to \{1, 2, \ldots, m\}$. The switched system is then

$$x^{\Delta}(t) = \mathcal{A}_{c(t)}x(t), \quad t \in \mathbb{T}, t \geq 0 \text{ and } x(0) = x_0. \tag{3.5}$$

We modify the assumptions from Chapter 2 as follows:

A1 The switching signal c is constrained as $c : \{\mu_i\} \to \{1, 2, \ldots, m\}$.

A2 The eigenvalues of all of the $\mathcal{A}_i$ are strictly within the smallest Hilger circle of the set $\{\mu_1, \mu_2, \ldots, \mu_m\}$. (This means each $\mathcal{A}_i$ is stable with respect to the time scale $\mathbb{T}$.)

A3 Each $\mathcal{A}_i$ is regressive. (Meaning that $(I + \mu_i\mathcal{A}_i)^{-1}$ exists $\forall\ t \in \mathbb{T}$.)

A4 The family $\{\mathcal{A}_i\}$ is pairwise commutative, i.e. $\mathcal{A}_i\mathcal{A}_j = \mathcal{A}_j\mathcal{A}_i\ \forall\ i, j$. Note: it can be shown that, if A and BK commute, then all of the $\mathcal{A}_i$ commute (see below).

A5 $\mathbb{T}$ has the following properties: (i) $0 \in \mathbb{T}$, (ii) $\mathbb{T}$ is unbounded above, (iii) $\mathbb{T}$ has graininess $0 < \mu_{\min} \leq \mu_i \leq \mu_{\max}$ for $i = 1, 2, 3, \ldots, m$.

Note that A2-A5 are very similar to those from Chapter 2. A1 has changed to indicate a switching constraint.

Commutativity

One question that arises is: "What is the requirement for a set of subsystem matrices $\mathcal{A}_i$ to be commutative?" There are two parts to the answer. First, from the definitions (3.4) and (1.7), we can show that

$$
\begin{aligned}
\mathcal{A}_i\mathcal{A}_j &= \text{expc}(\mu_i A)(A+BK)\text{expc}(\mu_j A)(A+BK) \\
&= \left(I + \frac{1}{2}\mu_i A + \ldots\right)(A+BK)\left(I + \frac{1}{2}\mu_j A + \ldots\right)(A+BK) && \text{(3.6a)} \\
&= \left((A+BK) + \frac{1}{2}\mu_i A(A+BK) + \ldots\right)\left(I + \frac{1}{2}\mu_j A + \ldots\right)(A+BK) && \text{(3.6b)} \\
&= (A+BK)\left(I + \frac{1}{2}\mu_i A + \ldots\right)\left(I + \frac{1}{2}\mu_j A + \ldots\right)(A+BK) && \text{(3.6c)} \\
&= (A+BK)\left(I + \frac{1}{2}\mu_j A + \ldots\right)\left(I + \frac{1}{2}\mu_i A + \ldots\right)(A+BK) && \text{(3.6d)} \\
&= \left((A+BK) + \frac{1}{2}\mu_j (A+BK)A + \ldots\right)\left(I + \frac{1}{2}\mu_i A + \ldots\right)(A+BK) && \text{(3.6e)} \\
&= \left(I + \frac{1}{2}\mu_j A + \ldots\right)(A+BK)\left(I + \frac{1}{2}\mu_i A + \ldots\right)(A+BK) && \text{(3.6f)} \\
&= \mathcal{A}_j\mathcal{A}_i && \text{(3.6g)}
\end{aligned}
$$

Due to the form of (3.6a), we get commutativity if we swap the first and third terms. To do this, we need A and BK to commute. If they do, then we can make the step from (3.6b) to (3.6c). We can swap the middle terms of (3.6c) because they both contain only powers of A and μ_i and μ_j are simply scalars. This gives (3.6d). Moving from (3.6d) to (3.6e) again requires that A and BK commute. Assuming they do, we then get the result in (3.6g), which shows that $\mathcal{A}_i$ and $\mathcal{A}_j$ commute.

Next, if A and BK commute, then $ABK + BKA = 0$. To simplify things, let $X := BK$. We can now vectorize and use the properties of Kronecker products to say

$$
\begin{aligned}
(A^T \otimes_k I)\text{vec } X - (I \otimes_k A)\text{vec } X &= 0 \\
\left[(A^T \otimes_k I) - (I \otimes_k A)\right] \text{vec } X &= 0 \\
\left[(-A^T \otimes_k I) + (I \otimes_k A)\right] \text{vec } X &= 0 \\
(A^T \oplus_k A)\text{vec } X &= 0 && \text{(3.7)}
\end{aligned}
$$

where $\otimes_k$ and $\oplus_k$ are the Kronecker product and Kronecker sum, respectively. The Kronecker sum is defined as

$$(-A^T \oplus_k A) := \left[(-A^T \otimes_k I) + (I \otimes_k A)\right]. \tag{3.8}$$

Let $X = BK$, then (3.7) becomes

$$(-A^T \oplus_k A)(I \otimes_k B)\text{vec } K = 0. \tag{3.9}$$

Thus, A and BK will commute *if and only if* vec K is in the null space of the first two terms [11]. (3.9) is a solvable equation which produces a commuting A and BK. Note that K is not uniquely defined by (3.9). We can use this freedom to specify a set of desired eigenvalues for $A + BK$ and then have an algorithm choose K to put the eigenvalues of $A + BK$ as close to the desired set as possible.

Constrained Temporal Region of Stability

Before presenting Theorem 3.1 (a modified version of Theorem 2.1), we need to note two things. First, the primary difference between 3.1 and 2.1 is that each $\{\mu_i, \mu_j\}$ pairs with a specific $\{\mathcal{A}_i, \mathcal{A}_j\}$. Notice that the indices of the $\mathcal{A}$'s and the μ's are coupled. This follows from the problem setup (A1) and yields a system which switches coefficients depending on the current graininess. This means we can change the behavior of the system (and the region of stability) based on both our choice (or what is given to us) of A, B, and K as well as our choice of μ_i's.

We also need to modify the definition of K_i given in (2.25) so that

$$K_{i,k} := 2 \text{ Re } \Lambda_i + \mu_k \Lambda_i^* \Lambda_i. \tag{3.10}$$

Note that $K_{i,k}^{-1}$ always exists because $K_{i,k}$ is diagonal and has non-zero eigenvalues as a result of A2, just as it did in (2.25). We can now state the theorem.

Theorem 3.1 (Constrained TRoS). *Under assumptions A1-A5, given a set of matrices $\mathcal{A}_i = S^{-1} J_i S$ for $1 \leq i \leq m$ where the J_i are Jordan-epsilon form matrices with*

$J_i = \Lambda_i + N$ and S is a simultaneous similarity transform, there exists a region $\mathcal{R}_c \in \mathbb{R}^2$ consisting of pairs $\{\mu_i, \mu_j\}$ such that

$$\prod_{k=1}^{m} K_{k,i}^{-1} K_{k,j} > (I + \mu_i \Lambda_i^*)(I + \mu_i \Lambda_i), \quad \forall\, i, j = 1, \ldots, m \tag{3.11}$$

with $0 < \mu_{\min} \leq \mu_i \leq \mu_{\max}$ for all $t \in \mathbb{T}$.

A full proof of this theorem can be found in Appendix A and is very similar to the proof for Theorem 2.1.

Examples

Whereas in Theorem 2.1 we found continuous TRoS's with upper bounds defined by the equality of (2.28), in this case, continuous regions do not have any meaning, only discrete points corresponding to values for μ_i. The following figures have black (or dark) +'s where the inequality of (3.11) is satisfied (i.e. "valid" points), and red (or light) o's where it is not. Similar to the general case, the upper limits on the TRoS plots are equal to $\mu_{\max}$. Note that $\mu_{\max}$ is essentially the same as in the general case but must be calculated slightly differently because the $\mathcal{A}_i$ now vary depending on the choice of μ_i.

We use MATLAB to generate random, commuting, non-diagonal $A, B \in \mathbb{R}^{3\times3}$ pairs, and then calculate K (also in $\mathbb{R}^{3\times3}$) according to (3.9) and a user specified (but arbitrarily chosen) set of desired eigenvalues of $A + BK$. Each example has three distinct eigenvalues. The eigenvalues of A have a positive real part, while the desired (and actual) eigenvalues of $A + BK$ have a negative real part. Note that all of the following figures are most easily viewed in color.

Example 1

Figure 3.2 is a plot of the eigenvalues of A and $A + BK$ and the Hilger circle corresponding to $\mu_{\max}$ for $A + BK$. The desired eigenvalues were $[-1.9, -1.2 \pm 0.3i]$. Figures 3.3a, 3.4a, and 3.5a show that the region changes based on choices of μ_i, both the number of choices and the values of those choices. For example, $\{0.3, 0.45\}$ (approximately) is not a valid point in either of the first two plots, but is valid in the third. The corresponding figures 3.3b, 3.4b, and 3.5b demonstrate how the eigenvalues of the $\mathcal{A}_i$'s change with various choices of μ_i. The Hilger circle encompassing all eigenvalues (corresponding to $\mu_{\max}$ for the system) is also plotted in each right-hand figure.

It is important to note that valid points above the $\mu_i = \mu_j$ line allow for more flexibility in the system. These points mean that the graininess can "upshift," or increase. With no valid points above the diagonal, the system is "trapped" and cannot increase the graininess. For example, if the system described in Figure 3.3 started with $\mu_1 \approx 0.15$, it could only decrease the graininess (i.e. "downshift") or stay at approximately 0.15. If it decreased, it could never return to 0.15. The systems described by figures 3.4 and 3.5, however, have complete freedom. Intermediate steps may be required, but they can upshift or downshift to reach every allowable graininess.

Example 2

Figures 3.6-3.9 are the same as Example 1, but with a different A, B, and K. Note that the algorithm for calculating K put the eigenvalues of $A + BK$ as close to the desired values $[-1.9, -1.2 \pm 0.3i]$ as possible.

Example 3

Figures 3.10-3.13 are very similar to Examples 1 and 2, but with a different A, B, K, and desired eigenvalues $[-0.3, -0.4 \pm 0.1i]$.

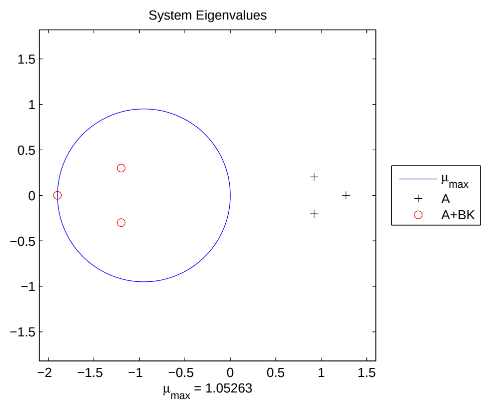
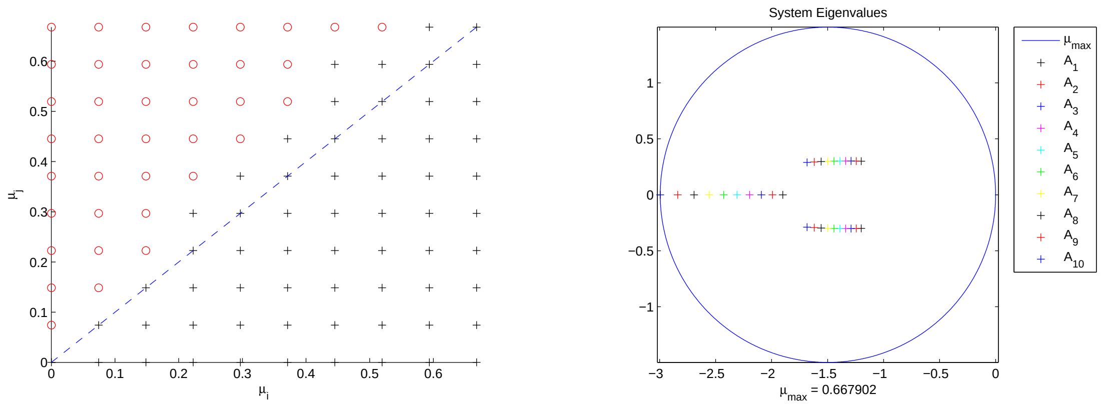
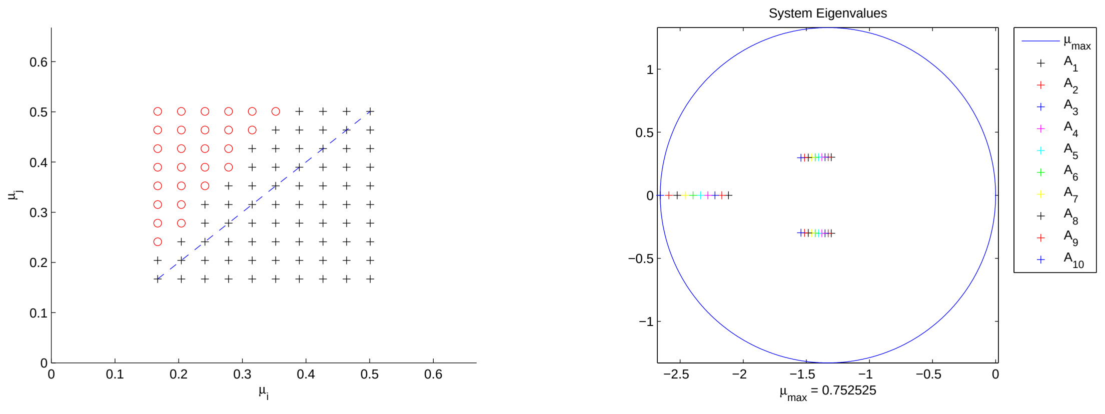


Figure 3.2. Eigenvalues of A and $A + BK$ for Example 1.



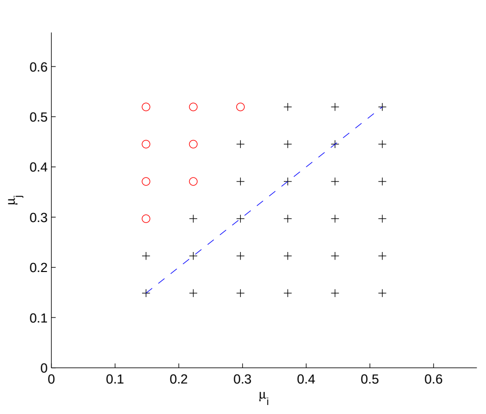
(a) (b)

Figure 3.3. $\mathcal{R}_c$ for $\mu_i < \mu_{\max}$ and $i = \{1, \ldots, 10\}$.



(a) (b)

Figure 3.4. $\mathcal{R}_c$ for $.25\mu_{max} < \mu_i \leq .75\mu_{max}$ and $i = \{1, \ldots, 10\}$.


(a)

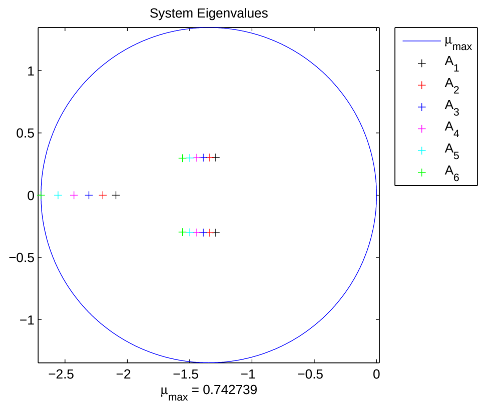

(b)

Figure 3.5. $\mathcal{R}_c$ for $\mu_i < \mu_{\max}$ and $i = \{3, \dots, 8\}$.

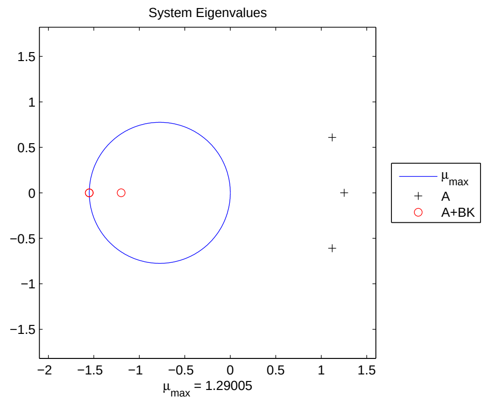
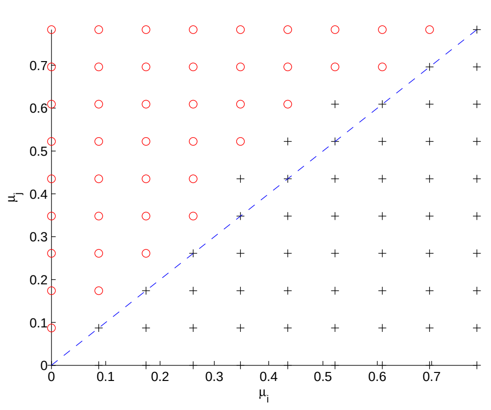


Figure 3.6. Eigenvalues of A and $A + BK$ for Example 2.


(a)

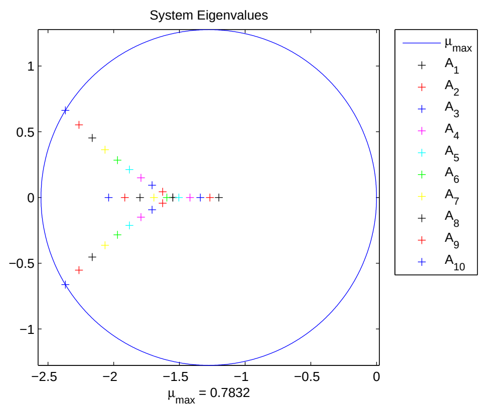

(b)

Figure 3.7. $\mathcal{R}_c$ for $\mu_i < \mu_{\max}$ and $i = \{1, \dots, 10\}$.

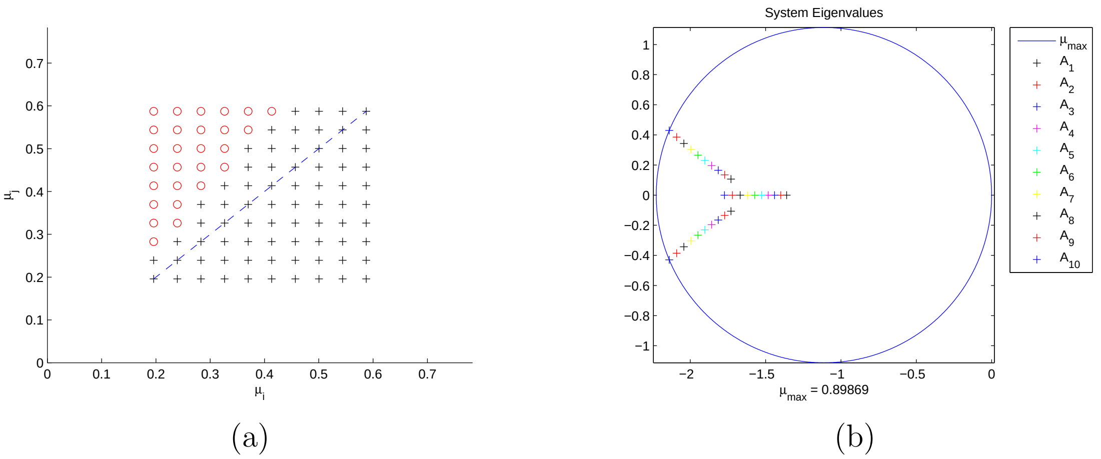


Figure 3.8. $\mathcal{R}_c$ for $.25\mu_{max} < \mu_i \leq .75\mu_{max}$ and $i = \{1, \ldots, 10\}$.

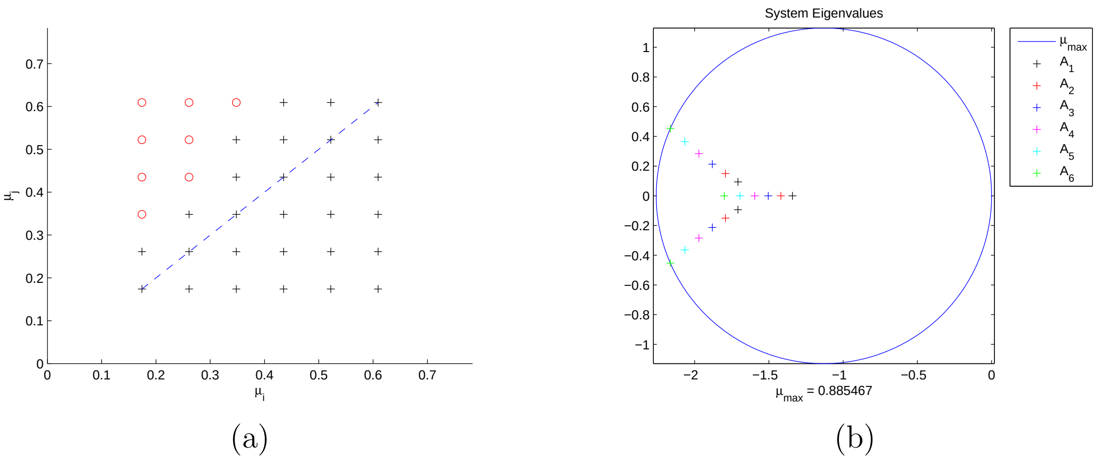


Figure 3.9. $\mathcal{R}_c$ for $\mu_i < \mu_{\max}$ and $i = \{3, \ldots, 8\}$.

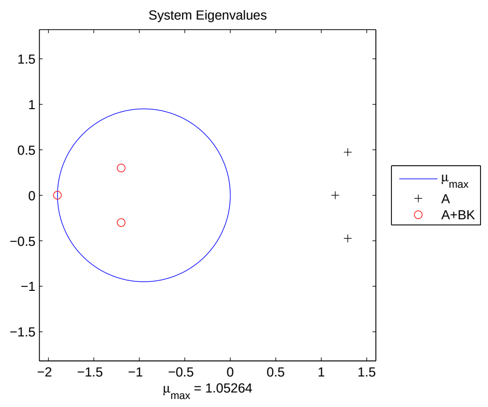


Figure 3.10. Eigenvalues of A and $A + BK$ for Example 3.

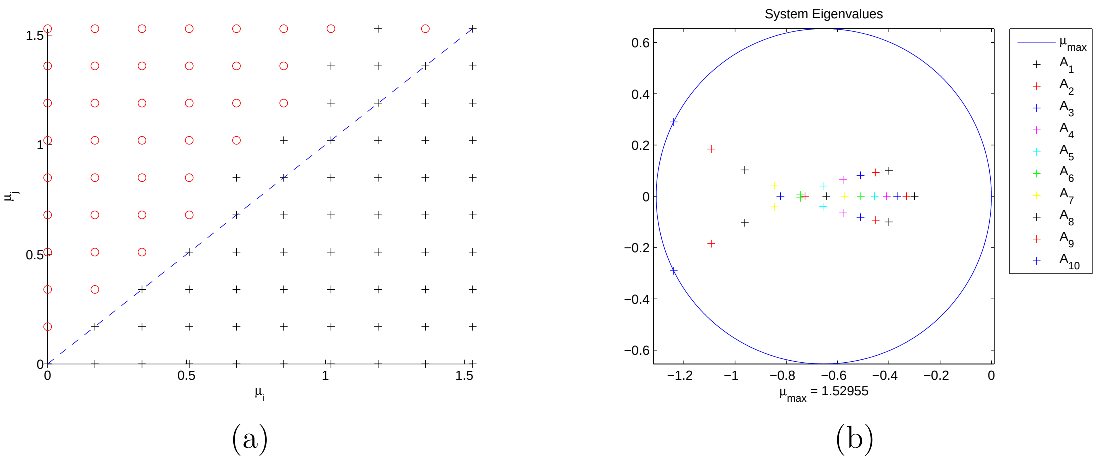


Figure 3.11. $\mathcal{R}_c$ for $\mu_i < \mu_{\max}$ and $i = \{1, \ldots, 10\}$.

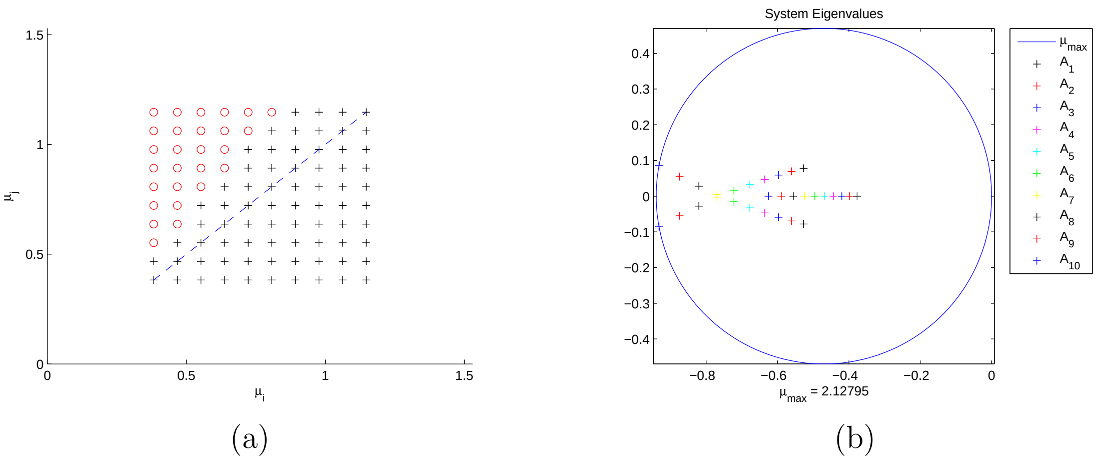


Figure 3.12. $\mathcal{R}_c$ for $.25\mu_{max} < \mu_i \leq .75\mu_{max}$ and $i = \{1, \ldots, 10\}$.

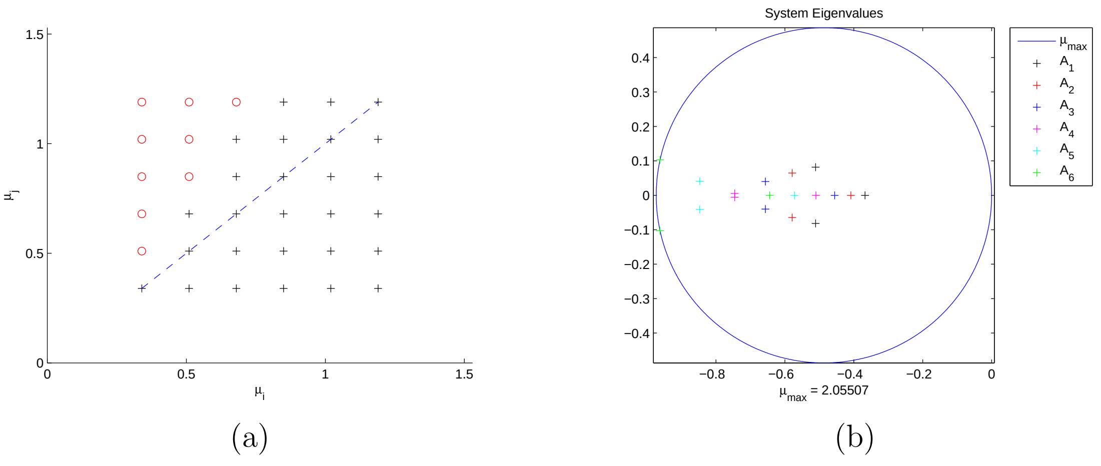


Figure 3.13. $\mathcal{R}_c$ for $\mu_i < \mu_{\max}$ and $i = \{3, \ldots, 8\}$.

CHAPTER FOUR

Conclusions

General Conclusions

The goal of this thesis was to investigate the stability of switched linear systems with non-diagonalizable system matrices on non-uniform discrete time domains. This focus arose out of a desire to study dynamic systems that operate on time domains other than $\mathbb{R}$ or $\mathbb{Z}$, such as those found in distributed control systems. What we found was that placing a condition on the time domain (i.e. time scale) itself allowed us to guarantee stability of a class of these systems, namely those with commuting system matrices. But a few things can be said about the criteria we established.

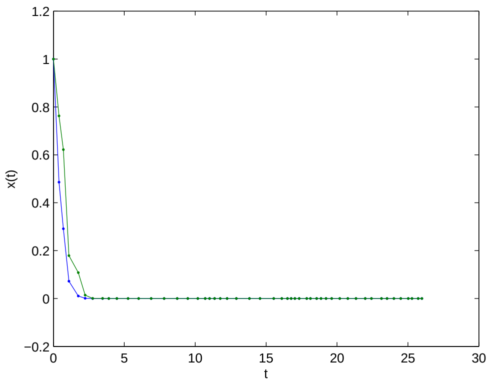
First, we wanted a non-diagonalizable solution and found one. This is good news as, often, matrices cannot be completely diagonalized. Second, we can treat a non-diagonalizable system as if it were diagonalizable from the point of view of the TRoS. This simplifies the calculations, making it easier to compute the temporal region of stability (TRoS). In particular, this might be the key to being able to implement a process based on these results in an embedded microcontroller instead of a sophisticated piece of software like MATLAB . One application of this would be allowing the microcontroller to determine when the next sampling period should occur based on the previous sampling period(s), thus, in effect, *designing* the time scale in real-time. An example of this type of *adaptive sampling* technique (based on different criteria) is presented in [14].

The third point to make is that Lyapunov methods, and thus our regions, are very conservative. While the TRoS's that we present are *sufficient*, they are by no means necessary. Figure 4.1 and Figure 4.2 both show stable systems, but the system in Figure 4.2 is operating with μ's outside the TRoS. Note that, because $\mu \leq \mu_{\max}$, we must eventually choose a point within the region, hence the marks in the lower, right-hand corner. In other words, we stay outside the region as long as possible, but are eventually forced back inside. Figure 4.3 shows that we can cause systems to be unstable, but it is significantly outside the TRoS, and in fact, outside the $\mu_{\max}$ limit for stability of each sub-system. In other words, while the mathematics allow us to *guarantee* stability within the TRoS, all *numerical* examples have thus far only required stability of each sub-system (i.e. assumption A2). (Figures 4.1-4.3 were generated in MATLAB using work done in [6]).

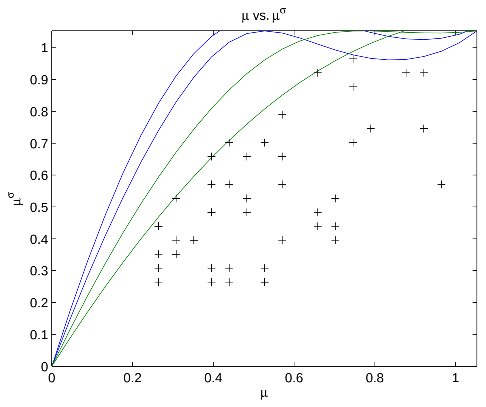
The final note about Chapter 2 we want to make is that our assumptions are fairly restrictive. Dynamical systems that can be described by switched LTI systems *with commuting matrices* form a very restrictive class of systems.

In Chapter 3, we applied the general results of Chapter 2 to a constrained switching case, where the active system matrix was dictated by the graininess. We were able to determine point-wise whether a certain switching pair (i.e. from μ_1 to μ_2) was valid or not, yielding a region of discrete points where stability was guaranteed. Just as we did above, we want to make a few notes about these findings.

First, we note that the fewer μ points (values) we use, the "better" our TRoS. By this, we mean that fewer choices of μ_i lead to a larger relative area of $\{\mu_i, \mu_j\}$ pairs which satisfy the inequality in Theorem 3.1. This makes some intuitive sense in that more points means more systems, and more systems means more conditions that have to be met in order to guarantee stability. The second item we highlight is that the TRoS also depends on the *values* of the graininess μ_i. Again, this is somewhat intuitive because the inequalities defining the TRoS take into account *all* of the systems. Since each system has different eigenvalues and the eigenvalues play a crucial role in determining stability, including systems which have "bad" eigenvalues will constrain the region more.

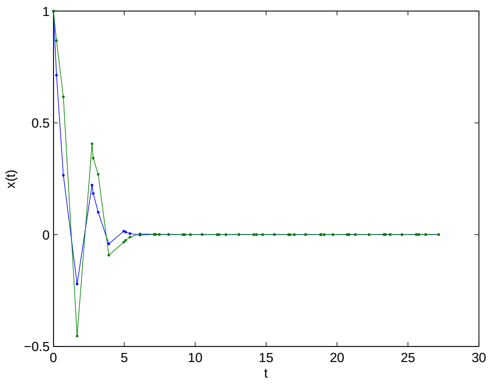


(a)

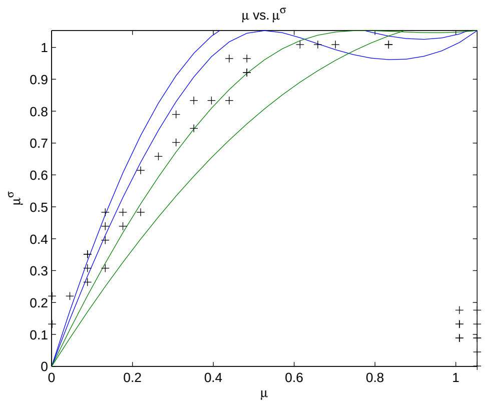


(b)

Figure 4.1. Stable system with $\{\mu, \mu^\sigma\} \in \mathcal{R}$. Each "+" represents a $\{\mu, \mu^\sigma\} \in \mathbb{T}$.

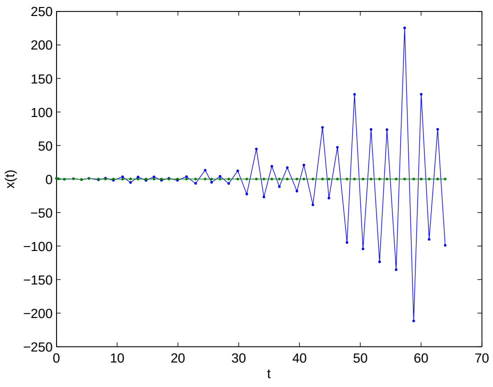


(a)

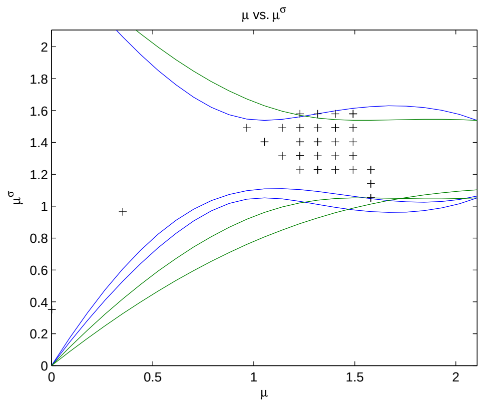


(b)

Figure 4.2. Stable system with $\mu \leq \mu_{\max}$.



(a)



(b)

Figure 4.3. Unstable system with $\mu \leq 2\mu_{\max}$.

The final point we want to make about all of these results is that they are really building blocks for further study. Because this analysis is so conservative, we need to look for stability criteria that are less restrictive. To this end, we next pose some possible avenues of future research.

Open Questions

The first question we ask is: what about other constrained switching problems? While we present results for the general case and *one* specific constrained switching problem, there are (theoretically infinitely) many such constrained problems. Finding others which manifest themselves in practice and analyzing them could prove fruitful.

Next, what happens if we allow the system to get outside of the TRoS? We saw in the previous section that these regions are quite conservative. If we do allow the system to go outside, can we say *anything* about whether it is stable or not? This was part of the premise of [14], which presents results along the lines of Chapter 3; although they are somewhat different. One item presented in [14], which was not in Chapter 3, is a metric to determine the "average" stability of the system. This yields a sub-question: what can we use as a metric for determining "average" stability?

Finally, what happens if we relax other constraints? For example, if we remove instantaneous stability of each sub-system, can we impose a switching mechanism that still maintains stability with out completely avoiding the unstable sub-systems? Likely, there are constraints that we can relax in order to better model real systems.

APPENDICES

APPENDIX A

Constrained Temporal Region of Stability

To investigate stability of (3.5), we define a Lyapunov candidate

$$V = x^T P x \tag{A.1}$$

where $P = P^T > 0$. To ensure stability, we need

$$\begin{aligned} & V^{\Delta} < 0 \\ \Rightarrow\quad & \mathcal{A}_i^T P + P\mathcal{A}_i + \mu_i \mathcal{A}_i^T P \mathcal{A}_i + (I + \mu_i \mathcal{A}_i^T) P^{\Delta} (I + \mu_i \mathcal{A}_i) < 0. \end{aligned} \tag{A.2}$$

We now set

$$\mathcal{A}_i^T P + P\mathcal{A}_i + \mu_i \mathcal{A}_i^T P \mathcal{A}_i = -M_i, \tag{A.3}$$

where $M_i = M_i^T > 0$ and solve for P. It can be shown [22] that P solves (A.3) for all i if

$$P(t) = \int_{\mathbb{S}_t} \Phi_{\mathcal{A}_i}^T(s,0) M_i(t) \Phi_{\mathcal{A}_i}(s,0) \Delta s \tag{A.4}$$

and (leaving $(s,0)$ off each $\Phi_{\mathcal{A}}$ term to shorten the equation)

$$M_i(t) = \int_{\mathbb{S}_t} \Phi_{\mathcal{A}_1}^T \dots \int_{\mathbb{S}_t} \Phi_{\mathcal{A}_j}^T \dots \int_{\mathbb{S}_t} \Phi_{\mathcal{A}_m}^T Q(t) \Phi_{\mathcal{A}_m} \Delta s_m \dots \Phi_{\mathcal{A}_j} \Delta s_j \dots \Phi_{\mathcal{A}_1} \Delta s_1, \quad \forall\ j \neq i \tag{A.5}$$

where $\mathbb{S}_t = \mu_i \mathbb{N}_0$, $Q = Q^T > 0$ is an arbitrary "seed" matrix, and $\Phi_{\mathcal{A}_k}(s,0)$ is the transition matrix that solves $y^{\Delta}(s) = \mathcal{A}_k y(s)$ with $s \in \mathbb{S}_t$ and an initial condition $y(0) = y_0$. For each μ_i, $\mathbb{S}_t$ is a constant-graininess time scale, so

$$\Phi_{\mathcal{A}_k}(s_k, 0) = e_{\mathcal{A}_k}(s_k, 0) = (I + \mu_i \mathcal{A}_k)^{\frac{s_k}{\mu_i}}. \tag{A.6}$$

Note that s is subscripted with the index k for the system it belongs to but really depends on graininess i, which might be different from k. Substituting (A.3) into (A.2) yields

$$(I + \mu_i \mathcal{A}_i^T) P^{\Delta} (I + \mu_i \mathcal{A}_i) - M_i < 0, \quad \text{for } t \in \mathbb{T} \tag{A.7}$$

where $P^\Delta = \frac{P_j - P_i}{\mu_i}$. Note that the only terms in (A.7) which depend on t are μ_i and μ_j.

Substituting (A.5) into (A.4), we get

$$\begin{aligned} P &= \int_{\mathbb{S}_t} \Phi_{A_i}^T(s_i,0) M_i \Phi_{A_i}(s_i,0)\Delta s_i \\ &= \int_{\mathbb{S}_t} \Phi_{A_i}^T(s_i,0) \left(\int_{\mathbb{S}_t} \Phi_{A_j}^T(s_j,0) \dots Q \dots \Phi_{A_j}(s_j,0)\Delta s_j \right) \Phi_{A_i}(s_i,0)\Delta s_i \\ &= \int_{\mathbb{S}_t}\int_{\mathbb{S}_t} \Phi_{A_i}^T(s_i,0)\Phi_{A_j}^T(s_j,0) \dots Q \dots \Phi_{A_j}(s_j,0)\Phi_{A_i}(s_i,0)\Delta s_j \Delta s_i. \end{aligned} \tag{A.8}$$

Since Q in (A.5) may be any arbitrary, positive definite matrix, we choose $Q = S^*S$, where S is the simultaneous Jordan epsilon similarity transform and $*$ denotes conjugate transpose. Substituting this and applying the Jordan epsilon similarity transform $\mathcal{A}_i = S^{-1}J_iS$ to (A.8) gives

$$\begin{aligned} P &= \int_{\mathbb{S}_t}\int_{\mathbb{S}_t} \left(S^*\Phi_{J_i}^*(s_i,0)\Phi_{J_j}^*(s_j,0)\dots S^{-*} \right) S^*S \left(S^{-1}\dots \Phi_{J_j}(s_j,0)\Phi_{J_i}(s_i,0)S \right) \Delta s_j \Delta s_i \\ &= S^* \left[\int_{\mathbb{S}_t} \Phi_{J_i}^*(s_i,0)\Phi_{J_i}(s_i,0)\Delta s_i \int_{\mathbb{S}_t} \Phi_{J_j}^*(s_j,0)\Phi_{J_j}(s_j,0)\Delta s_j \dots \right] S. \end{aligned} \tag{A.9}$$

For $m = 2$, we use Lemma 2.1 (2.18) and the definition of $K_{i,k}$ (3.10) to say

$$\begin{aligned} P_i &= S^* \left[\int_{\mathbb{S}_t} \Phi_{J_1}^*(s_1,0)\Phi_{J_1}(s_1,0)\Delta s_1 \int_{\mathbb{S}_t} \Phi_{J_2}^*(s_2,0)\Phi_{J_2}(s_2,0)\Delta s_2 \right] S \\ &= S^* \left[\left(-K_{1,i}^{-1} + E1_{1,i}\right)\left(-K_{2,i}^{-1} + E1_{2,i}\right) \right] S \\ &= S^* \left[K_{1,i}^{-1}K_{2,i}^{-1} + E2_i \right] S, \end{aligned} \tag{A.10}$$

where $E1_k$ and $E2_k$ are defined as in (2.17b) and (2.17c) except μ is now μ_k. Similarly

$$P_j = S^* \left[K_{1,j}^{-1}K_{2,j}^{-1} + E2_j \right] S. \tag{A.11}$$

Inserting P_i and P_j from (A.10) and (A.11) into (A.7) and eliminating S gives

$$\frac{1}{\mu_i}(I+\mu_iJ_i)^* \left[K_{1,j}^{-1}K_{2,j}^{-1} + E2_j - K_{1,i}^{-1}K_{2,i}^{-1} - E2_i \right](I+\mu_iJ_i) + K_{j,i}^{-1} - E1_{j,i} < 0. \tag{A.12}$$

We rearrange to obtain

$$\begin{aligned}
\frac{1}{\mu_i}(I+\mu_i J_i)^* \left[K_{1,j}^{-1}K_{2,j}^{-1} - K_{1,i}^{-1}K_{2,i}^{-1} + E2_j - E2_i\right](I+\mu_i J_i) &< -K_{j,i}^{-1} + E1_{j,i} \\
K_{1,j}^{-1}K_{2,j}^{-1} - K_{1,i}^{-1}K_{2,i}^{-1} + E2_j - E2_i &< \mu_i (I+\mu_i J_i)^{-*}\left(-K_{j,i}^{-1} + E1_{j,i}\right)(I+\mu_i J_i)^{-1} \\
K_{1,j}^{-1}K_{2,j}^{-1} - K_{1,i}^{-1}K_{2,i}^{-1} &< -\mu_i (I+\mu_i J_i)^{-*}(I+\mu_i J_i)^{-1}K_{j,i}^{-1} + E3_{i,i},
\end{aligned} \tag{A.13}$$

where $E3_{i,k}$ is defined as in (2.17d) except μ is now μ_k and μ^σ is μ_j. Continuing,

$$\begin{aligned}
K_{1,j}^{-1}K_{2,j}^{-1} - K_{1,i}^{-1}K_{2,i}^{-1} &< -\mu_i[(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1}K_{j,i}^{-1} + E3_{i,i} \\
K_{1,i}K_{2,i}K_{1,j}^{-1}K_{2,j}^{-1} - I &< -\mu_i[(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1}K_{i,i} + K_{i,i}E3_{i,i}K_{j,i} \\
K_{1,i}K_{2,i}K_{1,j}^{-1}K_{2,j}^{-1} &< I - \mu_i[(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1}K_{i,i} + K_{i,i}E3_{i,i}K_{j,i} \\
K_{1,i}K_{2,i}K_{1,j}^{-1}K_{2,j}^{-1} &< [(I+\mu_i J_i)(I+\mu_i J_i^*) - \mu_i K_{i,i}][(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1} \\
&\quad + K_{i,i}E3_{i,i}K_{j,i}.
\end{aligned} \tag{A.14}$$

Multiplying $(I+\mu_i J_i)(I+\mu_i J_i^*)$ out gives

$$\begin{aligned}
(I+\mu_i J_i)(I+\mu_i J_i^*) &= I + 2\mu_i \text{ Re } \Lambda_i + \mu_i^2 \Lambda_i \Lambda_i^* + \mu_i(1+\mu_i\lambda_i)N^* + \mu_i(1+\mu_i\bar{\lambda}_i)N \\
&\quad + \mu_i^2 N N^* \\
&= I + \mu_i K_{i,i} + E4_{i,i},
\end{aligned} \tag{A.15}$$

where we define $E4$ as in (2.17e) with the same exception as previously. Thus we have

$$\begin{aligned}
K_{1,i}K_{2,i}K_{1,j}^{-1}K_{2,j}^{-1} &< [I + \mu_i K_{i,i} + E4_{i,i} - \mu_i K_{i,i}][(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1} \\
&\quad + K_{i,i}E3_{i,i}K_{j,i} \\
K_{1,i}K_{2,i}K_{1,j}^{-1}K_{2,j}^{-1} &< [I + E4_{i,i}][(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1} + K_{i,i}E3_{i,i}K_{j,i} \\
K_{1,i}K_{2,i}K_{1,j}^{-1}K_{2,j}^{-1} &< [(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1} + E4_{i,i}[(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1} \\
&\quad + K_{i,i}E3_{i,i}K_{j,i}.
\end{aligned} \tag{A.16}$$

Let $E5$ be defined as in (2.17f) with the usual exception. Then

$$\begin{aligned}
K_{1,i}K_{2,i}K_{1,j}^{-1}K_{2,j}^{-1} &< [(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1} + E5_{i,i} \\
K_{1,i}^{-1}K_{2,i}^{-1}K_{1,j}K_{2,j} &> \left([(I+\mu_i J_i)(I+\mu_i J_i^*)]^{-1} + E5_{i,i}\right)^{-1} \\
K_{1,i}^{-1}K_{2,i}^{-1}K_{1,j}K_{2,j} &> \left([I+\mu_i K_{i,i} + E4_{i,i}]^{-1} + E5_{i,i}\right)^{-1} \\
K_{1,i}^{-1}K_{2,i}^{-1}K_{1,j}K_{2,j} &> \left([(I+\mu_i \Lambda_i^*)(I+\mu_i \Lambda_i) + E4_{i,i}]^{-1} + E5_{i,i}\right)^{-1}. \qquad \text{(A.17)}
\end{aligned}$$

We define a region $\mathcal{R}_{c\varepsilon}$ given by (A.17) for $i = 1, 2$. Then we note that

$$\lim_{\varepsilon\to 0} \mathcal{R}_{c\varepsilon} = \mathcal{R}_c \qquad \text{(A.18)}$$

because

$$\lim_{\varepsilon\to 0} \left([(I+\mu_i \Lambda_i^*)(I+\mu_i \Lambda_i) + E4_{i,i}]^{-1} + E5_{i,i}\right)^{-1} = (I+\mu_i \Lambda_i^*)(I+\mu_i \Lambda_i). \qquad \text{(A.19)}$$

Applying this to (A.17) yields the set of equations in Theorem 3.1 for $m = 2$:

$$K_{1,i}^{-1}K_{2,i}^{-1}K_{1,j}K_{2,j} > (I+\mu_i \Lambda_i^*)(I+\mu_i \Lambda_i) \qquad \text{for } \; i, j = 1, 2. \qquad \text{(A.20)}$$

This can be generalized for $m > 2$ to get the statement of Theorem 3.1. Thus, to guarantee stability of the switched linear system (3.1), it is sufficient to show that $\{\mu_i, \mu_j\} \in \mathcal{R}_c$, regardless of whether the system is simultaneously diagonalizable or not, which is the essence of the theorem statement.

APPENDIX B

MATLAB Function Reference

This appendix contains the documented source code for the primary MATLAB m-files used to generate examples in this thesis. The following is a list of these files, which then appear in the same order:

muregion.m Plots the general TRoS and boundary curves.

mubounds.m Calculates the boundary curves for the general TRoS.

BWmuregion.m Plots the discrete TRoS for the constrained problem.

inside_bound.m Simulates a system with choices of μ inside the TRoS.

violate_bound.m Simulates a system with choices of μ outside the TRoS.

muregion.m

```
function varargout = muregion(A,method,varargin)
%MUREGION Plots mu/mu_sigma bounds/regions.
%   MUREGION plots the analytic or numeric ("brute force") mu/mu_sigma
%   region. The analytic version computes the polynomial bounds and
%   shades the region under their minimum. The numeric version
%   approximates the region numerically.
%
%   MUREGION(A,'a') plots the analytic mu/mu_sigma region for A.
%   MUREGION(A,'n') plots the numeric mu/mu_sigma region for A.
%   MUREGION(A,'n',...) plots the numeric mu/mu_sigma region for
%       A and uses other specified parameter/value pairs (see below).
%
%   Required inputs:
%       A - a cell array of system matrices (num_sys x 1)
%       METHOD - compute region using analytic ('a') or numeric ('n')
%           method
%   Optional inputs: (specified as 'parameter',value)           {default}
%       'mu' - vector of mu/mu_sigma points to use (see muvec)   {muvec(A)}
%       'terms' - number of terms to use to approximate infinity {100}
%       'Q' - seed matrix for M (must be same size as A{1})      {V'*V}
%       'bounds' - cell array containing vectors of boundary
%           curves                                               {calc}
%       'bounds_method' - either 's' or a number of points at
%           which to evaluate mu_sigma numerically
%                                         {follows METHOD, i.e. 'a'->'s'}
%       'num_zero' - "numerical zero" offset for inequality
%           checking near 0                                      {0}
%   Optional outputs:
%       BOUNDS - cell array containing vectors of boundary curves
%           (see MUBOUNDS)
%       HBOUNDS - vector of handles to boundary curves
%       NAMES - vector of names of each boundary curve (given as
%           A{sys#},{eigenvalue#}{pos/neg} e.g. A1,2+)
%       P - the output of TSALE_SOLVE (a cell array)
%
%   See also MUREGIONGUI, MUBOUNDS, MUVEC, PLOT_MUBOUNDS, TSALE_SOLVE,
%       TIMESCALE, BWMUREGION.
%
%   Calls:      tsale_solve         muvec
%               timescale           mubounds
%               plot_mubounds
%
%   John Miller, 2/16/09


%% Check Inputs

if( isempty(A) ), error('A is empty.'), end
if( size(A{1},1) ~= size(A{1},2) ), error('A{1} is not square'), end
if( method ~= 'a' && method ~= 'n' )
    error('VER must = ''a'' or ''n''')
end
if( mod(nargin,2) ~= 0 )
    error(['Optional arguments must be supplied'...
        'in ''parameter'',value pairs.'])
end

% Determine optional inputs
for n=1:2:nargin-2
    switch varargin{n}
        case 'mu'
            mu = varargin{n+1};
        case 'terms'
            terms = varargin{n+1};
        case 'Q'
            Q = varargin{n+1};
        case 'bounds'
            bounds = varargin{n+1};
        case 'bounds_method'
```

```
            bounds_method = varargin{n+1};
        case 'num_zero'
            num_zero = varargin{n+1};
    end
end

% Set default values if not set by user
if( ~exist('mu','var') ), mu = muvec(A); end
if( ~exist('terms','var') ), terms = 100; end
if( ~exist('Q','var') )
    % Calc Q from diagonalization of A
    [S,J] = eig(A{1});
    Q = S'*S;
end
if( ~exist('bounds_method','var') )
    % Determine mubounds method from region method
    if( method == 'a' ), bounds_method = 's';
    else bounds_method = 100; end
end
if( ~exist('num_zero','var') ), num_zero = 0; end

%% Bounds
% Calculate bounds if they don't exist
if( ~exist('bounds','var') )
    [minBound, bounds] = mubounds(A,mu,'method',bounds_method);
else
    % Calculate minimum if bounds were input
    minBound = max(mu)*ones(1,length(mu));
    for n=1:numel(bounds)
        minBound = min([minBound; bounds{n}]);
    end
end

% Plot bounds
[hBounds, names] = plot_mubounds(mu,bounds);


%% Region
hold on

% Fill region based on user input (analytic or numeric)
if( method == 'n' )
    % Calc via numeric method
    P = numeric(A,mu,terms,Q,num_zero);
else
    % Fill temporal region (analytic)
    fill([mu,mu(end),mu(1)],[minBound,0,0],'r')
end
```

```
hold off

% Determine desired outputs
if nargout == 1, varargout = {bounds}; end
if nargout == 2, varargout = {bounds, hBounds}; end
if nargout == 3, varargout = {bounds, hBounds, names}; end
if nargout == 4, varargout = {bounds, hBounds, names, P}; end



%% ========================== Sub-functions ==============================

% --- NUMERIC --------------------------------------------------------------
function P = numeric(A,mu,terms,Q,num_zero)

% Get parameters
num_sys = length(A); dim = length(A{1});

% Identity matrix definition
I = eye(dim);

% Initialize mu_sigma vector
mu_sig = mu;

% Initialize arrays
M = cell(num_sys,1);
B = cell(num_sys,length(mu),length(mu_sig));
B_evmax = zeros(length(mu),length(mu_sig));
area = cell(size(B_evmax));

% Create timescale with intervals based on mu's
ts_data = zeros(1,length(mu)+1);
for t=1:length(mu)
    ts_data(t+1) = ts_data(t)+mu(t);
end
T = timescale(ts_data,'d');

% Calc M_n(mu) using integral method
% Note: A's index grabs all but nth matrix
for n=1:num_sys
    M{n} = tsale_solve( A((1:num_sys)~=n) ,Q,T,terms);
end

% Calc P(t) (for all t in T)
% Can choose whether to use P1,2,... by setting n=1,2,... in A{n} & M{n}
P = tsale_solve(A{1}, M{1}, T, terms);
```

```
% Iterate over all mu values
for m1=1:length(mu)

    % Iterate over all mu_sigma values
    for m2=1:length(mu_sig)

        % ------------------- Calc Pdel ------------------------
        % Pdel = P_sig-P / mu
        Pdel = (P{m2} - P{m1}) / mu(m1);

        % --------------------- Calc B -------------------------
        for n=1:num_sys
            B{n,m1,m2} = -M{n}{m1} + (I+mu(m1)*A{n}')*Pdel*(I+mu(m1)*A{n});
        end

        % Get max eig-val for each B
        B_evmax(m1,m2) = max( eig(B{1,m1,m2}) );
        for n=2:num_sys
            B_evmax(m1,m2) = max([eig(B{n,m1,m2}); B_evmax(m1,m2)]);
        end % for n

        % Create mu_sig array for 2D plot
        area{m1,m2} = [mu(m1) mu_sig(m2)];
    end % for m2

end % for m1

% Create 2D plot of mu vs. mu_sigma
for m=1:length(mu)
    % Invalid area (above curve)
    % plot(mu(m)*ones(1,length(mu_sig(B_evmax(m,:)>=num_zero))),...
    %     mu_sig(B_evmax(m,:)>=num_zero), 'y');
    % Valid area (under curve)
    plot(mu(m)*ones(1,length(mu_sig(B_evmax(m,:)<num_zero))),...
        mu_sig(B_evmax(m,:)<num_zero), 'r+');
end

% --------------------------------------------------------------------
```

mubounds.m

```
function varargout = mubounds(A,mu,varargin)
%MUBOUNDS Calculates mu/mu_sigma bounds.
%   MUBOUNDS(A,MU) calculates the boundary curves for the mu/mu_sigma
%   region of A using either 1) a routine to numerically find the zeros of
%   the polynomials which define the bounds or 2) MATLAB to solve the
%   polynomials symbolically. Note: the symbolic solver takes significantly
%   longer for more than about 3 systems (i.e. length(A)=3).
%
%   MUBOUNDS(A,MU,'method',POINTS) applies the numeric routine using MU and
%   a linearly spaced vector of length POINTS for mu_sigma [from min(MU) to
%   max(MU)].
%   MUBOUNDS(A,MU,'method','s') solves the polynomials symbolically using
%   MATLAB's SOLVE command.
%
%   MINBOUND = ... returns the points corresponding to the minimum over
%   all boundary curves.
%   [MINBOUND, BOUNDS] = ... returns the minimum and all the curves.
%
%   Required inputs:
%       A - a cell array of system matrices (num_sys x 1)
%       MU - vector of points at which to plot boundary curves
%   Optional input: (specified as 'parameter',value)             {default}
%       'method' - 's' = have MATLAB solve equations symbolically,
%           POINTS = solve equations numerically with specified
%           number of mu_sigma points                            {100}
%   Optional outputs: (in order)
%       MINBOUND - (1 x length(MU)) vector of points giving the minimum
%           boundary curve
%       BOUNDS - (num_sys x dim x 2) cell array of (1 x length(MU)) vectors
%           containing boundary curves
%
%   See also MUREGION, MUVEC, PLOT_MUBOUNDS, BWMUBOUNDS.
%
%   John Miller, 2/20/09

%% Check Inputs
if( isempty(A) ), error('A is empty.'), end
if( size(A{1},1) ~= size(A{1},2) ), error('A{1} is not square'), end
if( isempty(mu) ), error('MU is empty.'), end
if( ~isvector(mu) ), error('MU must be a 1 x n or n x 1 vector.'), end

% Turn off "divide by zero" warning (and save the previous state)
%   Note: 'eval(left_term)' issues this warning, but we handle it with
%   'if isfinite(val)'
div0_state = warning('off','MATLAB:divideByZero');

% Assign default values
method = 1; points = 100;

% Determine optional inputs
for n=1:2:nargin-2
    switch varargin{n}
        case 'method'
            if varargin{n+1} == 's'
```

```
                method = 2;
            elseif( isnumeric(varargin{n+1}) && varargin{n+1} > 0 )
                points = varargin{n+1};
            else
                warning('mubounds:input',...
                    'Optional METHOD input is incorrect, using default.')
            end
    end
end

%% Calc
% Get parameters
num_sys = length(A); dim = length(A{1});

% Pre-allocate memory
S = cell(num_sys,1);
J = cell(num_sys,1);
e = zeros(num_sys,dim);
bounds = cell(num_sys,dim,method);
minBound = max(mu)*ones(1,length(mu));

% Create symbolic equation terms
top = '('; bot = '(';
top_term = '(2*real(e(%d,k)) + x*abs(e(%d,k))^2)';
bot_term = '(2*real(e(%d,k)) + mu(m)*abs(e(%d,k))^2)';

for n=1:num_sys
    % Diagonalize A's
    [S{n},J{n}] = eig(A{n});

    % Get eigenvalues
    e(n,:) = diag(J{n});

    % Setup left side of symbolic equation
    if( n==1 )
        top = [top sprintf( top_term ,n,n)];
        bot = [bot sprintf( bot_term ,n,n)];
    else
        % (multiply added to front of string)
        top = [top sprintf( ['.*' top_term] ,n,n)];
        bot = [bot sprintf( ['.*' bot_term] ,n,n)];
    end
end

% Create left & right terms of symbolic equation
left_term = [top ')./' bot ')'];
right_term = '((1+mu(m)*conj(e(n,k)))*(1+mu(m)*e(n,k)))';
```

```
% Determine method to use (2=symbolic or 1=numeric)
if( method == 2 )
    % Create symbolic equation (remove periods from left side)
    s = [strrep(left_term,'.','') '=' right_term];

    % Solve symbolically
    % Note: returns two solutions (corresponding to +/-)
    Sol = solve(s,'x');

    % Evaluate & plot bounds for each mu & eigenvalue
    for n=1:num_sys
        for k=1:dim
            % Specify bounds to contain row vectors
            bounds{n,k,1} = zeros(1,length(mu));
            bounds{n,k,2} = bounds{n,k,1};

            % Evaluate symbolic equation for each mu
            for m=1:length(mu)
                bounds{n,k,1}(m) = eval(Sol(end-1));
                bounds{n,k,2}(m) = eval(Sol(end));
            end

            % Determine minimum bound under curves
            minBound = min([minBound; bounds{n,k,1}; bounds{n,k,2}]);
        end
    end

else % Non-symbolic (i.e. numeric)

    % Create mu_sigma vector
    x = linspace(min(mu),max(mu),points);

    % Iterate over all mu/mu_sig/n combinations
    for m=1:length(mu)
        for k=1:dim
            % Evaluate left side for current m & k and all of x (mu_sig)
            left = eval(left_term);

            for n=1:num_sys
                % Evaluate right side for current m, n, & k
                right = eval(right_term);

                % Specify bounds to contain row vectors
                if( m==1 ), bounds{n,k} = zeros(1,length(mu)); end

                % Boundary point corresponds to min over all mu_sig points
                [val idx] = min(abs(left-right));
                % Get current boundary point and avoid Inf values
                if( isfinite(val) )
                    bounds{n,k}(m) = x(idx);
                else
                    bounds{n,k}(m) = max(mu);
                end

                % Determine minimum bound under curves
                if( m==length(mu) )
                    minBound = min([minBound; bounds{n,k}]);
                end
            end
        end
    end

end

% Restore "divide by zero" warning back to it's original state
warning(div0_state)

% Assign outputs if requested
if nargout == 1, varargout = {minBound}; end
if nargout == 2, varargout = {minBound, bounds}; end
```

BWmuregion.m

```
function varargout = BWmuregion(ABK,mu,mu_sig,varargin)
%BWMUREGION Plots mu/mu_sigma bounds/region of the bandwidth problem.
%   BWMUREGION plots a numeric version of the mu/mu_sigma region for the
%   bandwidth problem using the bounds inequalities. The BW problem
%   consists of a closed-loop feedback system, which is switched according
%   to the choice of mu. Thus, for every value of mu (and mu_sigma) there
%   is a new "system" (i.e. system matrix, script A).
%
%   BWMUREGION(ABK,...) plots the numeric mu/mu_sigma region for
%       ABK and uses other specified parameter/value pairs (see below).
%
%   Required inputs:
%       ABK - a (2x1) cell array of the A & A+BK system matrices, where
%           ABK{1} = A and ABK{2} = A+BK
%       MU - vector of points to use for mu (see MUVEC)
%       MU_SIG - vector of points to use for mu_sigma (see MUVEC)
%   Optional inputs: (specified as 'parameter',value)             {default}
%       'PlotBounds' - 1=plot boundary lines; 0=don't             {0}
%   Optional outputs:
%       BOUNDS - cell array containing vectors of boundary curves
%       HBOUNDS - vector of handles to boundary curves
%       NAMES - vector of names of each boundary curve (given as
%           A{sys#},{eigenvalue#}{pos/neg} e.g. A1,2+)
%       AREA - (length(MU) x length(MU_SIG) x dim) matrix of the
%           difference between the LHS and RHS of the bounds equation
%           over the mu/mu_sigma region
%
%   See also BWMUBOUNDS, BWMUVEC, PLOT_MUBOUNDS, MUREGION.
%
%   Calls:     BWmubounds              plot_mubounds
%
%   John Miller, 3/5/09


%% Check Inputs

if( isempty(ABK) ), error('ABK is empty.'), end
if( size(ABK{1},1) ~= size(ABK{1},2) ), error('ABK{1} is not square.'), end
if( size(ABK{1},1) ~= size(ABK{2},1) )
    error('ABK{1} & ABK{2} are not the same size.')
end
if( mod(nargin-1,2) ~= 0 )
    error(['Optional arguments must be supplied'...
        'in ''parameter'',value pairs.'])
end

% Check for A + BK stable
if( get_mu_max(ABK{2}) < 0 )
    error('A+BK is not stable on R (i.e. mu_max_ABK < 0)')
end
% Check that A & A+BK commute
temp = norm(ABK{1}*ABK{2}-ABK{2}*ABK{1});
if( temp > 1e-8 )
    error('A & A+BK do not commute, ||diff|| = %g',temp)
end

% Check that mu & mu_sig are vectors
if( ~isvector(mu) ), error('MU must be a vector.'), end
if( ~isvector(mu_sig) ), error('MU_SIG must be a vector.'), end

% Determine optional inputs
for n=1:2:nargin-4
    switch varargin{n}
        case 'PlotBounds'
            plotBounds = varargin{n+1};
    end
end
```

```
%% Bounds

% Calculate bounds & area
[minBound, bounds, area] = BWmubounds(ABK,mu,mu_sig);

% Form names
names = cell(numel(bounds),1);
for n=1:numel(bounds)
    names{n} = sprintf('k=%d',n);
end

% Check if user wants to plot bounds
if( exist('plotBounds','var') && plotBounds == 1 )
    % Plot bounds
    [hBounds, names] = plot_mubounds(mu,bounds,names);
else
    % Create empty variables to return
    hBounds = []; names = [];
end


%% Region

% Fill the region by plotting idividual points
hold on
for m1=1:length(mu)
    for m2=1:length(mu_sig)
        % Check that bounds ineq is satisfied for both eigenvalues
        if( sum(area(m1,m2,:)) >= 0 )
            S = 'k+';
        else S = 'ro';
        end
        % Plot the point
        plot(mu(m1),mu_sig(m2),S);
    end
end
hold off
xlabel('\mu_i'), ylabel('\mu_j')

% Plot 3D region
% figure(gcf+1)
% h = surf(mu, mu_sig, log10(area(:,:,1)'));
% set(h,'EdgeAlpha',.05)
% %caxis([min(min(area(:,:,1))) 5])
% caxis([-5 5])
% %zlim([-5 5])
% view(0,90)
% figure(gcf+1)
% h = surf(mu, mu_sig, log10(area(:,:,2)'));
% set(h,'EdgeAlpha',.05)
% %caxis([min(min(area(:,:,2))) 5])
% caxis([-5 5])
% %zlim([-5 5])
% view(0,90)

% Determine desired outputs
if nargout == 1, varargout = {bounds}; end
if nargout == 2, varargout = {bounds, hBounds}; end
if nargout == 3, varargout = {bounds, hBounds, names}; end
if nargout == 4, varargout = {bounds, hBounds, names, area}; end
```

inside_bound.m

```
function varargout = inside_bound(A,mu,varargin)
%INSIDE_BOUND Simulates a system with points under the mu/mu_sigma bound.
%   [X,T]=INSIDE_BOUND(A,MU) simulates a randomly switched system with
%   system matrices A and vector of availble mu choices MU and returns X,
%   the state response of the system, and T, the time vector over which X
%   is calculated. Mu points are chosen randomly below the mu/mu_sigma
%   bound.
%
%   INSIDE_BOUND(A,MU,CHOICES) allows the variable CHOICES to determine
%   the number of choices of mu_sigma values at each mu point.
%
%   [X,T,T_MU,T_MU_SIG] = ... returns the points chosen for mu
%   and mu_sigma (similar to TSMU, but not a timescale object).
%   [...,MU_RESETS] = ... returns the indices of where mu_sigma is chosen
%   small because it must be (i.e. has to be less than mu_max).
%
%   Required inputs:
%       A - a cell array of system matrices (num_sys x 1)
%       MU - vector of points at which to plot boundary curves
%           (see MUVEC)
%   Optional inputs:                                          {default}
%       MU_MIN - minimum value mu is allowed to take          {.2*mu_max}
%   Optional outputs:
%       X - system response
%       T - time vector
%       T_MU - vector of mu points chosen
%       T_MU_SIG - vector of mu_sigma points chosen
%
%   See also VIOLATE_BOUND, MUBOUNDS, MUVEC.
%
%   Calls:     mubounds         get_system_response
%
```

```
%   John Miller, 5/15/09

%% Check inputs
if( isempty(A) ), error('A is empty.'), end
if( size(A{1},1) ~= size(A{1},2) ), error('A{1} is not square'), end
if( isempty(mu) ), error('MU is empty.'), end
if( ~isvector(mu) ), error('MU must be a 1 x n or n x 1 vector.'), end

% Assign default values
mu_min = .2*max(mu);

% Determine if optional input is valid
if( nargin > 2 )
    if( varargin{1} > 0 )
        mu_min = varargin{1};
    else
        warning('inside_bound:input',...
            'Optional input is incorrect, using default.')
    end
end

%% Setup
t = zeros(1,50);                % Init time vector
t_mu = zeros(length(t)-1,1);    % Init chosen mu points vector
t_mu_sig = t_mu;                % Init chosen mu_sigma points vector
mu_sig = mu(10);                % Init mu_sigma
idx = 10;                       % Init mu_sigma index
mu_resets = 1;                  % Init vector of points where mu is reset

%% Calc
% Calc mu/mu_sigma bound
minbound = mubounds(A,mu);

% Choose mu values that don't violate bound & build time vector
for i=1:length(t)-1
    % mu_sig is now the current mu value
    t_mu(i) = mu_sig;
    t(i+1) = t(i)+mu_sig;

    % Find the limits of allowable mu_sig choices (above mu_min, below
    % the bound)
    high = find(mu < minbound(idx),1,'last');
    low = find(mu > mu_min,1);

    % Check for no allowable points
    if( (high-low) <= 0 )
        error('No allowable points were found: i=%d, high=%d, low=%d',...
            i,high,low)
    else
        % Choose a random mu_sigma value below the bound
        idx = ceil(low+(high-low)*rand);
        mu_sig = mu(idx);
    end

    % Save current mu_sigma point
    t_mu_sig(i) = mu_sig;
end

% Calc switched system response
x = get_system_response(A, t_mu, 3);

% Determine desired outputs
if nargout == 2, varargout = {x, t}; end
if nargout == 3, varargout = {x, t, t_mu}; end
if nargout == 4, varargout = {x, t, t_mu, t_mu_sig}; end
```

violate_bound.m

```
function varargout = violate_bound(A,mu,varargin)
%VIOLATE_BOUND Simulates a system with points above the mu/mu_sigma bound.
%   [X,T]=VIOLATE_BOUND(A,MU) simulates a randomly switched system with
%   system matrices A and vector of availble mu choices MU and returns X,
%   the state response of the system, and T, the time vector over which X
%   is calculated. Mu points are semi-randomly chosen above the minimum
%   mu/mu_sigma bound.
%
%   VIOLATE_BOUND(A,MU,CHOICES) allows the variable CHOICES to determine
%   the number of choices of mu_sigma values above the bound.
%
%   [X,T,T_MU,T_MU_SIG] = ... returns the points chosen for mu
%   and mu_sigma (similar to TSMU, but not a timescale object).
%   [...,MU_RESETS] = ... returns the indices of where mu_sigma is chosen
%   small because it must be (i.e. has to be less than mu_max).
%
%   Required inputs:
%        A - a cell array of system matrices (num_sys x 1)
%        MU - vector of points at which to plot boundary curves
%           (see MUVEC)
%   Optional inputs:                                            {default}
%       CHOICES - number of mu_sigma value choices above the    {3}
%            bound
%   Optional outputs:
%       X - system response
%       T - time vector
%       T_MU - vector of mu points chosen
%       T_MU_SIG - vector of mu_sigma points chosen
%       MU_RESETS - indices of T_MU where mu_sigma is small
%
%   See also INSIDE_BOUND, MUBOUNDS, MUVEC.
%
%   Calls:     mubounds         get_system_response
%
%   John Miller, 5/15/09

%% Check inputs
if( isempty(A) ), error('A is empty.'), end
if( size(A{1},1) ~= size(A{1},2) ), error('A{1} is not square'), end
if( isempty(mu) ), error('MU is empty.'), end
if( ~isvector(mu) ), error('MU must be a 1 x n or n x 1 vector.'), end

% Assign default values
choices = 3;

% Determine if optional input is valid
if( nargin > 2 )
    if( varargin{1} > 0 )
        choices = varargin{1};
    else
        warning('violate_bound:input',...
            'Optional input is incorrect, using default.')
    end
end

%% Setup
t = zeros(1,50);                % Init time vector
t_mu = zeros(length(t)-1,1);    % Init chosen mu points vector
t_mu_sig = t_mu;                % Init chosen mu_sigma points vector
mu_sig = mu(1);                 % Init mu_sigma
idx = 1;                        % Init mu_sigma index
mu_resets = 1;                  % Init vector of points where mu is reset

%% Calc
% Calc mu/mu_sigma bound
minbound = mubounds(A,mu);

% Choose mu values that violate bound & build time vector
for i=1:length(t)-1
    % mu_sig is now the current mu value
    t_mu(i) = mu_sig;
    t(i+1) = t(i)+mu_sig;

    % Find the next mu_sig value just above the bound
    idx = find(mu > minbound(idx),1);
    % Check if we are at mu_max
    if( isempty(idx) || mu(idx) == max(mu) )
        % Randomly choose 1 of the 1st CHOICES mu values
        idx = ceil(choices*rand);
        mu_sig = mu(idx);
        % Add next point's index to list of where mu is reset
        mu_resets = [mu_resets i];
    else
        % Set the allowable choices based on how close we are to mu_max
        if idx > find(mu==max(mu))-choices
            % Choose one of available values between current and mu_max
            idx = ceil(idx+(length(mu)-idx)*rand);
        else
            % Choose one of CHOICES values above the bound
            idx = ceil(idx+choices*rand);
        end
        % Get the mu_sigma value chosen
        mu_sig = mu(idx);
    end
    % Save current mu_sigma point
    t_mu_sig(i) = mu_sig;
end

% Calc switched system response
x = get_system_response(A, t_mu, 3);

% Determine desired outputs
if nargout == 2, varargout = {x, t}; end
if nargout == 3, varargout = {x, t, t_mu}; end
if nargout == 4, varargout = {x, t, t_mu, t_mu_sig}; end
if nargout == 5, varargout = {x, t, t_mu, t_mu_sig, mu_resets}; end
```

BIBLIOGRAPHY

[1] B. Allen, “Experimental investigation of a time scales linear feedback control theorem,” Master’s thesis, Baylor University, 2007.

[2] R. Bishop and R. Dorf, *Modern Control Systems*, 9th ed. Upper Saddle River, NJ: Prentice-Hall, 2001.

[3] M. Bohner and A. Peterson, *Dynamic Equations On Time Scales: An Introduction With Applications.* Boston, MA: Birkhäuser, 2001.

[4] M. Bohner and A. Peterson, Eds., *Advances in Dynamic Equations On Time Scales.* Boston, MA: Birkhäuser, 2003.

[5] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. New York, NY: Oxford University Press, 1999.

[6] D. Cranor, “Testing the mu-sigma boundary for randomly switched systems over discrete time scales,” 2008, result of independent undergraduate research study.

[7] J. DaCuhna, “Lyapunov stability and floquet theory for nonautonomous linear dynamic systems on time scales,” Ph.D. dissertation, Baylor University, 2004.

[8] J. DaCunha, “Stability for time varying linear dynamic systems on time scales,” *J. Computational and Applied Mathematics*, vol. 176, no. 2, pp. 381–410, Apr. 2005.

[9] ——, “Transition matrix and generalized matrix exponential via the Peano-Baker series,” *J. Difference Equations and Applications*, pp. 1245–1264, 2005.

[10] ——, “Instability results for slowly time varying linear dynamic systems on time scales,” *J. Mathematical Analysis and Applications*, pp. 1278–1289, April 2007.

[11] J. Davis, private communication.

[12] J. Davis, J. Miller, I. Gravagne, R. Marks II, and A. Ramos, “Stabilty of switched linear systems on non-uniform time domains,” 2008, submitted to *IEEE Trans. on Systems, Man, and Cybernetics - Part B.*

[13] I. Gravagne, J. Davis, and J. DaCuhna, “A unified approach to high-gain adaptive controllers,” 2009, submitted to *SIAM J. Control and Optimization.*

[14] I. Gravagne, J. Davis, J. DaCunha, and R. Marks II, “Bandwidth reduction for controller area networks using adaptive sampling,” in *Proc. Int. Conf. Robotics and Automation*, New Orleans, LA, Apr. 2004, pp. 5250–5255.

[15] S. Hilger, "Ein maßkettenkalkül mit anwendung auf zentrumsmannigfaltigkeiten," Ph.D. dissertation, Universität Würzburg, 1988.

[16] M. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra.* New York, NY: Academic Press, 1974.

[17] R. Horn and C. Johnson, *Matrix Analysis.* Cambridge University Press, 1996.

[18] B. Jackson, "A general linear systems theory on time scales: Transforms, stability, and control," Ph.D. dissertation, Baylor University, 2007.

[19] W. Levine, Ed., *The Control Handbook.* Boca Raton, FL: CRC Press, Inc., 1996.

[20] D. Liberzon, *Switching in Systems and Control.* Boston, MA: Birkhäuser, 2003.

[21] A. Lyapunov, "The general problem of the stability of motion," *Int. J. Control*, vol. 55, pp. 521–790, 1992.

[22] A. Ramos, "Stability of hybrid dynamical systems: Analysis and design," Ph.D. dissertation, Baylor University, Aug. 2009.

[23] G. Strang, *Introduction to Applied Mathematics.* Wellesley, MA: Wellesley-Cambridge Press, 1986.