The Dissertation Committee for Andrew J. Zelenak Certifies that this is the approved version of the following dissertation:

Nonlinear Control With Two Complementary Lyapunov Functions

Committee:

Sheldon Landsberger, Supervisor

Mitchell Pryor

Ashish Deshpande

Benito Fernàndez

Doug Kautz

Nonlinear Control With Two Complementary Lyapunov Functions

by

Andrew J. Zelenak, B.S.E.; M.S.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2016

Dedication

To my family: Steve, Amy, Georgianna, and Andrea.

Acknowledgements

I am particularly grateful to:

- My advisors, Dr. Mitch Pryor and Dr. Sheldon Landsberger, for numerous critiques, insights, and support
- The other members of my committee:
    - Dr. Benito Fernàndez, for a detailed, technical review of my algorithm and several valuable suggestions
    - Doug Kautz, for his support and expertise while at Los Alamos National Laboratory
    - Dr. Ashish Deshpande, for his critique and support
- Dr. Maruthi Akella, for two insights and his referral to several publications
- The DET division of Los Alamos National Laboratory, for its support
- The students of the Nuclear Robotics Group for many insights and suggestions

Nonlinear Control With Two Complementary Lyapunov Functions

Andrew J. Zelenak, Ph.D.

The University of Texas at Austin, 2016

Supervisor: Sheldon Landsberger

If a Lyapunov function is known, a dynamic system can be stabilized. However, computing or selecting a Lyapunov function is often challenging. This dissertation presents a new approach which eliminates this challenge: a simple control Lyapunov function [CLF] is assumed then the algorithm seeks to reduce the value of the Lyapunov function. If the control effort would have no effect at any iteration, the CLF is switched in an attempt to regain control. There is some flexibility in choosing these two complementary CLF's but they must satisfy a few characteristics. The method is proven to asymptotically stabilize a wide range of nonlinear systems and was tested on an even broader variety in simulation. It was also tested on an industrial robot to provide compliant behavior. The simulated and hardware demonstrations provide a broad perspective on the algorithm's usefulness and limitations. In comparison to the ubiquitous PID controller, the algorithm's advantages include enhanced performance, ease of tuning, and extensions to higher-order and/or coupled systems. Those claimed advantages are validated by a test with four engineering students, which validates the controller as a viable option for nonlinear control (even at the undergraduate level). The algorithm's drawbacks include the necessity of a dynamic model and, when linearization is required, the reliance on a small simulation time step; however, for the motivating application –interactive industrial robotic systems – both requirements

were already met. Finally, the developed software was released to the public as part of the Robot Operating System (ROS) and the details of that release are included in this report.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 The need for an updated controls curriculum

This dissertation was motivated, in part, by a perceived shortfall in the undergraduate controls curriculum. In 2015, the vast majority of undergraduate engineering programs in the United States are accredited by the non-profit, non-governmental organization known as ABET (Accreditation Board for Engineering and Technology). This accreditation process ensures that each program meets minimum quality standards and that graduates "are ready to enter their professions." [Pradhan, 2012] Part of the accreditation checklist for most disciplines is a baseline competence in control theory, which is the science of manipulating or regulating a dynamic system. Control theory is applied in nearly every field in engineering, from optimizing chemical processes to performing complex motions with aircraft.

ABET's requirements related to control theory is that students must use "state-of-the-art technology." [Felder, 2003] However, the irony is that most institutions – at the undergraduate level - choose to focus on the control of Linear Time Invariant (LTI) dynamic systems, which are a small segment of all possible systems. It is possible that engineering schools choose to focus on linear systems because they may be simpler to understand than a general, nonlinear system. Perhaps the schools feel that knowledge of how to control a linear system is sufficient since some nonlinear systems can be linearized and controlled (at least locally) by linear techniques such as pole placement. Perhaps schools acknowledge the importance of nonlinear controls, but a deeper examination of the topic is not possible due to time constraints and other priorities.

A list of controls-related topics studied by undergraduate mechanical engineers at the Massachusetts Institute of Technology (MIT) ["MIT Subject Listing"] underscores the point that nonlinear control theory is mostly neglected in engineering curricula. The list of topics in the two required controls classes includes:

- Linearization of equations of motion
- Matrix eigenvalue problems
- Linear systems theory
- Linear algebra
- Laplace transform
- Transfer functions, time response and frequency response, poles and zeros
- PID compensation
- Root-locus design concepts

Notice that the only technique on the list which applies to nonlinear systems is linearization. In other words, an MIT undergraduate's only option for controlling a nonlinear system is to linearize and hope it behaves well enough for pole placement, PID control, root-locus, or one of the other linear techniques to work. Commonly, the linearization will only be accurate locally and it will certainly not yield ideal performance.

The potential benefits of nonlinear control over linear control depend on the specifics of any given situation, but they may include:

- The asymptotic stabilization of systems which would be uncontrollable by linear techniques. For example, [Gustafsson, 1995] discusses why a nonlinear term must be included to minimize the swaying of a crane's load

during motion. The equations of motion for the crane are coupled, which is a common limiter for linear controllers.

- The ability to tune the controller for arbitrarily optimal performance at the expense of higher computational effort. [Di Palma, 2005]
- The ability to stabilize time-varying systems such as the dulling of a cutter during a machining operation.
- Improved robustness. [Coleman, 1994]

Coleman's quantitative comparison between a linear PID controller and two nonlinear techniques was based on a third-order motor speed control problem. The PID controller performed worse on step response tests for both the nominal plant and a perturbed plant (Figure 1).

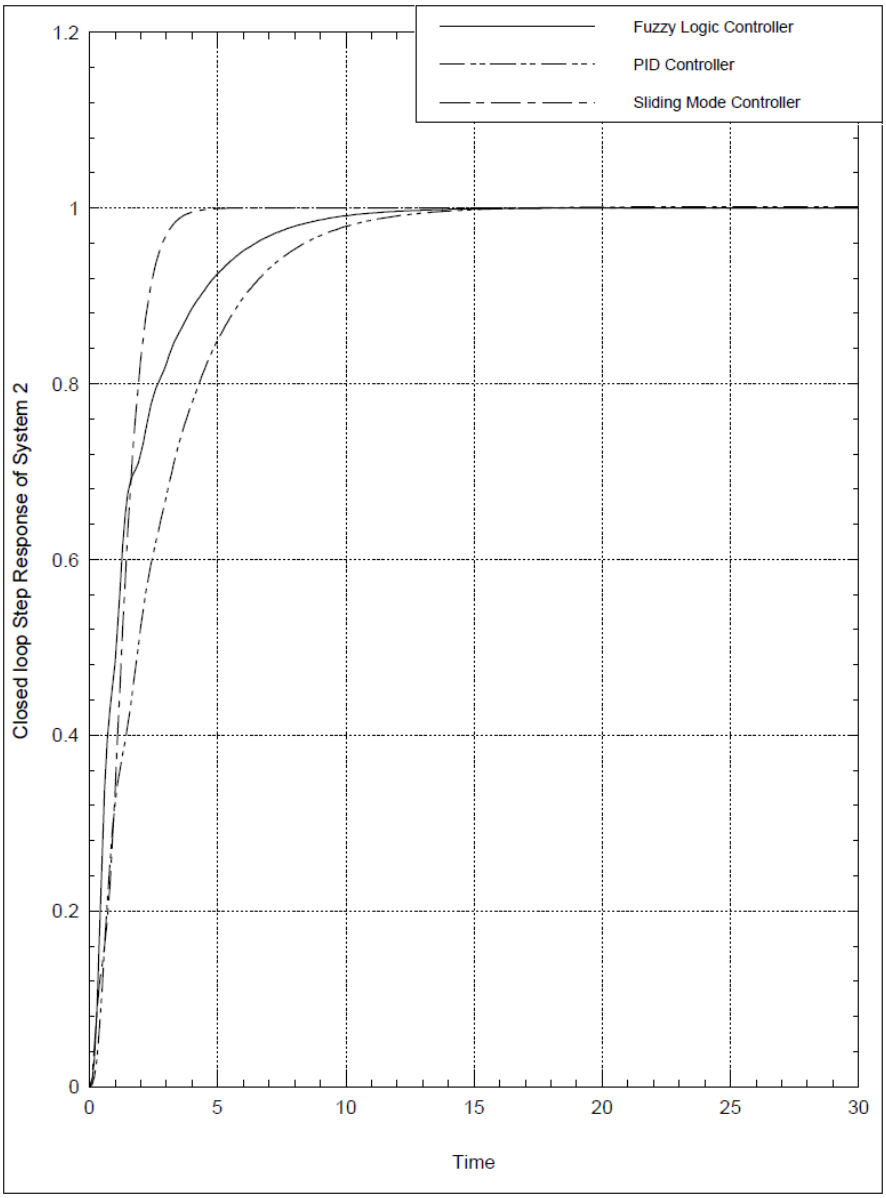


Figure 1. Robustness comparison: Note that the PID controller is slower to stabilize this perturbed system than the nonlinear control techniques.

Furthermore, although linear systems are generally regarded as simpler, the earliest research into control theory actually focused on nonlinear systems. Aleksandr Lyapunov's Second Method, which describes an intuitive basis for understanding the stability of all dynamic systems, was published in 1892. The linear PID controller, which is nearly

ubiquitous today, was not developed until three decades later [Minorsky, 1922]. The author's personal experience and observations suggest that many of the tools developed specifically for the control of linear systems (frequency analysis, Nyquist stability criterion, etc.) are non-intuitive and may be unnecessarily confusing for undergraduate students. It may be more useful and educational to include the study of nonlinear systems earlier in the curriculum. Once nonlinear control theory is understood, the advanced control topics related to linear systems may also seem simpler. For example, the linear controllability matrix was not well-explained in the author's controls classes, but it derives from the nonlinear controllability matrix where the terms have straight-forward interpretations as an "open-loop drift" vector and a closed-loop "controlled" vector (as explained in Section 2.3).

Since undergraduate engineering students are taught almost exclusively linear control theory and PID controller design (and have been for decades), it is no surprise that PID controllers are **by far** the most common type of controller in industry, comprising 97% of all applications [Desborough, 2000]. Of the 64 respondents from a survey of industry engineers [Cook, 2009], not a single one rated knowledge of PID control as "not required" for entry-level engineers. PID controllers are also popular since, given their ubiquitous nature, there are many software tools (such as MATLAB's sisotool, etc.) that simplify PID tuning. Cominos states that the major benefits of a PID controller are simplicity, robustness, a wide range of applicability, and near-optimal performance in some cases [Cominos, 2002]. Astrom [2002] and the National Instruments Corporation [2011] concur that simplicity is a major factor that has contributed to the popularity of PID control.

Clearly PID controllers are a powerful tool and they are relatively easy to use, but the PID controller has severe drawbacks. It is largely applicable only to the simplest type of system (Single-Input-Single-Output) [Garcia, 2005] and it suffers from performance limitations. Atherton [1999] and Sung [1996] list some of the well-known performance limitations of PID control, which may include (depending on the specific system)

- overshoot and/or slow rise time,
- long settling time,
- derivative kick,
- difficulty controlling integrating, resonant and unstable processes,
- difficulty controlling processes with large time delays,
- and integral windup.

The presented nonlinear controller does not solve all of these problems, but it can improve performance significantly for first-order systems while being extensible to much larger and more complex systems. Figure 2 is a performance comparison with several PI and PD controllers on a first-order linear system. The details of this illustrative example are presented in Section 4.3. Notice how the nonlinear controller does not overshoot the setpoint and the 0 to 100% rise time is faster by 30%. Additionally, the nonlinear controller achieves the faster rise time with a less extreme control effort. Increasing the proportional term to an even greater value (which is not a recommended technique) would not improve the PI/PD controller performance significantly as most of the delay occurs at the “knee” where the error measurement is very small.

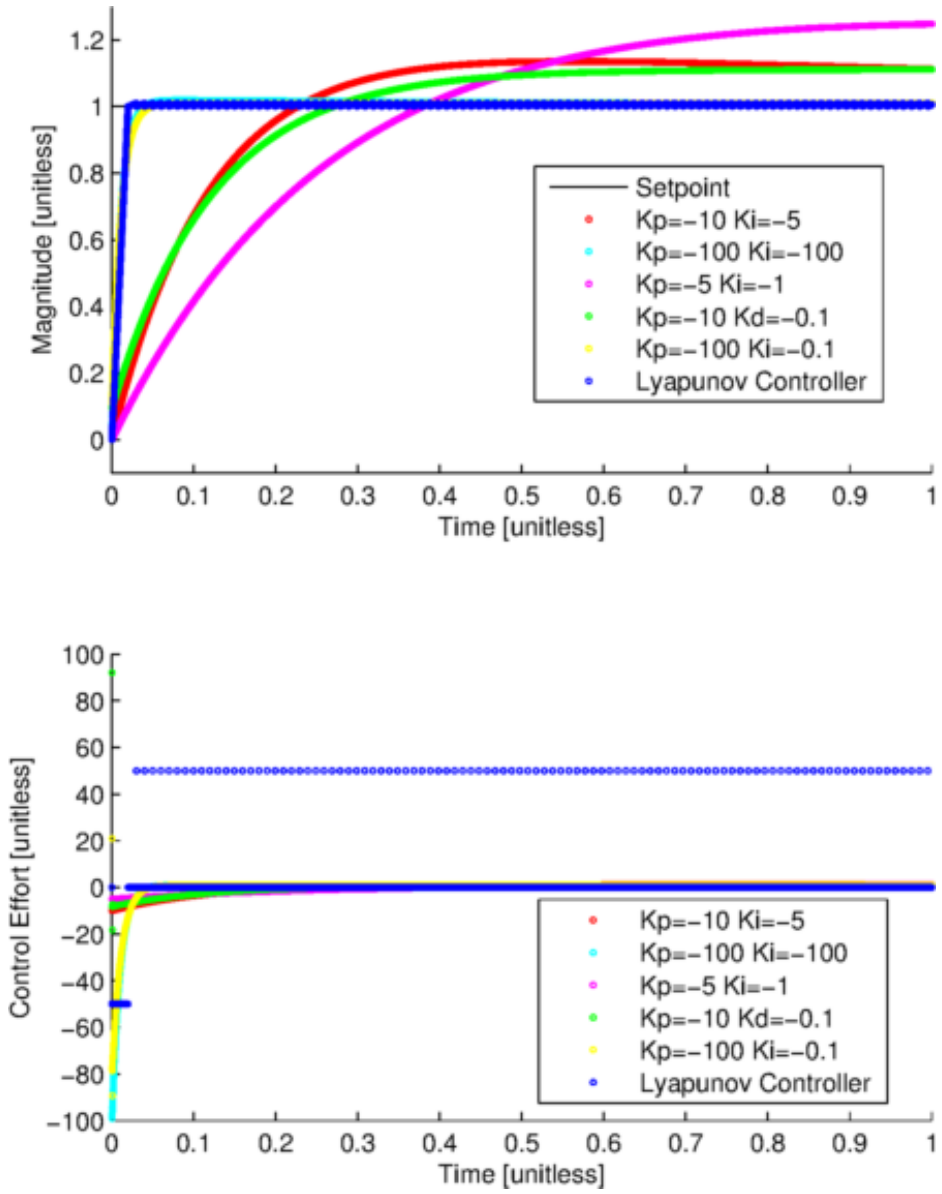


Figure 2. Comparison with PI/PD controllers. Note that the Lyapunov controller (dark blue) rises to the setpoint more quickly and without overshoot.

Rivera [1987] claims that the PID controller is the "natural choice" only "in the absence of nonlinearities, constraints, or multivariate interactions." Perhaps part of the PID controller's prevalence could be explained by the adage, "If your only tool is a hammer,

then every problem looks like a nail." A simple, practical example where a PID controller is not appropriate is power through a resistor:

$$P = I^2R \tag{1.1}$$

It might be important to control such a system, for example, by regulating current through an electric heater. A PID controller is not a good choice for this problem because of the nonlinear current term. As a quick demonstration, assume the controls designer chose to linearize this system about a nominal operating point of 50 Watts (P=50W, I=7.07A, R=1Ω):

$$P \approx 50 + 14.14(I - 7.07) + 50(R - 1) \tag{1.2}$$

If the user then selects a setpoint of 100W, the current flow calculated from Equation 1.2 would be 10.6 A. However, that heat setting will actually produce a heat output of 113W. Depending on the application, 13% error may be significant.

Cook and Samad [Cook, 2009] surveyed 64 industry engineers and 109 university faculty and found there are many disparities between the controls requirements of industry and the curricula in undergraduate controls classes. In particular, "survey data suggest that Model Predictive Control (MPC)[1] is an area of interest for industry that is not typically covered in a curriculum aimed at entry-level engineers." Industry respondents also identified "hands-on experience" as the most lacking attribute of entry-level engineers. Lyapunov stability methods, adaptive control, and robust control were cited by respondents as topics that should be covered in order to prepare entry-level engineers better.

[1] Model predictive control is a powerful nonlinear control technique.

One day, undergraduate engineers may not be limited to such a small toolset and nonlinear control techniques may thus see wider application in industry. One goal of this dissertation is to encourage this transition by extending a general and intuitive nonlinear analysis technique known as Lyapunov's Second Method. A useful new control algorithm known as a *switched Lyapunov controller* is thus developed and presented. It provides a more intuitive path towards the integration of nonlinear control into an undergraduate curricula as well as being a viable controller for use in the motivating application area: robotics. This document describes the theory behind the switched Lyapunov controller and evaluates the method on a broad range of systems to characterize and validate its performance. Two software packages are developed to make the proposed switched Lyapunov controller available for modern control problems in the robotics and other domains, and they are evaluated by a set of students who have completed an introductory course in control theory but have yet to take a course in nonlinear control.

1.2 A HISTORY OF LYAPUNOV STABILITY THEORY

Since the publication of *The General Problem of Stability in Motion* [Lyapunov, 1892], Lyapunov theory has been applied to check the stability of nonlinear systems. At the time of his dissertation's publishing, Aleksandr Lyapunov was a 35-year-old doctoral student in mathematics at Moscow University. Previously, he had studied hydrostatics and been advised by Pafnuty Chebyshev (a famous statistician in his own right). Lyapunov's stability studies began with an interest in the stability of rotating fluids and he developed two methods that are still used frequently. The First Method is based on linearization and its usefulness for nonlinear dynamic systems is limited.

Lyapunov's Second Method has become the *de facto* standard for stability analysis of nonlinear systems. It is based on the concept that if an energy-like quantity can be defined for a system, and if that "energy" is always dissipated, then the system must move toward a state of lower energy.

Other mathematicians had realized that a system's energy could be used to analyze its stability, but Lyapunov's genius lay in his recognition that the energy-like quantity need not be a physical quantity. Any scalar function is suitable as long as it satisfies a few constraints. Slotine and Li [1991] give a very clear proof of the Second Method that non-mathematicians can easily follow, and its details are presented towards the end of this chapter.

The Second Method is quite powerful and it can be applied to simple linear and nonlinear systems. Its major drawback is that a scalar-like function satisfying a partial differential equation must be found. For complex systems, it can be increasingly difficult to derive or guess a suitable "energy" function (called a *Lyapunov function*).

Lyapunov's Second Method was originally used to investigate the stability of an open-loop dynamic system, but it has found many other applications. For example, in 1983 Artstein discovered that a stabilizing control effort for a system exists if and only if a Lyapunov function exists. Furthermore, the stabilizing control effort can be derived directly from the Lyapunov function. This was a significant innovation because it turned an analysis tool into a *controller synthesis* tool. Artstein's research, and more recent contributions along the same line, are examined more closely in Chapter Two.

In addition to motion stability analysis and controller synthesis, Lyapunov's Second Method has been applied to practical problems such as:

- The calculation of the proper power for transmissions in a wireless network [Devane, 2012]
- The design of artificial intelligence algorithms [Perkins, 2002, Yerramalla, 2003]
- The prediction of chaotic motion in dynamic systems [Ryabov, 2011]

It has also been used as a stability analysis tool for models across various disciplines:

- Modeling of food chains in biology [Hsu, 1995]
- Modeling of human metabolic processes [Iglesias, 2010]
- Modeling of power systems during transient surges [Willems, 2003]
- Modeling of fluid flow [Mulone, 1989]

On a theoretical level, it has been proposed that:

- A Lyapunov controller can regulate Schrodinger's Equation, which defines the probabilistic state of an atom [Mirrahimi, 2013]. In this case the "energy function" is based on the conservation of probability.

These examples underscore the wide applicability of Lyapunov's Second Method.

This dissertation focuses on the application of Lyapunov theory to stabilize dynamic systems. The primary goal was to make practical the control of large, high-order, or coupled nonlinear systems. Part of that goal required a new approach to Lyapunov controller synthesis because, despite the power and elegance of the theory behind Lyapunov controller synthesis, it has two major drawbacks. Per Freeman and Kokotovic] [1996], "To become practical, [Lyapunov controller synthesis algorithms] must overcome

two important obstacles. First, they must expand their geometric methods to incorporate uncertainties in the system models. Second, they must deal with the crucial shortcomings of the Lyapunov approach, namely, the lack of tools for the systematic construction of Lyapunov functions."

1.3 CONTRIBUTIONS

Freeman and Kokotovic have pointed out that 1) Lyapunov controllers need improved robustness properties and 2) the difficulty of proposing and validating a Lyapunov function needs to be eliminated. *A major contribution of this effort is a technique that solves the second point by removing the necessity for a Lyapunov function*. That technique is presented in Chapter 3 and it works for a large variety (but not all) nonlinear systems. The subset of nonlinear systems which cannot be controlled is explained in Section 3.1. With the impediment of deriving a Lyapunov function removed, the practical application of Lyapunov's Second Method for controller synthesis becomes much simpler.

Clearly it is not enough to introduce a new control algorithm. Hundreds have been proposed in the literature but only a handful are used in industry [Desborough, 2000]. To be useful, a new algorithm should ideally

- be validated against industry standards, and
- be as accessible as possible, including
    - availability on popular, modern platforms, and
    - accessibile to engineers with an undergraduate level of education.

Thus, these are the goals of this dissertation. *A major step to that effect was the release of two open-source software packages that encapsulate the new theory and make*

it accessible. The first package presents a simple, graphical interface which makes it easy to learn and begin controlling nonlinear systems. However, it has speed and memory limitations, so the second software package was designed for high-performance applications (although its interface is more cumbersome). Both packages are freely available on popular platforms (ROS and MATLAB) and testing with the software packages has validated the switched Lyapunov controller against the ubiquitous PID controller.

1.4 CONVENTIONS AND ASSUMPTIONS

This document represents dynamic systems with state-space equations. The common alternative for linear systems is transfer functions. However, the state-space representation is most convenient for Lyapunov theory so it is used here. Furthermore, in many engineering disciplines (particularly mechanical), the state-space form is more intuitive and easier to interpret in relation to the physical system. This should not be a cause for concern because one can always convert a state-space representation to a transfer function and vice versa.

According to the Merriam-Webster dictionary, the words *dynamic* and *dynamical* are adjectives. It is common to see both in the literature, e.g. *dynamic system* vs. *dynamical system*. *Dynamic* is used in this document.

Since the field of control theory is very broad, some assumptions were made to narrow the scope of this document:

- All quantities are perfectly known or measured. Noise, measurement error, and unobservable systems are not considered.

- Only continuous-time systems are considered. There is no treatment of discrete-time systems.

The consideration of these complexities would not negate this work, but a formal analysis of their effects deserves additional study.

1.5 PRIOR KNOWLEDGE OF THE READER

This document is written for the reader who understands some concepts in control theory and calculus. These topics are generally taught in undergraduate classes. Any subject that would require a deeper-than-undergraduate-level background is explained more thoroughly. For a deep understanding of this document, prior knowledge of the following topics is necessary:

- Basic control theory notation
    - x is a state, y is an output, u is an input
- State-space representation of a system
- Stability of a dynamic system
    - Asymptotic versus marginal stability
- Phase diagrams
- Partial derivatives
- Basic linear algebra
- Controllability and observability
- Robustness
- Set notation is occasionally used to simplify formulas

1.6 A DETAILED REVIEW OF LYAPUNOV'S SECOND METHOD

It is important to understand Lyapunov's Second Method since the presented algorithm is an extension of it. Readers familiar with this method are encouraged to skim

this section. Consider a mass-spring-damper system. If the system is in motion, it has kinetic energy. Clearly, the system is approaching a resting point if its kinetic energy dissipates, and when its kinetic energy falls to zero, the system is *asymptotically stabilized* at its resting point. Notice that the kinetic energy for the system is a scalar no matter how complicated the system is.

Lyapunov's Second Method takes this concept of energy dissipation and extends it; the Second Method can be applied even if there is no quantification of physical energy for a system. Any scalar function of the system's states which dissipate over time can be used to prove stability (provided it meets a few conditions). That the scalar function need not be related to the actual energy of the system was Lyapunov's key insight.

The Second Method is also referred to as Lyapunov's Direct Method because the nonlinear system is analyzed directly and without linearization. The indirect method that requires linearization is known as Lyapunov's First Method or Lyapunov's Linearization Method. The First Method is still taught and used, but not as commonly as the Second because its applicability is limited. For example [Slotine and Li, 1991, pg. 53], consider the system

$$\dot{x} = -2x + bx^5$$

Linearization about $x = 0$ reveals that the linear coefficient matrix A has purely negative eigenvalues ($\lambda = -2$ in this case) so it is locally asymptotically stable. But linearization is inconclusive for the related system,

$$\dot{x} = bx^5$$

since linearization does not produce any linear terms. (The linearization about $x = 0$ is $\dot{x} \approx 5bx^4 * x|_{x=0} = 0$).

For convenience, the statement of Lyapunov's Second Method (Theorem 1) and an explanation follow.

Theorem 1 (Stability for a Static Lyapunov function)

For a system with state(s) x, assume *that there exists a scalar function* $V(\boldsymbol{x})$ *with continuous first partial derivatives such that*

- $V(\mathbf{0}) = 0$ *and* $V(\boldsymbol{x}) > 0$ *(i.e.* $V(\boldsymbol{x})$ *is positive definite)*
- $V(\boldsymbol{x}) \to \infty$ *as* $\|\boldsymbol{x}\| \to \infty$ *(i.e.* $V(\boldsymbol{x})$ *is decrescent)*
- $\dot{V}(\boldsymbol{x})$ *is negative definite*

then the equilibrium at the origin is globally asymptotically stable.

$V(\boldsymbol{x})$ is known as a *Lyapunov function* for the system. It's interesting to note that a Lyapunov function is likely not unique[2]. Figure 3 shows an example of a positive definite Lyapunov function for a first-order system. Clearly if *V* decreases, the system is moving towards the equilibrium point at zero. For a second-order system, one could imagine a bowl-shaped Lyapunov function, as shown in Figure 4.

[2] Refer to [Slotine and Li, 1991], page 62 for a graphical proof of Theorem 1.

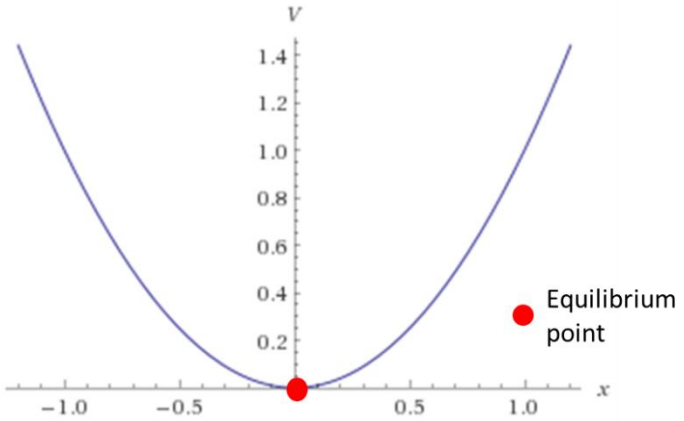


Figure 3. A positive definite Lyapunov function for a first-order system

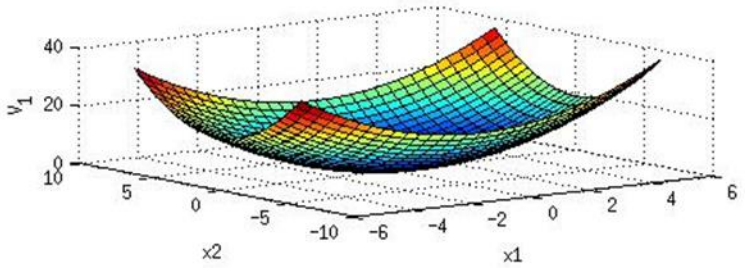


Figure 4. A positive definite Lyapunov function for a second-order system

Note that the origin is typically presented as the equilibrium point of interest for Lyapunov analyses. This might seem limiting because the designer may want to steer the system towards a different point. However, a coordinate shift can be used to present any other point as the equilibrium point of interest. For example, controlling $\dot{x} = x - 2$ to the origin is equivalent to controlling $\dot{x} = x$ to $x = 2$. So, there is no loss of generality.

Variations on Theorem 1

There are several variations on Theorem 1:

- If $V(\boldsymbol{x})$ is positive semi-definite, the system may not be asymptotically stable. It may come to rest at a local stability point that is not the origin. This situation is portrayed in Figure 5.
- If the requirement that $V(\boldsymbol{x}) \to \infty$ as $\|\boldsymbol{x}\| \to \infty$ is dropped, then the state could lie in a region where $V(\boldsymbol{x})$ does not dissipate properly and, of course, stability is not guaranteed. This situation is portrayed in Figure 6.

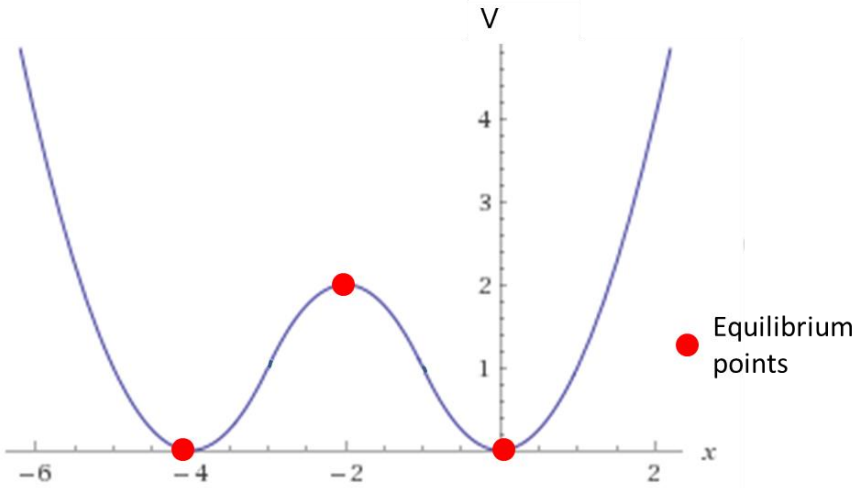


Figure 5. $V(x)$ is positive semi-definite so there are equilibrium points besides the origin.

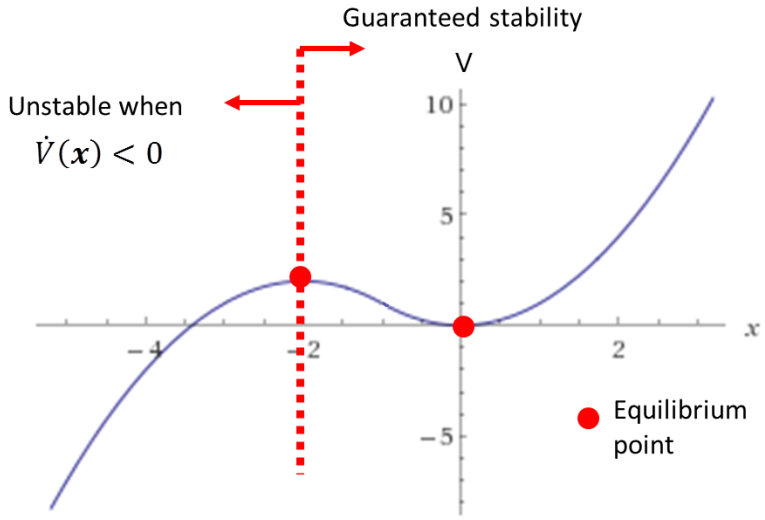


Figure 6. $V(x)$ does not have the right form as $x \to -\infty$ so multiple equilibrium points or instability are possible.

The practical application of Lyapunov's Second Method

The previous section describes why stability is guaranteed if a Lyapunov function for a system is known, but it did not discuss how to discover the Lyapunov function in the first place nor how to check whether a *candidate Lyapunov function* is valid. Finding a valid Lyapunov function is arguably the greatest challenge to the application of Lyapunov's Second Method [Freeman and Kokotovic, 1996], and more recent efforts to address this issue are discussed below.

If it is possible to derive a physically meaningful, scalar quantification of a system's energy, then that expression can be used a Lyapunov function. For example, [Aström, 2000] uses a Lyapunov function based on the kinetic energy of an inverted pendulum to stabilize it. His Lyapunov function is:

$$V = \frac{(E - E_0)^2}{2}$$

where E is the kinetic energy of the pendulum:

$$E = E_r + E_t = \frac{1}{2}J\dot{\theta}^2 + mgl(1 - cos\theta)$$

E_0 is the desired kinetic energy of the system, J is its moment of inertia, m is the pendulum's mass (point mass assumption), and l is the length of the pendulum. θ is the angle of the pendulum, where $\theta = 0$ corresponds to the pendulum in a downward position. By choosing the control effort:

$$u = k(E - E_0)\dot{\theta}\cos(\theta - 1)$$

The time derivative of V after the control effort is applied is:

$$\dot{V} = -mlk\left[(E - E_0)\dot{\theta}cos\theta\right]^2$$

$\dot{V}$ is negative as long as the quantity in the bracket is nonzero, so the system is stabilized at $E = E_0$.

There are several analytical techniques from the literature that can be used to define a Lyapunov function if the model of a system is in a particular form. Tan [2004] derived a globally valid Lyapunov function for polynomial, input-affine systems. Margaliot [1999] developed an analytical Lyapunov function for *fuzzy* systems, i.e. systems that are described linguistically with fuzzy logic. Other researchers who have presented techniques for finding locally-valid Lyapunov functions include [Curtis, 2004, Topcu, 2008]. Sassano [2013] recently presented a technique for deriving a Lyapunov function by solving a system of partial differential equations (but it only provides local stability, too). Unfortunately, Freeman and Kokotovic [1996] assert there is no general technique that applies globally to systems of any form and nothing in the recent literature has changed this.

If these analytical techniques fail, it is sometimes possible to guess the form of a candidate Lyapunov function then verify that $\dot{V}(\boldsymbol{x})$ is negative definite.

1.7 A Very Brief Description of the Method

The subject of this dissertation, the switched Lyapunov controller, is (to the best of our knowledge) unique in that it does not require any effort from the designer to derive a

Lyapunov function or guess at and verify a candidate Lyapunov function. The idea is to assume a Lyapunov function with a simple form, V_1. If the stabilizing control effort as calculated from V_1 begins to fail, the algorithm switches to another Lyapunov function, V_2. V_1 and V_2 are *jointly exhaustive*. For a single-input system of the form:

$$\dot{\mathrm{x}}_i = \mathrm{f}_i(\mathbf{x}, \mathrm{u}), \quad i \in \{1, \ldots, n\}$$

$V_1(\boldsymbol{x})$ and/or $V_2(\boldsymbol{x})$ is guaranteed to be reducible (i.e. $\frac{\partial \dot{V}_1}{\partial u}$) at any point in state-space. The three necessary assumptions are:

- $\dot{\boldsymbol{x}}$ is continuously differentiable. This is necessary for systems where linearization is required.
- "Small" control efforts are sufficient to stabilize the system. This is necessary for systems where linearization is required.
- $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1 \ldots n\}$

Chapter Three gives a full explanation of the method. Section 7.6 presents a modification to the algorithm which allows the $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1 \ldots n\}$ assumption to be dropped for dynamic systems in "normal form."

1.8 CHAPTER OUTLINE

Chapter Two comprises the literature review. A broad overview of all nonlinear control algorithms helps to identify the relative strengths and weaknesses of the proposed algorithm compared to the alternatives. There has been some prior art that used dynamic or switched Lyapunov functions as well, so a more focused and deep review of those most-similar algorithms follows the broad overview. The much deeper review is useful on a conceptual level (i.e. understanding that Lyapunov functions need not be static) and ensures the switched Lyapunov controller is unique.

Chapter Three presents the theoretical formulation of the switched Lyapunov controller.

Chapter Four covers the first software package based on the proposed control method, which is a MATLAB simulator with an easy-to-use Graphical User Interface (GUI). The MATLAB GUI was developed first because MATLAB has an extensive library of built-in mathematical functions, good graphical capabilities, and a simple GUI development environment, so it makes for a good prototyping language. Unfortunately, MATLAB is an interpreted language, which makes it rather slow. Thus, the MATLAB GUI is not a good choice for the control of hardware at high control frequencies nor for simulating large or complex systems. However, the GUI is very good for simulating the controller's performance offline and getting graphical and numerical feedback on its performance. The chapter includes results from a broad variety of these simulations that benchmark the switched Lyapunov controller against other algorithms, demonstrate the breadth of its applicability, and identify several quirks related to the controller's performance.

Chapter Five encompasses the second software package which is oriented towards speed and high performance. It was written in C++ and released for the Robot Operating System (ROS). ROS is a collection of open-source software that allows for rapid integration and simulation of hardware and its popularity has been growing rapidly. The release of the switched Lyapunov control package on ROS was a calculated step towards improving its acceptance and visibility among the robotics community. The chapter covers two demonstrations where the ROS controller's capabilities are displayed. One of these

demonstrations was performed with a physics simulator while the other was performed on hardware that is relevant for manufacturing applications. The chapter concludes with a comprehensive simulation of all of the asymptotically-stabilized systems from the previous chapter lumped together as one system. This final simulation displays the capabilities of the controller for very large, complicated systems where the high performance of the C++ implementation is necessary (the MATLAB GUI would not suffice). Together, Chapters Four and Five validate the switched Lyapunov controller.

Chapter Six returns to the initial premise: many engineering applications would be better served if engineers with only an undergraduate degree had the capability to consider and develop nonlinear controllers. To this end, a brief tutorial for the MATLAB package was developed that requires only the background knowledge of an undergraduate control curricula. It is our contention that this effort eliminates a major obstacle for the development of nonlinear controllers, and the availability of a simple tutorial provides an opportunity to evaluate this hypothesis and identify any remaining gaps.

Chapter Seven describes several improvements that were attempted in order to make the control algorithm more practical. Some of the improvements (including integral action, decibel units, and extra terms to ensure robustness) provided tangible benefits so they were included in the software releases. Other improvements such as finer integration accuracy and a different linearization technique were tested and did not yield tangible benefits, so they were not included in the software packages we have released. The chapter also describes a modification of the algorithm which is particularly useful on single-input systems having a relative degree of one or higher, such as "integrators."

Chapter Eight summarizes the controller and package capabilities and makes several recommendations for future work.

Chapter 2: Literature Review

It is important to examine the presented algorithm in the context of other control algorithms from the literature. When is it appropriate to use a switched Lyapunov controller? When would a different algorithm be a better choice? To help understand these questions, Chapter 2 begins with a brief overview of the various nonlinear control algorithms that a controls engineer has at his disposal, followed by a discussion of how the switched Lyapunov controller fits. The chapter concludes with a discussion of the concept of controllability, which is important in determining whether the switched Lyapunov algorithm could possibly control a system.

2.1 AN OVERVIEW OF NONLINEAR CONTROL ALGORITHMS

Gain Scheduling

Gain scheduling is "an attempt to apply the well-developed linear control methodology to the control of nonlinear systems." [Slotine and Li, 1991] A sufficient quantity of operating points to cover the system's region of operation is selected and a linear model of the system's behavior is derived at each point. The designer can choose from a broad array of well-defined linear control techniques to define the linear controller at each point. The techniques which might be used include pole placement, systematic or intuitive PID tuning, linear quadratic regulators, etc. Between operating points, the controller's parameters are often interpolated.

It is difficult to characterize the performance of a gain-scheduled controller in absolute terms because the term encompasses such a broad range of linear control techniques. The specific linear technique which gets interpolated can be selected depending on the performance metrics of an individual problem, and there are always trade-offs involved. For example, if robustness is the primary performance metric, then several PID controllers might be implemented. This would allow for robustness at the expense of optimality, among other tradeoffs. The performance of a gain-scheduled controller can be quite good and even approach optimal if modeling error is small and the number of linearized operating points is large. There are variations of gain-scheduled controllers which are robust [Hencey, 2010] or free from chatter (e.g. gain-scheduled proportional control).

Although gain-scheduled controllers look quite good in the controller comparison of Table 2, they have severe drawbacks. When good performance from a highly nonlinear plant is crucial, many operating points are required and the linearizations become extensive. Furthermore, the typical gain-scheduled controller does not compensate for a time-varying plant (although there are a few adaptive variations, e.g. [Annaswamy, 2008], [Pickhardt, 1998]). Slotine and Li describe the gain-scheduling-related stability theorems as "weak." Finally, the designer who implements a gain-scheduled controller must be knowledgeable enough to wisely choose his linearization points and a suitable linear control technique.

Feedback Linearization

Feedback linearization (FBL) can be used when a good model of the system is known and the control effort can be inverted, subtracted, or divided to cancel any nonlinear terms. After the nonlinear terms are canceled, additional effort can be applied to control the system like any linear system. The most general case [Khalil, 1996] is:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{\beta}^{-1}(\boldsymbol{x})[\boldsymbol{u} - \boldsymbol{\alpha}(\boldsymbol{x})] \tag{2.1}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \tag{2.2}$$

The basic approach is to identify an input $\boldsymbol{u}$ that cancels $\boldsymbol{\beta}^{-1}(\boldsymbol{x})$and $\boldsymbol{\alpha}(\boldsymbol{x})$, transforming the nonlinear system into a linear one. Once the system appears as linear, any type of control effort can be used to calculate a stabilizing $\boldsymbol{u}$. For example, simple proportional control can be used. There is a formulaic approach to FBL which involves Lie derivatives, and there are robust variations to FBL [Guillard, 2000].

The formulaic approach to FBL is one of its advantages and the concept of FBL is relatively easy to understand and implement. Unfortunately, it only applies to a limited set of nonlinear systems, it requires a good model and good derivatives, and, after the system is transformed, there may be unstable, unobservable internal dynamics.

Proportional-Integral-Derivative Controller

Proportional-integral-derivative (PID) controllers may have been the first type of nonlinear controller as research into PID's started in the early 20th century. Perhaps the first implementation was a mechanical PID-like controller that was linked to a gyrocompass for ship and airplane navigation around 1910. It was created by American Elmer Sperry, a prolific inventor and Cornell graduate who dabbled in chemistry and drove the first

American-made car in Europe. Sperry's control systems made a fortune [The New York Times, 1930]. Around the same time (1912), another American inventor (Morris Leeds) intuited that a control action more complex than bang-bang was necessary to stabilize a data recorder. Leeds patented the concept in 1920. Both Leeds and Sperry were inspired to make corrective control actions based on the rate of change of the error; they gleaned this intuition by observing human operators. [Bennet, 1993] A few years later, Minorsky translated the actions of a helmsman into the mathematical equations of a PID controller [Minorsky, 1922]. PID control was such an advancement over the prior state-of-the-art that secret meetings were held to disseminate the knowledge (the Gibson Island conferences of 1942). [Bennet, 1993] This was the dawning era of feedback control.

Since PID control is based on the error feedback from a system, it does not require a model of the system. This, along with its long history, may explain why the PID controller is still the most common industrial controller by a wide margin:

"Based on a survey of over eleven thousand controllers in the refining, chemicals and pulp and paper industries, 97% of regulatory controllers utilize PID feedback." [Desborough, 2000]

The controller's name is derived from its three terms. The Proportional feedback term is usually the largest; a proportional term alone will tend to oscillate around the setpoint and will generally operate with steady-state error. That steady state error can be corrected with the Integral term, which is proportional to the time-integral of the error. A downside of the integral term is that it can "wind up" to a very large value, which then takes a long time to unwind and causes a large overshoot.

The Derivative term is proportional to the instantaneous derivative of the error. If the error is increasing rapidly, a derivative term will increase the magnitude of the control effort to compensate. In this sense, a derivative term projects forward and reduces settling time. However, a derivative term can lead to instability, so it is often neglected (yielding a PI controller).

Although it can be fast and convenient to use a model-free PID controller, they generally do not provide optimal performance and they are restricted to smaller, less complex systems. Most of the manual approaches to PID tuning such as the Ziegler-Nichols method [Ziegler, 1942] are based on heuristics. It is possible to perturb a system with an impulse and derive the optimal PID parameters from its response and there are software packages for that purpose. However, it seems that any method of tuning a model-free controller requires some trial and error. Because a PID controller is linear, it can only be tuned for improved local performance on the linearized formulation of nonlinear problems.

Sliding Mode Controller

Sliding mode control consists of identifying a trajectory (or surface) that will carry a system towards the origin under open-loop control (like a marble rolling along a crack). The control law is then formulated to push the system towards the sliding surface. A *sliding mode* refers to the motion of the system along a sliding surface. An advantage of sliding mode control is that it can be extremely robust, while the obvious obstacle to implementation of sliding mode control is identification of the sliding surface(s). If a

system can be linearized, then the eigenvectors of the system matrix that are associated with negative eigenvalues can be used as sliding surfaces.

When a sliding surface has been hypothesized, there is a rather simple theorem to check whether it is feasible to reach the surface from a given point [Utkin, 1992]. The theorem depends on the asymptotic stability of the sliding surface, i.e. whether the system will approach the surface. For a distance from the sliding surface $\boldsymbol{\sigma}(\boldsymbol{x})$ and a Lyapunov function candidate $V\big(\boldsymbol{\sigma}(\boldsymbol{x})\big) = \frac{1}{2}\boldsymbol{\sigma}^T(\boldsymbol{x})\boldsymbol{\sigma}(\boldsymbol{x})$, the sliding surface is reachable if:

$$\boldsymbol{\sigma}^T\dot{\boldsymbol{\sigma}} = \frac{\partial V}{\partial \boldsymbol{\sigma}}\frac{\partial \boldsymbol{\sigma}}{\partial t} = \frac{\partial V}{\partial t} < 0 \tag{2.3}$$

The first term in Equation 2.3 is the dot product $\boldsymbol{\sigma} \cdot \dot{\boldsymbol{\sigma}}$. In the single-input case, it is clear to see that Equation 2.3 checks whether σ and $\dot{\sigma}$ have opposite signs. If Equation 2.3 is not satisfied under open-loop control, the next task is to design a $\boldsymbol{u}(\boldsymbol{x})$ that causes it to become true. Once the surface has been reached, the system will (under ideal conditions) be restricted to the surface for all subsequent time as it "slides" towards the target.

After a valid sliding surface has been identified, the next issue is to identify its region of attraction. It may be necessary to include several surfaces to ensure the entire region of interest is covered. Assuming a suitable sliding surface(s) can be identified, other drawbacks remain. In order to force the system onto the sliding surface, a discontinuous switching controller is often used, i.e. the controller pushes towards the surface when the system is on one side, and it pushes in the opposite direction from the other side. This discontinuous behavior is characterized as *chatter* and it wastes energy and causes

premature wear on hardware systems. For first-order systems, the only control function that guarantees robust asymptotic stability is the discontinuous switching function:

$$\dot{x} = \partial - \eta * \text{sign}(\sigma) \tag{2.4}$$

Where ∂ accounts for uncertainty or disturbances in the system and η is a gain which is specified by the designer of the controller. The vernacular term for such a controller is a *bang-bang controller*. The following aside shows why a bang-bang controller (and the chatter it causes) are necessary to ensure robust asymptotic stability. This is relevant to the dissertation because the switched Lyapunov controller also tends to chatter as it drives a system towards asymptotic stability.

Aside: Proof of Robust Sliding Surface Stability with a Bang-Bang Controller (First-Order System)

Define a Lyapunov function candidate, based on the distance from a sliding surface, σ:

$$V(\sigma) = \frac{1}{2}\sigma^2 \tag{2.5}$$

In order for the system to asymptotically approach the sliding surface, the requirement is:

$$\dot{V} < 0 \; \forall \; \sigma \in \mathbb{R} \tag{2.6}$$

Evaluating Equation 2.6 based on Equation 2.5:

$$\dot{V} = \sigma\dot{\sigma} = \sigma\big(\partial - \eta * \text{sign}(\sigma)\big) < 0 \tag{2.7}$$

$$\sigma\left(\partial - \sigma\,\eta * \frac{|\sigma|}{\sigma^2}\right) = \sigma\partial - |\sigma|\,\eta < 0 \tag{2.8}$$

$$\partial < \eta * \text{sign}(\sigma) \tag{2.9}$$

Equation 2.9 shows that the system can be forced to asymptotically approach the sliding surface by choosing a sufficiently large gain (η) for the bang-bang controller.

Some techniques can be used to reduce chatter. For example, the control effort can be reduced exponentially as the system approaches the sliding surface. In this case, the control effort behaves smoothly if the system bounces back and forth across the switching surface. However, robustness is lost due to the fact that $\boldsymbol{u}(\boldsymbol{\sigma} \to \mathbf{0}) = \mathbf{0}$. In words, there is no extra control effort to account for modeling errors or noise when the system is very near to $\boldsymbol{\sigma} = \mathbf{0}$. Smoothness is gained at the expense of robustness.

In summary, sliding mode controllers are powerful, but the mathematics of identifying an appropriate sliding surface may be daunting and the designer needs to choose between robustness and chatter. This section also serves as a reminder of the power and broad applicability of Lyapunov's Second Method because it was used in both an existence theorem and a stability proof.

Backstepping Controller

Backstepping is a new nonlinear control technique developed around 1990 by a Serbian immigrant to the United States who was 57 years of age at the time [Kokotović, 1992]. Unfortunately, it is only applicable to a narrow range of systems that are in *strict feedback form* (Equation 2.10 – also referred to as *cascaded form*).

$$\left\{\begin{array}{c} \dot{\boldsymbol{x}} = f_x(\boldsymbol{x}) + g_x(\boldsymbol{x})z_1 \\ \dot{z}_1 = f_1(\boldsymbol{x}, z_1) + g_1(\boldsymbol{x}, z_1)z_2 \\ \dot{z}_2 = f_2(\boldsymbol{x}, z_1, z_2) + g_1(\boldsymbol{x}, z_1, z_2)z_3 \\ \vdots \\ \dot{z}_k = f_k(\boldsymbol{x}, z_1, z_2, \ldots, z_k) + g_1(\boldsymbol{x}, z_1, z_2, \ldots, z_k)u \end{array}\right\} \tag{2.10}$$

With these additional constraints on some terms:

$$f_i(\mathbf{0}, 0, \ldots, 0) = 0 \tag{2.11}$$

$$Over\ the\ region\ of\ interest\colon g_i(\boldsymbol{x}, z_1, \ldots, z_k) \neq 0\ for\ 1 \leq i \leq k \tag{2.12}$$

The f_i constraint is equivalent to requiring open-loop stability at the origin, and the g_i constraint is applied to avoid singularities during the backstepping process. It might seem unusual to find a real system in strict feedback form, but cascaded integrators are one type that fits the description. For example, (controlled jerk)➔acceleration➔velocity is one control problem where backstepping would be useful. In the literature, the practical problems that backstepping have been applied to include motion tracking of mobile robots [Jiang, 1997] and quadcopter motion control [Madani, 2006]. In general, however, most of the literature in backstepping focuses on theory and there has been relatively little practical application of the technique.

Notice that the control input in Equation 2.10 has a direct effect only on state z_k but each state is directly affected the previous state. The basic idea is to view z_k as a "fictitious" input which stabilizes z_{k-1}. z_{k-1} then stabilizes z_{k-2} and so on, moving upwards and stabilizing each state at a time. To calculate the stabilizing control effort *u*, one starts at the top, calculating a z_{k-1} that will stabilize **x** and recursively stepping back through the other states. A significant hurdle to the design of a backstepping controller is that, in order to start the calculations at **x**, one must know a control effort that will stabilize **x**. Lyapunov theory is typically used to make this initial calculation, meaning that the designer has to master both Lyapunov theory and backstepping. Fortunately, after the initial calculation of a stabilizing control effort for $\boldsymbol{x}$, the other calculations are formulaic in nature.

Kokotović's original backstepping controller is not adaptive [Kokotović, 1992], but many adaptive modifications have been proposed [Krstić, 1992]. These adaptive variations are capable of stabilizing systems with unknown parameters, which makes them robust in a sense. There are also robust modifications to the original backstepping technique that can handle bounded parametric uncertainties without adapting [Yu, 2011]. Some algorithms are both adaptive and robust [Yi, 2004] [Jiang, 1998] [Jiang, 2002] [Tomei, 2010].

Optimal Control

Optimal control requires a cost function based on the states and inputs of a system. The optimization problem is then solved, deriving the paths of each variable that will minimize the cost function at a given time. This is a very powerful approach and the only one that will yield an optimal solution. Optimal control also allows for the imposition of ancillary constraints such as saturation limits, energy limits, etc. Finally, the designer is free to specify the cost function, shaping the behavior of the system as he pleases. For example, the following are valid objectives:

- Minimize the rise time of the system,
- Minimize abrupt changes in the control effort,
- Minimize $\int u \cdot dx$, the energy input into the system,
- Any weighted combination of the above.

By comparison, the ideal performance and flexibility of an optimal controller makes the other controllers that have been reviewed seem like blunt instruments compared to a precision scalpel. The downside is that optimal control is essentially suitable only for offline applications. It has a high computational cost and cannot adjust to parameter

uncertainty, measurement noise, or other disturbances. Finally, the numerical methods that are used to solve the optimization problem are complex. The approach that should be taken depends on the system at end; certain optimization techniques (e.g. solving a steady-state Hamilton-Jacobi-Isaacs partial differential equation, [Freeman and Kokotović, 1996]) are only feasible for simple systems, for example, so the designer of an optimal controller needs to be deeply knowledgeable in the field.

In certain instances, there are some guarantees on robustness for optimal controllers. For example, if the system dynamics consist of linear differential equations and the cost function is quadratic, the optimization problem is known as a Linear Quadratic Regulator (LQR) and there are very wide margins of stability. Per [How, 2010], "these robustness margins are very large on the scale of what is normally possible for classical control systems." However, the authors are not aware of a general proof of robustness for optimal controllers.

The basic optimal controller is not adaptive and the *a priori* optimization of an uncertain system would seem to be impossible. However, there are online learning methods that approximate the behavior of an optimal controller while constantly updating the parameters of a system with an observer [Sutton, 1992] [Bertsekas, 1996]. This type of approach blurs the line between optimal control, Model Predictive Control, and machine learning.

For linear, time-invariant, continuous-time systems, [Vrabie, 2007] proposed an online "optimal" controller that ignores the internal dynamics of a system (system matrix A), does not require direct measurements of the state derivatives, and converges on the

optimal solution. This algorithm uses a critic to evaluate and adjust the performance of the controller in an online fashion.

Model Predictive Control

Model Predictive Control is similar to optimal control but it optimizes ahead for a prescribed number of time steps rather than for the entire timespan of interest. It is also known as Receding Horizon Control (while optimal control is referred to as Infinite Horizon Control). In practice, the MPC technique generally works well but it is almost never optimal. The major drawback of MPC may be its heavy computational cost: "MPC schemes tend to be quite costly computation-wise compared with classical control methods." [Necoara, 2013] For that reason, MPC is generally limited to applications with slow sampling rates. A survey by Qin [2000] found that 60% of industrial MPC applications were found in the chemical, refining, and petrochemical industries, while only 0.3% were concentrated in the aerospace/defense industry (which presumably requires higher sampling rates). It is also rare to see MPC applied to an online process.

As with optimal control, the root cause of MPC's high computational cost is the numerical solution of an optimization problem. The solutions to convex, linear optimization problems can be found rapidly and accurately. However, non-convex nonlinear optimization problems take much longer to solve. The solver is typically either truncated after a predefined period, yielding a rough approximation, or run for a longer period until a convergence threshold is reached [Diehl, 2009]. There has been heavy research into reducing the computational cost of MPC, including decomposing the single optimization problem into smaller problems [Necoara, 2013].

Extremum Seeking

Extremum seeking is a method where the gradient of an unknown dynamic system is discovered by applying perturbations. These perturbations can be systematic (e.g. sine waves) or stochastic. Once the minimum of a surface is identified, the corresponding parameter set can be used to control a system in an optimal way. Extremum seeking was a common technique in the mid-20th century and it has reemerged since a 2000 proof of stability [Krstić & Wang, 2000] [Liu & Krstić, 2012]. The main advantages are its model-free nature and its computational requirements (which are low enough to run in real time.) Drawbacks of the algorithm include a delay as the gradient is first mapped (refer to Figure 7) and the possibility of becoming "stuck" in local minima. There are robust and adaptive extensions of the technique [Zhang, 2009] [Bizon, 2010].

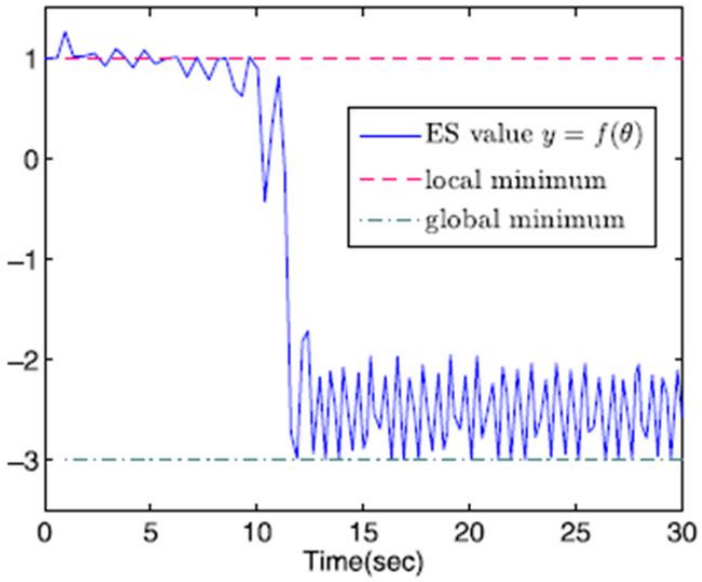


Figure 7. An application of extremum seeking. Note the delay as the controller searches for the global minimum, then the chatter after the global minimum is found.

Control Lyapunov Function

The concept of using a Lyapunov function to calculate a stabilizing control effort was proposed by [Artstein, 1983]. Artstein extended Lyapunov's Second Method to show that a stabilizing control effort exists for a system if and only if a *control Lyapunov function* [CLF] exists. The distinction between a CLF and a standard Lyapunov function, V, is that a CLF is a function of x and u. For a system of the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{2.13}$$

The asymptotic stability criterion for a CLF is:

$$\forall \boldsymbol{x} \neq \boldsymbol{0},\ \dot{V}(\boldsymbol{x}, \boldsymbol{u}) < 0 \tag{2.14}$$

Whereas a standard Lyapunov function is only a function of the state x but the asymptotic stability requirement is very similar:

$$\forall \boldsymbol{x} \neq \boldsymbol{0},\ \dot{V}(\boldsymbol{x}) < 0 \tag{2.15}$$

The other requirements for a CLF (continuous differentiability, positive-definiteness) are the same as for a standard Lyapunov function. Note that it is possible for a standard Lyapunov function to be an *implicit* function of u, which makes the distinction between a CLF and a standard Lyapunov function more confusing.

More importantly, once a CLF is known, a stabilizing u can be calculated with a simple formula:

$$u^* = \arg\min_{\boldsymbol{u}} \nabla V(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \arg\min_{\boldsymbol{u}} \dot{\mathrm{V}}\,(\boldsymbol{x}, \boldsymbol{u}) \tag{2.16}$$

So, the u that reduces V at the greatest rate is stabilizing. Equation 2.16 is a *static nonlinear optimization problem* and there are many techniques for solving such problems. There are

many other methods to calculate a stabilizing control effort when a CLF is known. Lin and Sontag [1991] is often cited.

In practice, the process of deriving a stabilizing control effort from a CLF is similar to testing stability with a standard Lyapunov function. A candidate CLF is proposed based on intuition, knowledge of the system, or trial-and-error. Assuming the candidate CLF is valid, a candidate stabilizing u is calculated with Equation 2.16 or proposed by other means. Finally, to check whether the candidate CLF is valid, the candidate u is plugged back into the system dynamics and Equation 2.14 is verified.

Artstein's formula for a stabilizing control effort from a CLF may be the most direct but other methods have also been proposed [Curtis, 2004] [Freeman and Kokotović, 1996]. In all of these approaches, the main challenge is the proposal and validation of a CLF. Per Freeman [1996], "the question of how to find a CLF remains open." There are several methods for calculating a locally valid CLF [Curtis, 2004] [Topcu, 2008] or a CLF which is valid for certain classes of system (e.g. polynomial and input-affine [Tan, 2004]), but there is no known formula for a general, global CLF.

Artstein does not discuss robustness in his original paper, but Freeman and Kokotović wrote a book on the subject [*Robust Nonlinear Control Design*, 1996]. Other researchers who have developed techniques for robust CLF control include [Feron, 1996] [Lin, 1996].

A potentially useful feature of Artstein's CLF-based controller is that it reduces V as rapidly as possible. There is freedom for the designer to tune the controller either by picking a different $V(x,u)$ or by modifying the equation for control effort and double-

checking that $\dot{V}(x) < 0$ is still satisfied. Freeman and Kokotović [1996] discuss the concept of inverse optimality, or how arbitrary performance metrics can be optimized while arbitrary constraints are met via a static nonlinear optimization of u. So, it is possible to tune a CLF-synthesized controller, but the underlying mathematics are difficult and the computational cost is high. The author wonders if it is worthwhile to tune a CLF-synthesized controller when the Optimal Control approach could be used instead.

Switched Lyapunov Controller

The details of the newly-proposed switched Lyapunov controller will be presented in the next chapter, but an overview of its features is presented here so it can be compared against other methods. The switched Lyapunov controller is computationally fast, with speed that is approximately on-par with PID controllers. Unlike PID controllers, it can be applied to systems of any order, coupled or uncoupled, and with any number of inputs. In terms of general applicability, it is matched only by Optimal Control, its cousin, Model Predictive Control, and Extremum Seeking. It is also similar to Optimal Control/Model Predictive Control because it requires a model of the system.

The performance of the switched Lyapunov controller has not been exhaustively studied yet, nor has its robustness as these were beyond the scope of this dissertation. We can say that it typically performs better than a PID controller with reasonable gains on a step response test because it rises faster and does not overshoot. A simulation in Chapter 4 has shown that it can be robust, but the extent of its robustness should be characterized better. Chapter 7 suggests techniques that can be used to ensure robustness in some situations.

The best traits of the switched Lyapunov controller are its aforementioned speed for high-dimensional or coupled problems, which cannot be matched by other methods, and the ease of defining and tuning the controller. No matter how large the dynamic system is, the switched Lyapunov controller requires just three parameters to be set and/or tuned. Those parameters are controller aggressiveness (a scalar), size of the time step (a scalar), and a vector of saturation limits. For example, Table 1 gives a comparison on the parameters that must be defined for a 100^{th}-order, 100-input PID controller vs a 100^{th}-order, 100-input switched Lyapunov controller.

Table 1. Number of parameters to be tuned for each controller

PID	**Switched Lyapunov Controller**
100-vector of (+) saturation limits	100-vector of (+) saturation limits
100-vector of (-) saturation limits	100-vector of (-) saturation limits
100-vector of Proportional constants	Scalar of aggressiveness
100-vector of Integral constants	Scalar of time step
100-vector of Derivative constants	n/a
Total: 500 parameters (5*n*)	**Total: 202 parameters (2*n*+2)**

On the other hand, the switched Lyapunov controller has some limitations. Its performance, in some cases, depends on a small time step. This may limit its application to some hardware tasks and certainly requires more computational resources during simulations. It is also subject to performance quirks when mixed units are used. Since the

errors of every state are lumped into one scalar error measurement $V(\boldsymbol{x})$, a difference in units will weight one state more heavily than another. This is not necessarily a flaw because the designer may wish to place more emphasis on some errors (and correct them faster), but it is an oddity that needs to be understood. However, this weighting consideration applies to all CLF control methods. In Chapter 7, a method of normalization to eliminate incompatible units is presented.

Summary

Of the four classes of nonlinear controller that are suitable for large or complex control problems, the switched Lyapunov controller may be the fastest (computationally) and easiest to use. The author would recommend it whenever a PID controller is unsuitable and a fast control rate is necessary. If offline computation is a possibility, then Model Predictive Control or Optimal Control will yield better performance. The other nonlinear controller which is fast and suitable for complex systems is extremum seeking, but it has other drawbacks. Every other control algorithm that has been reviewed is a niche method.

Table 2. Comparison of nonlinear control algorithms

	Computationally Fast	Applicable to Large/Complex Systems	Model-Free	Instantaneous Action	Proven Stability	Does Not Chatter	Insensitive to Choice of Units	Zero Steady-State Offset	Robust
Gain Scheduling (many variations)	✓	✓	✗	✓	✓	✓	✓	✓	✓
PID Control	✓	✗	✓	✓	✗	✓	✓	✓	✓
Feedback Linearization	✓	✗	✗	✓	✓	✓	✓	✓	✓
Model Predictive Control	✗	✓	✗	✓	✗	✓	✓	✓	✓
Sliding Mode Control	✓	✗	✗	✓	✓	✗	✓	✓	✓
Backstepping	✓	✗	✗	✓	✓	✓	✓	✓	✓
Optimal Control	✗	✓	✗	✓	✓	✓	✓	✓	✗
Extremum Seeking	✓	✓	✓	✗	✓	✗	✓	✗	✓
CLF	✓	✗	✗	✓	✓	✗	✓	✓	✓
Switched Lyapunov Control	✓	✓	✗	✓	✓	✗	✗	✗	?

2.2 AN IN-DEPTH SURVEY OF DYNAMIC LYAPUNOV ALGORITHMS

While the previous section was a broad but shallow survey of all nonlinear control algorithms, this section focuses on a narrow but deep review of the literature that is most similar to the proposed switched Lyapunov controller. Both surveys are useful for placing the proposed controller in context. This section looks at cases in the literature where a Lyapunov function that changes form over time or based on the state has been proposed.

The concept of switching from one Lyapunov function to another is not new; it was proposed in 1991 by Peleties *et al*. who were studying switched dynamic systems. Since the system changes behavior rapidly, it makes sense on an intuitive level that the Lyapunov function should also switch behavior rapidly. Peleties's work was based a simple, first order linear system,

$$\dot{x} = A_i x \tag{2.17}$$

where A_i switches through a finite number of forms ($i \in \{1,2,\dots,m\}$) but the control system designer does not need to know when or where the system will switch *a priori*. Peleties derived several theorems that investigate how many different "Lyapunov-like" functions are needed to cover the entire possible state space and how large the region associated with each Lyapunov-like function should be as there cannot be any gaps between the regions. If the "energy" of the system decreases even as the system crosses from one region to another, then the system will be stabilized. Peleties did not propose any method to derive the Lyapunov-like function for each region, so his papers are not extremely helpful from a practical standpoint. In fact, the proposal and testing of multiple Lyapunov-like functions places a greater burden on the designer.

Recent authors who have expanded on Peleties' piecewise Lyapunov approach for switched dynamic systems include Branicky, [1998] Daafouz, [2002] and Du, [2007]. Whereas Peleties' work focused on very simple linear switched systems, Branicky

expanded the concept to include general nonlinear switched systems. Even with this expansion in scope, the applicability of these approaches was limited. Nevertheless, it was an important intellectual leap to consider multiple Lyapunov functions simultaneously.

There have been several authors who propose *fuzzy* Lyapunov functions, where fuzzy indicates that a smoothly-varying Lyapunov function is calculated by a blending of multiple simple functions. Margaliot [1998] appears to be the first researcher who published on this approach. A very nice feature of Margaliot's work is that the fuzzy Lyapunov functions can be derived *analytically*, eliminating the time-consuming "guess-and-check." Another feature is its ability to handle uncertainty (i.e. robustness). Margaliot [1999] extends the controller to be adaptive. Fuzzy rules of the controller adjust in an online fashion to track a $\dot{V}_{target}$ (convergence of $\dot{V}$ on $\dot{V}_{target}$ is guaranteed if $\dot{V}$ and $\dot{V}_{target}$ are chosen properly).

Margaliot's adaptive algorithm is a powerful tool but one major drawback is that the model of the plant must be *fuzzified*, i.e. the model cannot be described in numerical terms; it must be described in linguistic (i.e. if… then…) rules. This is likely to limit the complexity of the model. For example, Margaliot's adaptive algorithm could probably not describe nor stabilize a coupled system. Margaliot's choice of V and $\dot{V}_{target}$ must be made carefully; otherwise it may be impossible to satisfy $\dot{V} = \dot{V}_{target}$. The specific failure example given is when the system dynamics create the following decay in the Lyapunov "energy" of the system:

$$\dot{V} = x_2(\dot{x}_1 + p + qu) \tag{2.18}$$

But the target $\dot{V}$ that the designer chose for the controller to enforce is:

$$\dot{V}_{target} = -(x_1^2 + x_2^2) \tag{2.19}$$

Setting the two quantities equal ($\dot{V} = \dot{V}_{target}$) and solving for u yields:

$$u = \frac{-x_1^2}{qx_2} + \left(\frac{x_2 - x_1 - p}{q}\right) \tag{2.20}$$

Clearly Equation 2.20 does not yield a practical control effort when $x_2 = 0$ or $q = 0$. The singularities arising from $x_2 = 0$ are represented in Figure 8.

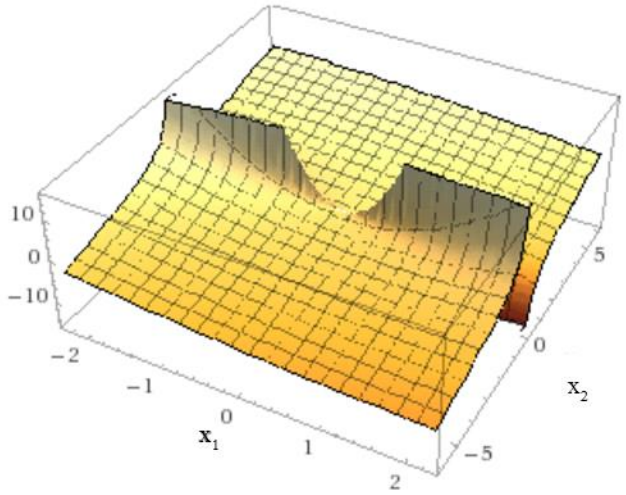


Figure 8. The surface created by Eqn. 2.20 for p=1, q=1. Note the discontinuity as $\|x_2\| \to 0$, which would lead Margaliot's controller to fail.

Tanaka [2007] shows that a fuzzy Lyapunov function can guarantee stability for a wider range of systems than a standard CLF or a piecewise Lyapunov function. However, there is still (at least in Tanaka's example) a point beyond which a single fuzzy Lyapunov function fails to guarantee stability. Despite these shortcomings, the author considers fuzzy Lyapunov functions to be another important milestone in expanding the concept of traditional CLF's.

Sassano [2013] presented the concept of time-varying Lyapunov functions. This is also a conceptually ground-breaking approach because, in the original formulation, Lyapunov functions could be a function of the state only ($V(\mathbf{x})$). Nor does time appear as an explicit argument for Control Lyapunov Functions ($V(\mathbf{x},\mathbf{u})$). Sassano names his new format of Lyapunov-like function $V(\mathbf{x},\mathbf{t})$ a "dynamic Lyapunov function." Unfortunately,

Sassano's approach requires the solution of a partial differential equation, so it is not clear that his approach makes the challenge of finding a suitable Lyapunov function any easier than the traditional method. Sassano's approach is also limited to ensuring local stability.

A very recent conference paper [McCourt, 2015] is the most similar to the presented algorithm. In fact, it is a remarkable coincidence that myself [Zelenak, 2015, Stabilization of nonlinear systems by switched Lyapunov function] and McCourt independently published our respective algorithms in the same month, 124 years after Lyapunov's thesis. McCourt echoes many of the points which I have emphasized previously in the literature review:

- "Finding a CLF may not be straightforward."
- There are well-known computational techniques for deriving a CLF only for a few types of system.

McCourt's approach is to pre-compute many CLF's, each of which can stabilize the system for a given range of inputs. His technique for deriving this set is clever and it is a significant contribution. When control would be lost with one CLF, a different CLF is applied. One of the nice aspects of McCourt's approach is that input constraints can be considered, and the range of validity for each CLF is known in advance. McCourt's approach is similar to ours in several respects:

- At each time, the control input is determined from a single CLF.
- When controllability would be lost with one CLF, the algorithm regains control by switching to a different CLF.
- McCourt uses the same V_1/V_2 notation in his examples that we have used.

However, there are several drawbacks to McCourt's approach. First, his example shows that V_1/V_2 do not always decrease asymptotically. Note the rise around t=2s in Figure 9. McCourt does not give an explanation for that failure. Furthermore, McCourt

focused solely on input-affine systems, i.e. systems of the form $\dot{x} = f(x) + g(x)u$, so his approach is not as general. Finally, there is a need to perform a sum-of-squares search to determine a set of CLF's for each system. This search starts with a known, stabilizing control input, and the need to know that input in advance is another difficulty. Simulation 7 of Chapter Four is an in-depth comparison against McCourt's method.

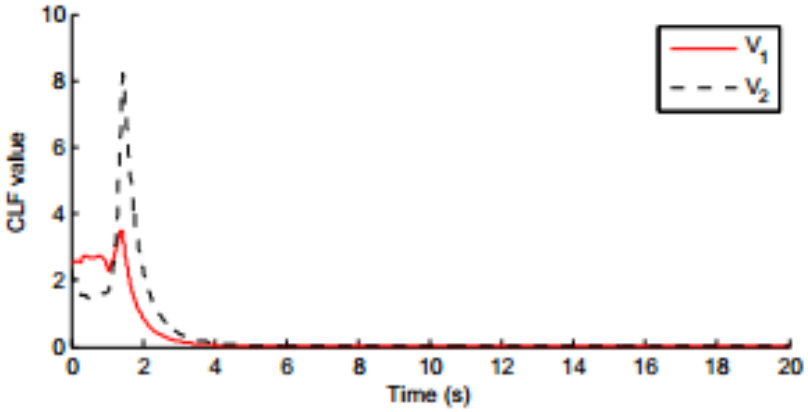


Figure 9. McCourt's CLF's. Note the spike in the values of both CLF's in the early stage of the simulation as the system is briefly unstable. It does recover at $t \approx 2s$.

In summary, over 24 years of research establish that a Lyapunov function need not be static. It is possible for a Lyapunov function to change form over time or as the state changes. Nevertheless, the dynamic Lyapunov functions that have been reviewed are limited to systems of a particular form (McCourt, Peleties, Daafouz, Du, Branicky, Margaliot) and/or limited to ensuring local stability (Tanaka, Sassano) and/or they may fail in unexpected ways (Margaliot).

2.3 CONTROLLABILITY

Regardless of the type of controller that an engineer might apply to stabilize any particular system, there are many systems which are physically impossible to regulate. Whether it is possible to regulate a given system is described by a property called "controllability." Controllability is global and binary: systems are either globally

controllable or they aren't. However, there are weaker variations such as local accessibility, which signifies that a system can only be regulated over a portion of the state space.

When presented with a new dynamic system, it is worthwhile to check its controllability before attempting to control it. In words, controllability means that an input can be applied to a dynamic system which will move it from any point in state space to any other point in a finite amount of time. A practical example of an uncontrollable system is a three-gimbal gyroscope which may undergo gimbal lock.

For linear systems, there is a mathematical formula for controllability that is well known and easily calculated. There are also controllability formulas for a common class of nonlinear systems. This section presents those formulas which will be applied during analysis of the switched Lyapunov control algorithm in Chapter Four.

Controllability of linear time-invariant systems

For a linear system of *n* states and *r* inputs of the form:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{2.21}$$

The controllability matrix is:

$$C = [B\ AB\ A^2B \ldots A^{n-1}B] \tag{2.22}$$

If the *C* matrix has full row rank, then the system is controllable. In other words, if the rows of the matrix are linearly independent, the system is controllable.

On an intuitive level, it is easy to understand why *B* is included in the controllability matrix. *B* describes the direct effect that each input has on each state. If each state is driven by a separate input, then *C* will have full rank (via *B*) and the system is controllable. The BA^i terms are present to check whether the open-loop dynamics are linearly independent from the control inputs. If the open-loop dynamics will move the system in a different

direction from the control input, then it gives another method for manipulating the system. Consider the second-order, single-input system:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} u \tag{2.23}$$

The input u will move the system along a $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ vector in the phase plane. The open-loop dynamics will move the system along a $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ vector in the phase plane. Since these two vectors are linearly independent, the system is controllable. The controllability matrix is:

$$C = [B \;\; AB] = \begin{bmatrix} 2 & 0 \\ 1 & -2 \end{bmatrix} \tag{2.24}$$

Which has rank(C)=2. This calculation confirms that the system is controllable.

The following graphic illustrates how one could use these two independent vectors to move the system from the origin to the point (-3,4). First, one could use a very negative control effort to drive the system to approximately (-3,-1.5). Then the control input could be shut off and the system would drift vertically to the target location.

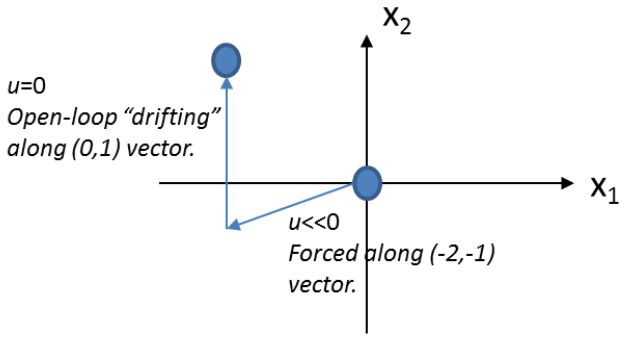


Figure 10. Open-loop "drift" adds another direction of control for this linear system.

Controllability of input-affine nonlinear systems

As with the linear system, there is an easy-to-calculate matrix that can be used to check controllability for input-affine nonlinear systems of the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \sum_{i=1}^{m} \boldsymbol{g_i}(\boldsymbol{x})\boldsymbol{u_i} \tag{2.25}$$

For nonlinear systems, a further delineation needs to be defined between *controllability* and *local accessibility.* Controllability implies that a system can be manipulated anywhere globally, but van der Schaft [2010] gives an example where a nonlinear system can only be manipulated locally. For that reason, local accessibility is more commonly studied for nonlinear systems, and it is a weaker form of controllability. The author thinks of accessibility as being able to "drift" a system in any direction, but not necessarily having control over the speed of the system. The matrix (i.e. the "accessibility distribution") that allows for a quick check of local accessibility for an input-affine system is given by Hedrick [2005]:

$$C = \left[\boldsymbol{g_1} \dots \boldsymbol{g_m} \left[\boldsymbol{g_i}, \boldsymbol{g_j}\right] \left[ad_{\boldsymbol{g_i}}^{k} \boldsymbol{g_j}\right] \dots [\boldsymbol{f}, \boldsymbol{g_i}] \dots \left[ad_{\boldsymbol{f_i}}^{k} \boldsymbol{g_i}\right]\right] \tag{2.26}$$

Where the system is locally accessible if C has full row rank. The bracket notation, e.g. $\left[\boldsymbol{g_i}, \boldsymbol{g_j}\right]$, denotes the Lie bracket operator:

$$\left[\boldsymbol{g_i}, \boldsymbol{g_j}\right] = \frac{\partial \boldsymbol{g_j}}{\partial \boldsymbol{x}} \boldsymbol{g_i} - \frac{\partial \boldsymbol{g_i}}{\partial \boldsymbol{x}} \boldsymbol{g_j} \tag{2.27}$$

$\left[ad_{\boldsymbol{g_i}}^{k} \boldsymbol{g_j}\right]$ denotes a higher-order Lie bracket (i.e. the Lie bracket of a Lie bracket). Hedrick also presents an algebraic trick that can *sometimes* be used to make more complicated nonlinear systems appear input-affine. The input-affine calculation is much simpler than the general nonlinear controllability calculation.

The $\boldsymbol{g_i}$ terms are analogous to the B terms in the linear case; they account for the direct effect of each control effort on each state. The other terms are all Lie brackets. Lie brackets have been described as a measurement of how two functions change together, similar to a dot product. If a Lie bracket is equal to zero, the functions change together and they do not contribute to controllability (similar to a dot product equal to one for parallel

unit vectors). So, the $[\boldsymbol{g_i}, \boldsymbol{g_j}]$ terms account for coupling; movement along the x_1 axis might cause the previously identical $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$ terms to become dissimilar, for example, which might open another "degree of freedom" for controlling the system.

Finally, similar to the BA^i terms in the linear case, the $[\boldsymbol{f}, \boldsymbol{g_i}]$ terms account for the similarity between the open-loop dynamics of the system and the direct effect due to the inputs, $\boldsymbol{u}$. If the open-loop dynamics of the system cause it to "drift" in a different direction from the inputs, then it is analogous to another degree of freedom for controlling the system and these Lie brackets will have nonzero terms. In some sense, it is nice to have more terms in the controllability matrix of a nonlinear system because it provides more "degrees of freedom" for controlling the system. The downside is that every term could change at any point, whereas they are constant for a linear system.

2.4 CHAPTER SUMMARY

This concludes Chapter Two, which presented essential background material on the field of nonlinear control theory and dynamic Lyapunov theory in particular. The next chapter delves into the mathematics of the novel nonlinear control algorithm which may be unique in its ability to control complex systems with minor computational expense and no delay.

Chapter 3: Description of the Control Algorithm

In this chapter, a control algorithm that stabilizes nonlinear systems with any order and number of inputs (provided several conditions) is described. The algorithm is based on Lyapunov's Second Method and it removes the burden of deriving or guessing-and-verifying a Lyapunov function. Briefly, the algorithm functions by assuming a CLF and linearizing the system dynamics to calculate a stabilizing control effort. If a singularity arises from the linearization, a different CLF with a pre-determined form is assumed and a stabilizing control effort is calculated from it. The linearizations derived from the two functions are jointly exhaustive: one is guaranteed to provide a stabilizing control effort (provided several conditions).

The default CLF (V_1) has a static form, globally. The form of the second CLF (V_2) varies in a pointwise fashion, so it is technically a set of CLF's.

The chapter begins with an explanation and proof for a second-order single-input system. A second-order system is sufficiently complex to prove the concept but simple enough for clear, intuitive proofs. Then, methods for extending the concept to systems of any order and any number of inputs is shown. The chapter includes a special analysis of the trivial case of first-order systems because they are very common, easy analyzed, and uniquely simple.

3.1 CONTROLLING A SECOND-ORDER SINGLE-INPUT SYSTEM

For a general autonomous second-order single-input system in state space form:

$$\dot{x}_1 = f_1(x_1, x_2, u) \tag{3.1a}$$

$$\dot{x}_2 = f_2(x_1, x_2, u) \tag{3.1b}$$

The goal is to regulate this system to the origin. The following CLF is assumed:

$$V_1 = \frac{1}{2}{x_1}^2 + \frac{1}{2}{x_2}^2 \tag{3.2}$$

Differentiating with respect to time:

$$\dot{V}_1 = x_1\dot{x}_1 + x_2\dot{x}_2 \tag{3.3}$$

The time derivatives of the states are approximated with a first order Taylor approximation about $(x_1, x_2, u = 0)$. For accurate linearization, a small control effort is assumed:

$$\dot{x}_i \approx f_i(x_1, x_2, 0) + \left(\frac{\partial f_i}{\partial u}\right)_{(x_1, x_2, 0)} u, \ i \in (1,2) \tag{3.4}$$

Aside: Derivation of Equation 3.4
The multivariate Taylor expansion of a function

$$\dot{x}_i = f(x_1, x_2, u) \tag{3.5}$$

about a point $(x_1, x_2, u = 0)$[3] is:

$$\dot{x}_i \approx f(x_1, x_2, 0) + \left(\frac{\partial f}{\partial x_1}\right)_{(x_1, x_2, 0)} (x_1 - x_1) + \left(\frac{\partial f}{\partial x_2}\right)_{(x_1, x_2, 0)} (x_2 - x_2) + \left(\frac{\partial f}{\partial u}\right)_{(x_1, x_2, 0)} (u - 0) \tag{3.6}$$

Which reduces to Equation 3.4.

Substituting and rearranging:

$$\dot{V}_1 \approx x_1 f_1 + x_2 f_2 + \left[x_1 \frac{\partial f_1}{\partial u} + x_2 \frac{\partial f_2}{\partial u}\right] u \tag{3.7}$$

$$u \approx \frac{\dot{V}_1 - x_1 f_1 - x_2 f_2}{\left[x_1 \frac{\partial f_1}{\partial u} + x_2 \frac{\partial f_2}{\partial u}\right]} \tag{3.8}$$

[3] Subsequently in Chapter Three, the notation of the point where an expression is evaluated will be dropped for compactness and it can be assumed that expressions such as f_i and $\frac{\partial f_i}{\partial u}$ are evaluated at $(x_1, x_2, u = 0)$, unless otherwise noted.

By substituting desired $\dot{V}_{target}$ for $\dot{V}_1$, Equation 3.8 gives a method of calculating the control effort to stabilize the system at a user-defined rate. The terms $\frac{\partial f_1}{\partial u}$ and $\frac{\partial f_2}{\partial u}$ are calculated by numerically differentiating the model. In the special case of an input-affine system (i.e. $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})u$), Equation 3.8 describes the system dynamics exactly.

The control effort from Equation 3.8 can fail in two ways. If $\frac{\partial f_1}{\partial u} = 0$ and $\frac{\partial f_2}{\partial u} = 0$, then the system is locally weakly uncontrollable at **x** and u is unbounded. In this case, we can only hope the system will "drift" to a region where it becomes controllable again. On the other hand, if $\frac{\partial f_1}{\partial u}$ and/or $\frac{\partial f_2}{\partial u}$ but $\left[x_1 \frac{\partial f_1}{\partial u} + x_2 \frac{\partial f_2}{\partial u}\right]$ is zero at **x**, then $V_1(\boldsymbol{x})$ was not a suitable Lyapunov function. The zero arises from a local cancellation of terms for this particular linearization, but a relationship between *u* and *V* can be regained by modifying the CLF. For the majority of simulations in Chapter Four, the threshold for switching to the modified CLF was $\left|x_1 \frac{\partial f_1}{\partial u} + x_2 \frac{\partial f_2}{\partial u}\right| < 0.001$. The modification was achieved by adding a "step" to the function:

$$V_2 = V_1\{0.9 + 0.1 \left|\frac{x_1}{x_{1,step}}\right| \tanh[A]\} \tag{3.9}$$

$$A = \beta\left((x_1{}^2 + x_2{}^2) - \left(x_{1,step}{}^2 + x_{2,step}{}^2\right)\right) + \gamma \tag{3.10}$$

Where $\gamma > 0$ and $\boldsymbol{x_{step}} = (x_{1,step}, x_{2,step})$ is the coordinate where control of V_1 is lost. The "sharpness" of the step is set using β which must be greater than 0.

Note that $\boldsymbol{x_{step}}$ is regarded as a constant and V_2 varies with $\boldsymbol{x_{step}}$ in a pointwise fashion, so V_2 is technically a set of functions. Its key feature is that $sign(\dot{V}_2) = sign(\dot{V}_1)$

at $\boldsymbol{x}_{step}$. That feature is useful in the proof of stability which follows and it is visually apparent from the plot of V_2 for $\boldsymbol{x}_{step} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$ as shown in Figure 11.

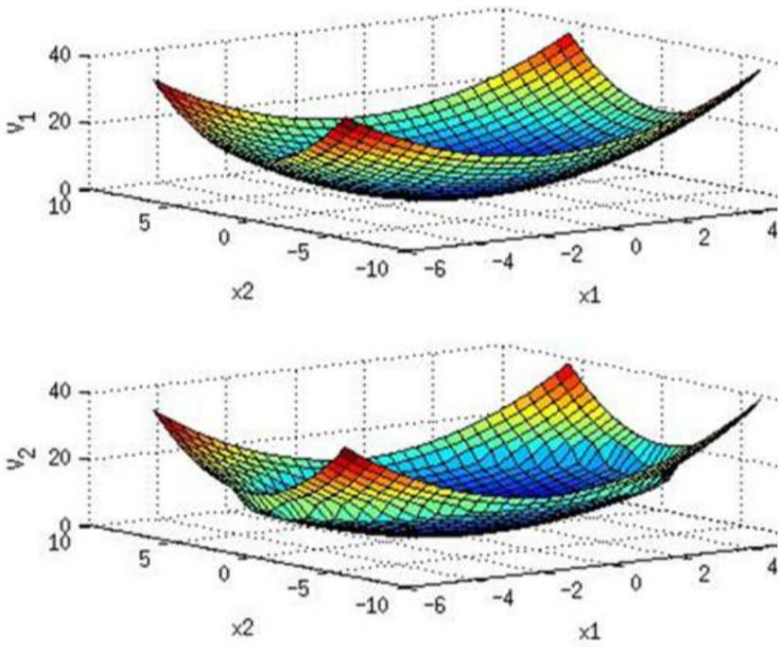


Figure 11. V_1 and V_2 with a step at (2,2) and γ=0.

Importantly, the step changes the relationship between $\dot{V}$ and u at $\boldsymbol{x}_{step}$ so that u has an effect (assuming $\frac{\partial f_1}{\partial u} \neq 0$ and $\frac{\partial f_2}{\partial u} \neq 0$). The time derivative of V_2 (linearized at $(x_1, x_2, u = 0)$) is:

$$\dot{V}_2 \approx \frac{\partial V_2}{\partial x_1} f_1 + \frac{\partial V_2}{\partial x_2} f_2 + \left(\frac{\partial V_2}{\partial x_1} \frac{\partial f_1}{\partial u} + \frac{\partial V_2}{\partial x_2} \frac{\partial f_2}{\partial u} \right) u \tag{3.11}$$

$$u \approx \frac{\dot{V}_2 - \frac{\partial V_2}{\partial x_1} f_1 - \frac{\partial V_2}{\partial x_2} f_2}{\left(\frac{\partial V_2}{\partial x_1} \frac{\partial f_1}{\partial u} + \frac{\partial V_2}{\partial x_2} \frac{\partial f_2}{\partial u} \right)} \tag{3.12}$$

Where

$$\frac{\partial V_2}{\partial x_1} = x_1\left\{0.9 + 0.1\left|\frac{x_1}{x_{1,step}}\right|\tanh[A]\right\} + 0.1V_1\left\{\frac{sign(x_1)}{x_{1,step}sign(x_{1,step})}\tanh[A] + 2\beta x_1\left|\frac{x_1}{x_{1,step}}\right| sech^2[A]\right\} \quad (3.13)$$

$$\frac{\partial V_2}{\partial x_2} = x_2\{0.9 + 0.1\left|\frac{x_1}{x_{1,step}}\right|\tanh[A] + 0.2\beta V_1\left|\frac{x_1}{x_{1,step}}\right| sech^2[A]\} \quad (3.14)$$

Similarly to Equation 3.8, Equation 3.12 can be used to calculate a control effort that will stabilize the system at a desired rate by substituting $\dot{V}_{target}$ for $\dot{V}_2$.

Switched Lyapunov Stability Corollary

The switched Lyapunov method builds upon the Lyapunov Theorem for Global Stability [Slotine and Li, 1991]. Thus we present a brief reproduction of the theorem followed by a corollary for a second order system with a switched Lyapunov function:

Theorem 1 (Global Stability for a Static Lyapunov function)

Assume there exists a scalar function $V(\boldsymbol{x})$ with continuous first partial derivatives such that

- $V(\boldsymbol{x})$ *is positive definite*
- $V(\boldsymbol{x}) \to \infty$ *as* $\|\boldsymbol{x}\| \to \infty$
- $\dot{V}(\boldsymbol{x})$ *is negative definite*

then the equilibrium at the origin is globally asymptotically stable.

Corollary 1 (Stability for a Second-Order, Single-Input Switched Lyapunov function)

Given a system described by $\dot{x}_i = f_i(x_1, x_2, u)$, $i \in (1,2)$, assume there exists a scalar function $V_1(\boldsymbol{x})$ with continuous first partial derivatives such that

- $V_1(\boldsymbol{x})$ *is positive definite*
- $V_1(\boldsymbol{x}) \to \infty$ *as* $\|\boldsymbol{x}\| \to \infty$

For accurate linearizations, assume that

- $\dot{\boldsymbol{x}}$ *has continuous first partial derivatives*
- *"Small" control efforts are sufficient to control the system.*

If there also exists a scalar function $V_2(\boldsymbol{x})$ *such that* $sign(\dot{V}_2) = sign(\dot{V}_1)$ *and*

- *at every point along the system's trajectory excluding the origin,* $\frac{\partial f_1}{\partial u} \neq 0$ *and* $\frac{\partial f_2}{\partial u} \neq 0$
- *at every point along the system's trajectory excluding the origin,* $\frac{\partial \dot{V}_1}{\partial u} \neq 0$ *and/or* $\frac{\partial \dot{V}_2}{\partial u} \neq 0$
- *if the linearization* $\frac{\partial \dot{V}_1}{\partial u}$ *is non-singular at a point, then* $\dot{V}_1(\boldsymbol{x})$ *is regulated to be negative*
- *if the linearization* $\frac{\partial \dot{V}_1}{\partial u}$ *is singular at a point, then* $\dot{V}_2(\boldsymbol{x})$ *is regulated to be negative*

then the equilibrium at the origin is globally asymptotically stable.

Proof of Corollary 1

If $\frac{\partial \dot{V}_1}{\partial u} \neq 0$ at every point, then the controller can reduce V_1 by Equation 3.8 and V_2 is never used. Hence $\dot{V}$ is negative definite and the system is asymptotically stable per Theorem 1.

If $\frac{\partial \dot{V}_1}{\partial u} = 0$ at any point then necessarily $\frac{\partial \dot{V}_2}{\partial u} \neq 0$ per the requirements of Corollary 1. Hence V_2 can be reduced by Equation 8 and $\dot{V}_2$ is negative. Per the $sign(\dot{V}_2) = sign(\dot{V}_1)$ requirement, $\dot{V}_1$ remains negative and the system remains asymptotically stabilized per Theorem 1.

Does the proposed algorithm satisfy Corollary 1?

Next, we prove that the previously described CLF's (V_1 and V_2) satisfy the requirements of Corollary 1. Clearly $V_1 = \frac{1}{2}{x_1}^2 + \frac{1}{2}{x_2}^2$ is continuous, radially unbounded, and positive definite. We can prove that $sign(\dot{V}_2) = sign(\dot{V}_1)$ using polar coordinates.

$$V_1 = \frac{r^2}{2} \tag{3.15}$$

$$\dot{V}_1 = r\frac{\partial r}{\partial t} \tag{3.16}$$

Since $r>0$,

$$sign(\dot{V}_1) = sign\left(\frac{\partial r}{\partial t}\right) \tag{3.17}$$

V_2 is applied when $\boldsymbol{x} = \begin{bmatrix} x_{1,step} \\ x_{2,step} \end{bmatrix}$ so Equation 3.9 reduces (in polar coordinates) to:

$$V_2 = \frac{r^2}{2}\{0.9 + 0.1\tanh(\gamma)\} \tag{3.18}$$

$$\dot{V}_2 = r\{0.9 + 0.1\tanh(\gamma)\}\frac{\partial r}{\partial t} \tag{3.19}$$

$$\therefore sign(\dot{V_2}) = \frac{\partial r}{\partial t} = \text{sign}(\dot{V_1}) \tag{3.20}$$

Is the linearization $\frac{\partial \dot{V}_1}{\partial u} \neq 0$ and/or the linearization $\frac{\partial \dot{V}_2}{\partial u} \neq 0$ at every point (excluding the origin)? We will describe a simple strategy to ensure $\frac{\partial \dot{V}_2}{\partial u} \neq 0$. The approximation of $\frac{\partial \dot{V}_2}{\partial u}$ is:

$$\frac{\partial \dot{V}_2}{\partial u} \approx \sum_{i=1}^{n} \frac{\partial V_2}{\partial x_i} \frac{\partial f_i}{\partial u} \tag{3.21}$$

For small γ (which is a user-specified parameter), the partial derivatives reduce to:

$$\frac{\partial V_2}{\partial x_1} \approx x_1(0.9 + 0.2\beta V_1 x_1) \tag{3.22}$$

$$\frac{\partial V_2}{\partial x_2} \approx x_2(0.9 + 0.2\beta V_1) \tag{3.23}$$

The expanded form of $\frac{\partial \dot{V}_2}{\partial u}$ is:

$$\frac{\partial \dot{V}_2}{\partial u} \approx x_1(0.9 + 0.2\beta V_1 x_1) \frac{\partial f_1}{\partial u} + x_2(0.9 + 0.2\beta V_1) \frac{\partial f_2}{\partial u} \tag{3.24}$$

$\boldsymbol{x} = \mathbf{0}$ is not a solution of interest to $\frac{\partial \dot{V}_2}{\partial u} = 0$ because it is the setpoint. A requirement of Corollary 1 is $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1,2\}$, so $\frac{\partial \boldsymbol{f}}{\partial u} = 0$ is not a solution of interest. The other possibility is a cancellation of terms for which there is only one β which satisfies $\frac{\partial \dot{V}_2}{\partial u} = 0$. There is only one β which satisfies $\frac{\partial \dot{V}_2}{\partial u} = 0$. The solution is:

$$\beta = \frac{-18\left(\frac{\partial f_1}{\partial u} x_1 + \frac{\partial f_2}{\partial u} x_2\right)}{V_1\left(\frac{\partial f_1}{\partial u} x_1^2 + \frac{\partial f_2}{\partial u} x_2\right)} \tag{3.25}$$

Thus, a practical strategy to avoid any chance of $\frac{\partial \dot{V}_2}{\partial u} = 0$ given $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1,2,\dots n\}$ is to choose a small γ and switch to a third Lyapunov function with a different β parameter whenever $\frac{\partial \dot{V}_2}{\partial u}$ is approximately zero. In all other ways, V_3 can be identical to V_2.

The final requirement of Corollary 1 is $\left[\frac{\partial f_1}{\partial u} \neq 0 \text{ and } \frac{\partial f_2}{\partial u} \neq 0 \right]$ which can be verified easily for each system. This concludes the proof.

Discussion of the proof

An assumption of Corollary 1 is $\left[\frac{\partial f_1}{\partial u} \neq 0 \text{ and } \frac{\partial f_2}{\partial u} \neq 0 \right]$, which was only included to facilitate the proof. It is related to the controllability of the system. Unfortunately, it restricts the applicability of Corollary 1 as shown in the Venn diagram of Figure 12. The rather small subset of accessible systems for which this proof does not apply:

- have open-loop "drift" dynamics that are orthogonal to the motion caused by the control effort and
- have a single non-zero $\frac{\partial f_1}{\partial u}$ or $\frac{\partial f_2}{\partial u}$ term.

Simulation 10: Second-Order PID Controller is an example of an accessible system where the switched Lyapunov algorithm fails. That type of higher-order integrator with a single control input is a particular weakness of the switched Lyapunov controller. The state-space equations of the system are:

$$\dot{x}_1 = x_2 \tag{3.31a}$$

$$\dot{x}_2 = \frac{1}{2}(-4x_2 - 3x_1 + u) \tag{3.31b}$$

And the switched Lyapunov controller fails, as we've described, because the system is accessible but $\frac{\partial f_1}{\partial u} = 0$. Section 7.6 discusses a modification to the algorithm that overcomes this limitation.

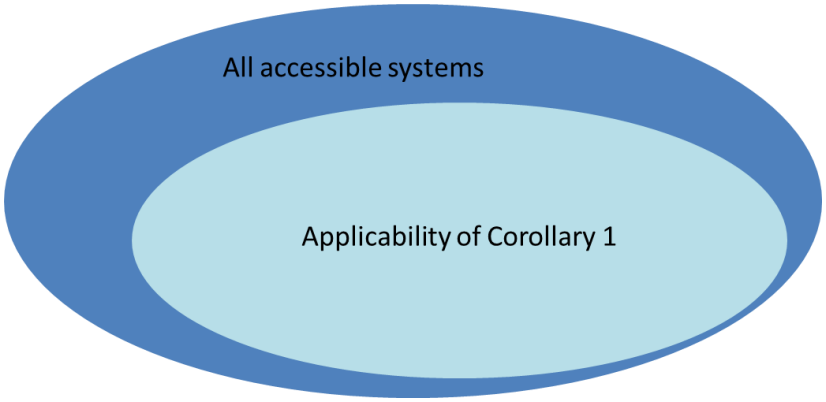


Figure 12. The proof does not cover a small subset of accessible systems.

Implications of linearizations in the derivation

Corollary 1 requires the state-space equations defining the system to have continuous first partials so the linearizations used to develop Equations (3.4-3.12) are accurate. For discrete implementations, it is implied that the simulation time step is small enough that the linearization accurately represents the system until the next time step. This implies that the system remains close to the switchpoint for any time step. Generally this is not an issue for simulations on modern computers because the time step can be arbitrarily small. It might be an issue for some hardware applications but not for the low-force, high-precision robotic contact tasks that were the early motivation for this research focus [Zelenak, 2015]. Sections 7.3 and 7.4 investigate the specifics of linearization in more detail.

Since the linearizations were developed around $u=0$, they may not be accurate for large control efforts. So, an assumption that the control efforts remain small is also required for the proof. Again, the motivating application allowed for this assumption.

Since linearization was used at each time step, the stability proof does not technically apply in a global sense. Rather, there is a region of attraction at each time step. If the system does not drift out of that region of attraction before the next time step, it will be asymptotically stabilized. The practical implication is that the controller should run at the highest possible frequency. If it fails to stabilize a system, the possible causes are:

- The control efforts were too large for accurate linearization about u=0, or
- The control frequency wasn't high enough for accurate linearization, or
- The system moves to a point where it is inaccesssible, or
- Equation 3.51 was not satisfied, or
- The control efforts were saturated.

The switched Lyapunov controller may not be a good candidate for stabilizing systems where large control efforts are expected.

A note on the semantics of Lyapunov functions

Upon peer review, the naming of V_1 has been a point of discussion. Is it a *candidate* or a verified Lyapunov function? Slotine and Li's [Slotine and Li, 1991] definition of a Lyapunov function follows:

If, in a ball B_{R_0}, the function V($\boldsymbol{x}$) is positive definite and has continuous partial derivatives, and if its time derivative along any state trajectory is negative semi-definite, i.e. $\dot{V}(\boldsymbol{x}) \leq 0$, then V(x) is said to be a Lyapunov function.

We assert that V_1 meets the definition and should be labeled a Lyapunov function if it meets the requirements of Corollary 1 since that ensures $V_1 \leq 0$. Otherwise, it would be labeled a candidate Lyapunov function. Since V_2 is also positive definite and $sign(\dot{V}_2)$= $sign(\dot{V}_1)$, V_2 should be labelled the same as V_1. We further assert that V_1 and V_2 are control Lyapunov functions [CLF's] since they are used for controller synthesis.

Arbitrary setpoints

Until this point, we have only considered stabilization at the origin. However, it is a common desire to stabilize a system at nonzero setpoints. Fortunately, the algorithm can be applied to arbitrary setpoints after implementing a coordinate frame translation. For example, controlling $\dot{x} = x - 2$ to the origin is equivalent to controlling $\dot{x} = x$ to $x = 2$. Thus, there is no loss of generality by basing the analysis on a setpoint at the origin. This topic is explored in more rigor by [Khalil, 1996, pg. 132].

Technically we have offered no proof that the algorithm can asymptotically stabilize non-autonomous systems nor that it can be applied to tracking problems, i.e. problems where the setpoint changes with time. Barbalat's Lemma can be used as a tool in such proofs, but it is generally applied on a system-by-system basis and does not seem helpful for a general proof. [Slotine and Li, 1991, pg. 125] Regardless, in our extensive testing the algorithm has performed quite well for time-varying problems (including non-autonomous systems and tracking problems). Refer to Chapter Four for more details on those simulations.

3.2 EXTENSION TO SYSTEMS OF ANY ORDER AND ANY NUMBER OF INPUTS

In this section, the method is extended to systems of arbitrary order and arbitrary number of inputs. A proof is presented for systems of arbitrary order but we were unsuccessful in formulating a proof for the most general case (arbitrary order and arbitrary number of inputs). However, we discuss why a stability proof is less important for such large systems.

Systems of any order

The key equations that must be rewritten for n^{th} order systems are Equations 3.8 and 3.12. The general state-space equation is:

$$\dot{\mathrm{x}}_i = \mathrm{f}_i(\mathbf{x}, \mathrm{u}), \quad i \in \{1, \dots, n\} \tag{3.32}$$

The second CLF V_2 is used when $\sum_{i=1}^{n} \mathrm{x_i} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}} \approx 0$, i.e. when the denominator of Equation 3.8 is nearly zero.

For the Default CLF

$$\mathrm{V}_1 = \frac{1}{2}\boldsymbol{x} \cdot \boldsymbol{x} \tag{3.33}$$

$$\mathrm{u} \approx \frac{\dot{\mathrm{V}}_{\mathrm{target}} - \sum_{i=1}^{n} \mathrm{x_i f_i}}{\sum_{i=1}^{n} \mathrm{x_i} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}}} \tag{3.34}$$

For the Second CLF

$$\mathrm{V}_2 = (\frac{1}{2}\boldsymbol{x} \cdot \boldsymbol{x}) * \left\{0.9 + 0.1 \left|\frac{x_1}{x_{1,step}}\right| \tanh\left[\beta\left(\boldsymbol{x} \cdot \boldsymbol{x} - \boldsymbol{x}_{step} \cdot \boldsymbol{x}_{step}\right) + \gamma\right]\right\} \tag{3.35}$$

Again, $\boldsymbol{x}_{step}$ is regarded as a constant and V_2 varies with $\boldsymbol{x}_{step}$ in a pointwise fashion, so it is technically a set of functions. The linearization of $\mathrm{V}_2(\boldsymbol{x}, \boldsymbol{x}_{step})$ is lengthy so we refer hereafter to a general formula for the linearization of V_1 or V_2:

General Form for Either CLF

$$\dot{\mathrm{V}} \approx \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i} + \left[\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}}\right] u \tag{3.36}$$

$$\mathrm{u} \approx \frac{\dot{\mathrm{V}}_{\mathrm{target}} - \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i}}{\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}}} \tag{3.37}$$

Corollary 2 (Stability for an nth-order, single-input switched Lyapunov function)

Given a system described by $\dot{x}_i = f_i(x_1, x_2, \dots, x_n, u),\ i \in (1,2,\dots,n)$, *assume there exists a scalar function* $V_1(\boldsymbol{x})$ *with continuous first partial derivatives such that*

- $V_1(\boldsymbol{x})$ *is positive definite*
- $V_1(\boldsymbol{x}) \to \infty$ *as* $\|\boldsymbol{x}\| \to \infty$

For accurate linearizations, assume that

- $\dot{\boldsymbol{x}}$ *has continuous first partial derivatives*
- *"Small" control efforts are sufficient to control the system.*

If there also exists a scalar function $V_2(\boldsymbol{x})$ *such that* $sign(\dot{V}_2) = sign(\dot{V}_1)$ *and*

- *at every point along the system's trajectory excluding the origin,* $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1 \dots n\}$
- *at every point along the system's trajectory excluding the origin,* $\frac{\partial \dot{V}_1}{\partial u} \neq 0$ *and/or* $\frac{\partial \dot{V}_2}{\partial u} \neq 0$
- *if the linearization* $\frac{\partial \dot{V}_1}{\partial u}$ *is non-singular at a point, then* $\dot{V}_1(\boldsymbol{x})$ *is regulated to be negative*
- *if* $\frac{\partial \dot{V}_1}{\partial u}$ *is singular at a point, then* $\dot{V}_2(\boldsymbol{x})$ *is regulated to be negative*

then the equilibrium at the origin is globally asymptotically stable.

Proof of Corollary 2

The proof of Corollary 2 is identical to the proof of Corollary 1.

Does the proposed algorithm satisfy Corollary 2?

Next, we prove that the previously described CLF's (V_1 and V_2) satisfy the requirements of Corollary 2. Clearly V_1 (Equation 3.33) is continuous, radially unbounded, and positive definite. We can prove that $sign(\dot{V}_2)$= $sign(\dot{V}_1)$ using "hyperspherical" coordinates—the higher-dimensional generalization of spherical coordinates.

$$V_1 = \frac{r^2}{2} \tag{3.38}$$

$$\dot{V}_1 = r\frac{\partial r}{\partial t} \tag{3.39}$$

Since *r>0*,

$$sign(\dot{V}_1) = sign\left(\frac{\partial r}{\partial t}\right) \tag{3.40}$$

V_2 is applied when $\boldsymbol{x} = \begin{bmatrix} x_{1,step} \\ x_{2,step} \end{bmatrix}$ so Equation 3.35 reduces to:

$$V_2 = \frac{r^2}{2}\{0.9 + 0.1\tanh(\gamma)\} \tag{3.41}$$

$$\dot{V}_2 = r\{0.9 + 0.1\tanh(\gamma)\}\frac{\partial r}{\partial t} \tag{3.42}$$

$$\therefore sign(\dot{V_2}) = \frac{\partial r}{\partial t} = \text{sign}(\dot{V_1}) \tag{3.43}$$

Is the linearization $\frac{\partial \dot{V}_1}{\partial u} \neq 0$ and/or the linearization $\frac{\partial \dot{V}_2}{\partial u} \neq 0$ at every point (excluding the origin)? We will describe a simple strategy to ensure $\frac{\partial \dot{V}_2}{\partial u} \neq 0$. The approximation of $\frac{\partial \dot{V}_2}{\partial u}$ is:

$$\frac{\partial \dot{V}_2}{\partial u} \approx \sum_{i=1}^{n} \frac{\partial V_2}{\partial x_i} \frac{\partial f_i}{\partial u} \tag{3.44}$$

For small γ (which is a user-specified parameter), the partial derivatives reduce to:

$$\frac{\partial V_2}{\partial x_1} \approx x_1(0.9 + 0.2\beta V_1 x_1) \tag{3.45}$$

$$\frac{\partial V_2}{\partial x_{i\neq 1}} \approx x_i(0.9 + 0.2\beta V_1) \tag{3.46}$$

The expanded form of $\frac{\partial \dot{V}_2}{\partial u}$ is:

$$\frac{\partial \dot{V}_2}{\partial u} \approx x_1(0.9 + 0.2\beta V_1 x_1)\frac{\partial f_1}{\partial u} + \sum_{i=2}^{n} x_i(0.9 + 0.2\beta V_1)\frac{\partial f_i}{\partial u} \tag{3.47}$$

$\boldsymbol{x} = \mathbf{0}$ is not a solution of interest to $\frac{\partial \dot{V}_2}{\partial u} = 0$ because it is the setpoint. A requirement of Corollary 2 is $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1 \dots n\}$, so $\frac{\partial \boldsymbol{f}}{\partial u} = 0$ is not a solution of interest. The other possibility is a cancellation of terms for which there is only one β which satisfies $\frac{\partial \dot{V}_2}{\partial u} = 0$. For first-order systems, the solution is:

$$\beta = \frac{-9}{x_1^3} \tag{3.48}$$

And for higher-order systems:

$$\beta = \frac{-18\sum_{i=1}^{n} \frac{\partial f_i}{\partial u} x_i}{V_1\left(\frac{\partial f_1}{\partial u} x_1^2 + \sum_{i=2}^{n} \frac{\partial f_i}{\partial u} x_i\right)} \tag{3.49}$$

Thus, a practical strategy to avoid any chance of $\frac{\partial \dot{V}_2}{\partial u} = 0$ given $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1,2, \dots n\}$ is to choose a small γ and switch to a third Lyapunov function with a different β parameter whenever $\frac{\partial \dot{V}_2}{\partial u}$ is approximately zero. In all other ways, V_3 can be identical to V_2.

The final requirement of Corollary 2 is $\frac{\partial f_i}{\partial u} \neq 0, i\epsilon\{1 \dots n\}$ which can be quickly verified for each system. This concludes the proof.

The first-order case

The first-order case is unique because switching to a second CLF does not alleviate the singularity type of error that was discussed for Equation 3.8. It is as if a first-order system is too simple to require a function. However, the second CLF does not cause any harm, and the formula that was given for systems of arbitrary order (Equation 3.37) remains valid. A brief explanation of the trivial first-order case follows and several simulations of first-order systems in Chapter Four validate that the second CLF does not affect performance in a significant way.

For an autonomous, first-order system of the form:

$$\dot{x}_1 = f_1(x_1, u) \tag{3.54}$$

And a default CLF:

$$V_1 = \frac{1}{2}{x_1}^2 \tag{3.55}$$

The linearization of $\dot{x}_1$ about $(x_1, u = 0)$ is:

$$\dot{x}_1 = f_1(x_1, 0) + \frac{\partial f_1}{\partial u}u \tag{3.56}$$

And the approximation for $\dot{V}_1$:

$$\dot{V}_1 \approx x_1 f_1(x_1, 0) + x_1 \frac{\partial f_1}{\partial u} u \tag{3.57}$$

Given a desired $\dot{V}_1$, the necessary control effort u can be calculated:

$$u \approx \frac{\dot{V}_1 - x_1 f_1(x_1, 0)}{x_1 \frac{\partial f_1}{\partial u}} \tag{3.58}$$

Equation 3.58 is singular when $x_1 = 0$ (not of interest because the origin is the setpoint) or $\frac{\partial f_1}{\partial u} = 0$. Unlike the higher-order systems, a different CLF does not alleviate the singularities. The control effort as calculated from linearization of the second CLF is:

$$u \approx \frac{\dot{V}_2 - \frac{\partial V_2}{\partial x_1} f_1(x_1, 0)}{\frac{\partial V_2}{\partial x_1} \frac{\partial f_1}{\partial u}} \tag{3.59}$$

The denominator of Equation 3.59 causes singularities in the same circumstances as Equation 3.8 because $\frac{\partial V_2}{\partial x_1}$ evaluated at $x_1 = 0$ is zero and the $\frac{\partial f_1}{\partial u}$ term is identical. So it does not resolve the singularity in Equation 3.8. Again, the second CLF does no harm but it also provides no benefit for first order systems.

Systems with more than one input

When there is more than one input to the system, the question arises: how should the control efforts be distributed to achieve the desired $\dot{V}_{target}$?

For a system with *m* inputs and *n* states, a naïve approach would be to focus all effort on the input u_{max} associated with the greatest effect on *V:*

$$\left\{ u_{max} \middle| \left| \frac{\partial V}{\partial u_{max}} \right| = \max \left| \frac{\partial V}{\partial u_i} \right|, i \in \{1, \ldots, m\} \right\} \tag{3.60}$$

and set all other inputs to zero. (We refer to u_{max} as the *dominant control effort*.) While effective, this approach would have undesirable effects for practical applications. Consider a car where only the most-effective tire can be driven at any instant. There would be severe chatter as the various wheels switch on and off and the resultant stress on the drivetrain would be higher, etc. A different method was devised to distribute the effort across all inputs and achieve the desired $\dot{V}_{target}$.

A modification to Equation 3.37 that incorporates *m* inputs is:

$$\dot{V} \approx \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i + \sum_{j=1}^{m} \left[\sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\partial f_i}{\partial u_j} \right] u_j \tag{3.61}$$

Equation 3.61 is the most general expression for the required linearization because it allows for systems of any order (n^{th} order) and with any number of inputs (*m* inputs). Define the following variables for compactness:

$$P \equiv \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i} \tag{3.62}$$

$$D_j \equiv \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}_j} \tag{3.63}$$

$$D_{max} \equiv \max D_j \tag{3.64}$$

Then Equation 3.61 can be rewritten:

$$\dot{\mathrm{V}} \approx P + \sum_{j=1}^{m} D_j u_j \tag{3.65}$$

Scale each control effort according to u_{max}, the most effective control effort:

$$u_i = u_{max} * \frac{D_i}{D_{max}} \tag{3.66}$$

Substituting Equation 3.66 into Equation 3.65 and rearranging:

$$u_{max} = \frac{D_{max}(\dot{V}-P)}{D_1^2+D_2^2+\cdots+D_m^2} \tag{3.67}$$

Equation 3.67 gives a method of calculating the largest control effort, which is the input that will have the most rapid, direct effect on V. The other control efforts will be proportionally less in magnitude and they can be calculated with Equation 3.65. The end result is a distributed control effort across all inputs that produces the desired $\dot{V}_{target}$.

Discussion of a proof for arbitrary order and arbitrary inputs

A proof of asymptotic stability for more than one input seems algebraically challenging. As with the simpler proofs presented above, the crux is proving that $\left[\frac{\partial V_1}{\partial \overline{\boldsymbol{u}}} = \mathbf{0}\right]$ and $\left[\frac{\partial V_2}{\partial \overline{\boldsymbol{u}}} = \mathbf{0}\right]$ cannot occur at the same instant (where $\frac{\partial V_1}{\partial \overline{\boldsymbol{u}}}$ and $\frac{\partial V_2}{\partial \overline{\boldsymbol{u}}}$ are r-vectors). Fortunately, the probability of inaccessibility decreases as more control inputs are added to the system, i.e. as r increases. From an intuitive standpoint, it would seem unlikely for all $\frac{\partial V_1}{\partial u_i}$, $i \epsilon r$ partial derivatives to be zero for a long control input vector, and even more unusual for $\frac{\partial V_2}{\partial \overline{\boldsymbol{u}}} = \mathbf{0}$ simultaneously. Indeed, failure seems much more likely on lower-order, underactuated systems. Our experience with simulations supports that conclusion.

The same insight can be gleaned from an examination of the controllability matrix for input-affine nonlinear systems. That matrix was defined in Chapter Two:

$$C = \left[g_1 \ldots g_m \left[g_i, g_j\right] \left[ad_{g_i}^k g_j\right] \ldots [f, g_i] \ldots \left[ad_{f_i}^k g_i\right]\right] \tag{3.68}$$

More inputs to the system increase the number of g terms in Equation 3.68, adding more columns and increasing the likelihood that all rows are linearly

independent. Thus, we argue, a proof for more inputs is actually less significant than the proof for a single input.

An approach that would allow the single-input proof to be applied on multiple-input systems is to check accessibility for one input at a time. An example for a two-input system:

- Check for u_1 only, using Equation 3.51 and setting u_2 to zero.
- At the points where u_1 fails to provide control, check for u_2, setting u_1 to zero and using Equation 3.51 again.

This is a "sufficient but not necessary" approach, i.e. it will never yield a false positive for accessibility but it may yield false negatives. There is a small chance that an interaction between u_1 and u_2 achieves accessibility and this method will not predict it, e.g. a multiplicative term such as $u_1 u_2$. As more inputs are added to a system, the likelihood of a false negative decreases.

Possible forms of the second CLF

There are many forms of V_2 that are capable of satisfying Corollary 2. Chapter 7 "A strategy for systems with relative degree greater than one" proposes a V_2 based on the absolute value of a transformed state variable:

$$V_2(\boldsymbol{\xi}) = V_1\left[0.9 + 0.1\left|\xi_r - 1\right|\right] \tag{3.69}$$

The important features of V_2 are:

- V_2 has symmetry about the origin so that $sign(\dot{V}_2) = sign(\dot{V}_1)$. Radial symmetry is one type of symmetry that facilitates this proof, but there are other options.
    - Since $V_1(\boldsymbol{x})$ is centered about the origin and positive definite, this implies that $V_2(\boldsymbol{x})$ is also centered about the origin and positive definite.
- Additional terms appear in the partial derivative $\frac{\partial \dot{V}_2}{\partial u}$ so that $\frac{\partial \dot{V}_2}{\partial u} \neq \frac{\partial \dot{V}_1}{\partial u} = 0$. In words, $\frac{\partial \dot{V}_2}{\partial u}$ must avoid some of the terms that cause singularities for $\frac{\partial \dot{V}_1}{\partial u}$ in order to be useful.

Convergence rate

Since $\dot{V}_{\text{target}}$ is specified by the user, it can be integrated to estimate the system's time to convergence on the setpoint. This is only an estimate because:

- Over the course of a simulation, the active Lyapunov function may switch between V_1 and V_2, and $\dot{V}_{\text{target}}$ applies to the active Lyapunov function. So, $\dot{V}_{\text{target}}$ is not a measure of change for the same function if switching occurs. However, the difference between $\dot{V}_1$ and $\dot{V}_2$ is typically small, so this effect will be small. For confirmation of that claim, refer to Figure 11.
- The system may not track the desired $\dot{V}_{\text{target}}$ target perfectly due to insufficiently small time steps, modeling error, etc.

Assuming these two effects are negligible, an estimate of the Lyapunov value at any time is:

$$V_1(\mathrm{t}) = V_1(0) + \int_{\tau_0}^{\tau_f} \dot{V}_{target} d\tau \tag{3.70}$$

This is a convenient formula because it allows the user to specify a $\dot{V}_{target}$ to match his performance specifications.

3.3 CHAPTER SUMMARY

This concludes Chapter Three, which described the switched Lyapunov control algorithm starting from simple systems and progressing to the most general case. The next chapter becomes more practical as it applies the algorithm to a wide array of dynamic simulations, with the ultimate goal of validating the algorithm.

Chapter 4: MATLAB Simulator

In this chapter, the MATLAB simulation software that is based on the switched Lyapunov control algorithm is explained. The software, "Lyapunov Nonlinear Control GUI," is freely available to the public at MATLAB's file exchange website [Zelenak, 2014] or directly from the GitHub code repository. The initial public reception is promising; since it was uploaded on November 2014, the package has been downloaded between 10-100 times monthly and has limited but extremely positive feedback with two 5-star ratings (Figure 13).

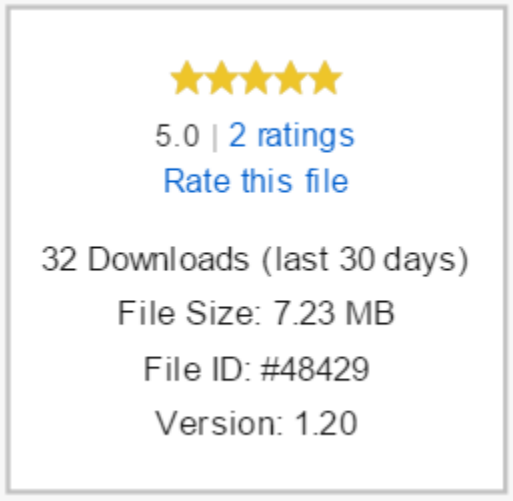


Figure 13. Statistics for the MATLAB GUI (February 12, 2016).

After the explanation of the software, the chapter concludes with the results of simulations used to discuss and validate the algorithm developed in Chapter 3.

4.1 Description of the Simulator

The MATLAB software is discussed in two parts. First is the interface where the user enters data, sends commands to the program, and sees plots of the simulation results

(i.e. the front end). Second is the code which runs behind the scenes and performs the calculations (i.e. the back end). Typically the back end would not be seen by the user (although anybody can investigate the source code.)

User Interface

The front end is a Graphical User Interface (GUI), which means that the user interacts using a mouse instead of entering text on a command line. GUI's have a much shallower learning curve than command-line interfaces, so it was an easy choice. The GUI was created with MATLAB's guide toolset (GUI Development Environment), which is easy to use and one of the main motivators for selecting the MATLAB environment. The GUI is shown in Figure 14.

Most of the data-entry fields are self-explanatory but there are helpful hints found by clicking on the blue question marks. The state-space equations that define the model and the plant are entered as separate text files which are then selected through the GUI. When the software is downloaded, it includes examples of 15 systems so that the user can become familiar with the required format. The software is set up to run a simulation of seven motors directly after it is downloaded. Notice the "Input Saturation" field; this field allows the designer to specify limits on the control efforts, such as maximum/minimum motor torques.

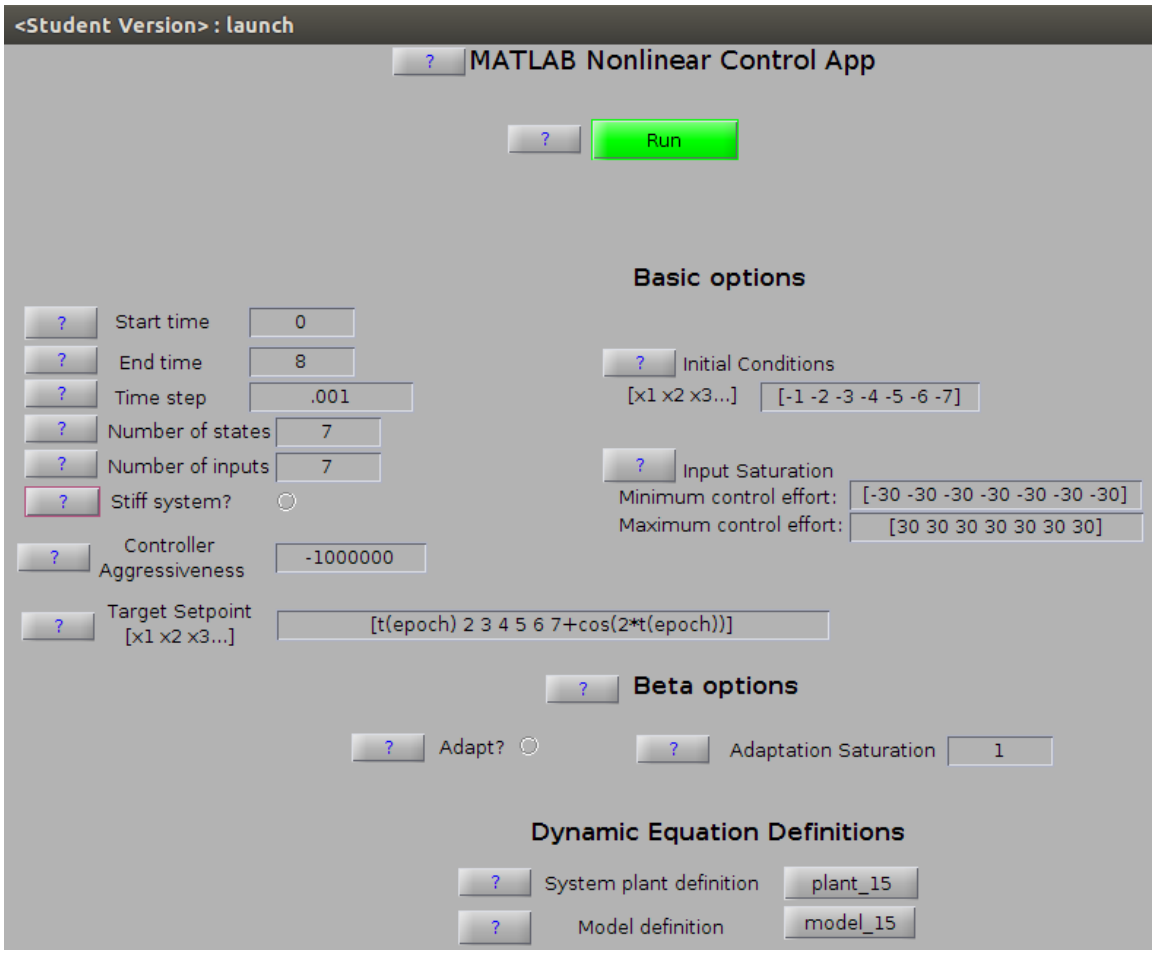


Figure 14. Graphical user interface of the MATLAB nonlinear control software.

In Figure 14 there is a section for "Beta Options," which includes an adaptation option. The adaptation option is not covered in this dissertation because it has not been thoroughly analyzed or tested, which is a topic for future consideration. However, the basic premise is that proportional feedback is applied to correct the partial derivative calculations if the system does not move in the intended direction. This could be useful, for instance, if there are modeling errors or if the plant might change over time.

Limitations of the GUI

When the simulation concludes, the program saves a data file that includes the state of the system and the control efforts that were calculated at every time step. This data is useful if the designer wants to examine the simulation results in detail. A few plots are also generated which help with debugging or tuning the controller. MATLAB has built-in memory constraints that limit the size of the data arrays that can be stored. On the 32-bit Ubuntu platform, arrays are limited to 312 million values long or 1.6 GB in total size [MathWorks Support Team, 2013]. That memory constraint limits the product of (simulation duration)*(time step). When a very small time step is used on a high-order system, it is easy to exceed MATLAB's memory limits. This was motivation for the switch to C++ for the high-performance implementation. While it is possible in the future to reduce the number of data points for future analysis to a subset of those generated, this feature was not developed here since the primary focus is the development and evaluation of the control algorithm itself.

Another factor that limits the size/complexity of the systems that can be controlled with the MATLAB simulator is the size of the text boxes in the GUI. Plotting is limited to seven states and, in general, the GUI becomes cumbersome to use if the system is higher than about fifth-order. Again, the capabilities could be extended by using a data file for larger systems, but this was not necessary given the availability of a C++ implementation.

Algorithm implementation

After the user clicks to start the simulation, the back end begins a simulation of the open-loop system. The plot from the open-loop simulation is useful for the user to check if

the system dynamics were entered correctly, as well as for comparing against the closed-loop performance. When the open-loop simulation has been completed, several events occur in rapid succession:

- A timer starts. The timer is used to check whether the closed-loop simulation is real-time or not. On all but the most simple first-order systems, the MATLAB simulator runs slower than real time.
- Data collection begins. With every time step, the state of the system and the calculated control efforts are logged for future analysis.
- Simulation of the dynamic system begins. At every time step, the control efforts are taken into account and an ordinary differential equation solver integrates the state-space equations of the system to calculate its state at the next time step to a high degree of accuracy.
- The switched Lyapunov control calculations occur, processing the present state of the plant and $\dot{V}_{target}$ to calculate a stabilizing control effort for the next time step. The equations of interest are 3.65 and 3.66. These are the formulas for the most general linearization, valid for systems of any order and any number of inputs.

When the simulation time reaches the end point that was defined by the user, the simulation stops and the relevant data is plotted for analysis.

MATLAB Differential Equation Solver

Integration of the state-space equations that define the system are achieved with MATLAB's ode23 ordinary differential equation solver. In an informal test, ode23 was slightly faster than the other common solvers (ode45 and ode113). However, in all cases, the solver is the bottleneck. To improve the performance as much as possible, the code requires the state-space equations to be vectorized, which means the. solver can input and solve for many points simultaneously. Vectorized code also takes advantage of multiple processor cores whereas loop-based code cannot. A blogger who tested vectorized MATLAB code versus for-loop-based code found that the vectorized code ran 43% faster [Lecoq, 2012]. The creators of MATLAB only claim that, "Vectorized code often runs much faster than the corresponding code containing loops." [MathWorks, 2015] Note that the selection of MATLAB ode solvers is limited to adaptive-step solvers [MathWorks, 2013, "Is there a fixed-step Ordinary Differential Equation solver in MATLAB 8.0?"].

The downside of vectorization is that specification of the state-space equations by the user is more difficult. For example, the vectorized version of $\dot{x} = x + u$ would be entered as:

$$\dot{x} = x(1,:) + u(1,:) \tag{4.1}$$

Where the colon notation signifies that many lines of data can be input simultaneously. Despite the steeper learning curves for users, it was decided that the significant performance improvement made vectorization worthwhile.

The GUI allows the user to specify whether the state-space equations are stiff or not. Stiff systems are slow to solve with ode23 so, if the user specifies a stiff system,

the program uses MATLAB's specialized ode23s solver. Indications that a system of ODE's may be stiff include:

- There is a mix of large and relatively small coefficients in the state-space equations.
- Some states vary much more rapidly than others.
- Running the simulator without the "stiff" option selected is much slower.

4.2 SIMULATIONS

This section details the many simulations that were conducted to validate the switched Lyapunov controller. The objective was to categorize the controller's performance over a wide range of systems, from very simple, low-order systems to complex, coupled, high-order systems. Under- and over-actuated systems are included. The GUI also includes a simulation that benchmarks the switched Lyapunov controller's performance against an industry-standard PID controller.

Practical considerations

Some modifications were applied to ensure the simulations were numerically tractable. To model saturation, control efforts were capped. The derivative of the Lyapunov function, $\dot{V}_{target}$, was tapered as the system approaches its setpoint; otherwise a tremendous control effort is required to maintain a constant $\dot{V}_{target}$ when V is small. With a finite simulation time step, the large control effort can cause overshoot. So $\dot{V}_{target}$ was tapered as follows:

$$\dot{V}_{target}(t) = \dot{V}_{target}(0) * \left(\frac{V(t)}{V(0)}\right)^2 \tag{4.2}$$

Some of the earliest users of the switched Lyapunov controller felt it was confusing to specify very large negative values for $\dot{V}_{target}(0)$, so the programs were adapted to convert the user-specified $\dot{V}_{target}(0)$ input from decibel units, i.e. $\dot{V}_{target}(0) = -10^{\left(\frac{\dot{V}_{target,dB}(0)}{20}\right)}$. This is more intuitive, for example, when a $\dot{V}_{target}(0)$ of -10^6 can be entered as -120 dB.

Notes on tuning the controller for optimal performance

It was clear from the derivation in Chapter Three that the switched Lyapunov controller is heavily dependent on linearization, and we discussed the need for a small time step to ensure that the linearizations are accurate. The default time step for most of the simulations is small relative to the period of interest (see e.g. Table 3) and uses even smaller time steps for larger or more complex systems. Of course, "small" is relative and it may depend on the coefficients, units, etc. of each particular system.

A very negative, aggressive value was applied as the default for $\dot{V}_{target}(0)$. This ensures that the system approaches its setpoint quickly. In our experience, it is rare for a system to overshoot the setpoint due to the tapering of $\dot{V}_{target}$ and the small time step, so there is little to no harm in selecting an aggressive $\dot{V}_{target}$. The potential downside to a large negative $\dot{V}_{target}$ value appears to be that it often saturates the inputs as it pushes the system rapidly.

Summary of simulations

- Simulation 1: First-Order RC Circuit

    - This system was chosen for the initial simulation because it is simple and practical.
- Simulation 2: First-Order Sensitivity Analysis
    - This system is nearly identical to Simulation 1. The difference is a systematic parameter study.
- Simulation 3: First-Order Robustness Analysis
    - Another first-order system, but the signs of the coefficients are reversed from Simulations ½ to demonstrate the flexibility of the controller.
- Simulation 4: Sensitivity Analysis of a Coupled Third-Order System
    - This highly-coupled system was contrived as a challenging yet controllable test for the control algorithm.
- Simulation 5: Tracking with Seven Motors
    - This example was inspired by the control of a seven degree-of-freedom robot.
- Simulation 6: Alternative Lyapunov Function Improves Performance
    - Simulation 6 continues with the same seven-motor state-space equations, but it focuses on how a second Lyapunov function improves the controller performance.
- Simulation 7: Comparison with McCourt's method

    - McCourt [2015] recently published a switched CLF nonlinear control algorithm which is similar to the presented method, so the two techniques are compared directly here.

- Simulation 8: van der Pol Oscillator
    - This was one of the first nonlinear systems ever studied in detail, as well as an interesting study of controllability, coupling, and limit cycles.
- Simulations 9-11: PID Comparisons
    - These simulations compared the proposed switched Lyapunov controller to industry-standard PID controllers on a range of systems from first- to third-order. Simulation 10 is a mass-spring-damper; the other two are contrived examples.
- Simulation 12: Very Large System
    - This system is covered in Chapter Five. Every other simulation was lumped into a massive, coupled nonlinear system to test the algorithm in an extreme fashion.

4.3 ASYMPTOTICALLY STABILIZED SYSTEMS

Simulation 1: First-order RC circuit

The first simulation is a simple first-order, linear system:

$$\dot{x} = \frac{1}{2}(u - x) \tag{4.3}$$

The system could represent many physical processes including the temperature change of a body (where *x* is temperature and *u* is heat flow into the system). It could also represent the RC circuit of Figure 15 where RC=$\frac{1}{2}$.

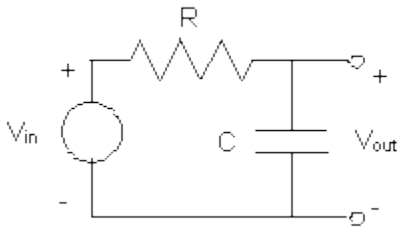


Figure 15. First-order RC circuit [Mastascusa].

$$\dot{V}_{out} = \frac{1}{RC}(V_{in} - V_{out}) \tag{4.4}$$

$$\downarrow$$

$$\dot{x} = \frac{1}{RC}(u - x) \tag{4.5}$$

The system is obviously controllable with a controllability matrix of $\left[-\frac{1}{2}\ \ \frac{1}{2}\right]$ (full row rank). The Lyapunov simulation parameters are given in Table 3 and it is a step response test. This simulation appeared insensitive as a wide range of simulation parameters yielded similar performance. (Its robustness is studied more formally in the next simulation.)

Table 3. Parameters of Simulation 1

Time step	2 ms
Control Effort Saturation	±10
Aggressiveness, $\dot{V}_{target}(0)$ [dB]	-120
V_2 step sharpness, β	0.5
γ	0.1
Switching threshold[4], $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$	0.001
Initial condition	0
Setpoint	1

Finally, the results of the PID simulation and the switched Lyapunov simulation are shown in Figure 16. Some remarks on the performance follow. For reference, there is a more formal comparison against PID controllers later in this chapter.

- Overshoot was negligible.
- The system settled within 0.3 seconds.
- The rise time was very fast.

If the second CLF had been applied, it would be visible in Figure 16. For this simulation, it was not necessary.

[4] Refer to Chapter Two, *Extension to Systems of Any Order and Any Number of Inputs*

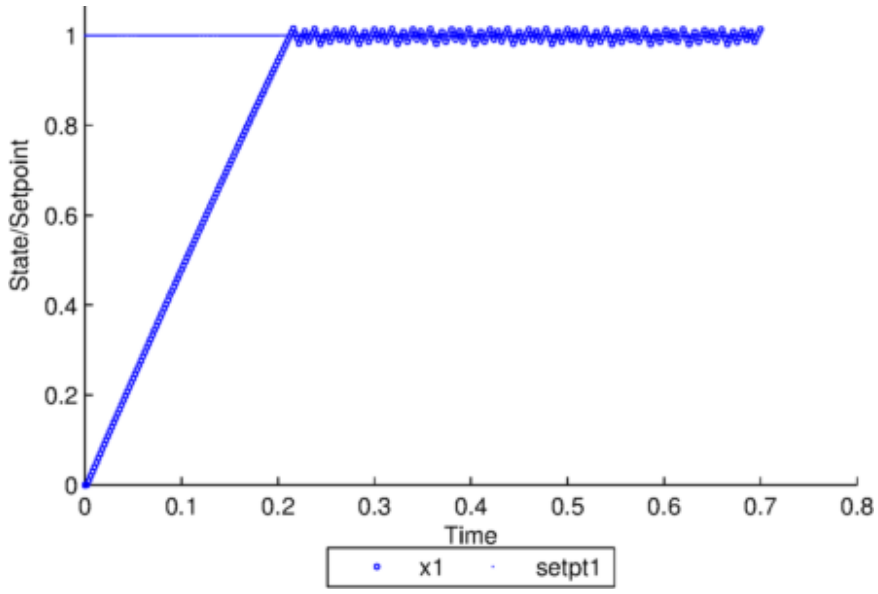


Figure 16. Regulation of a first-order RC circuit voltage. Note the limited overshoot and a bit of chatter about the setpoint.

The MATLAB program provides several additional plots that can be used to gain further insight into the algorithm. Figure 17 shows how the Lyapunov value approached zero exponentially over the course of the simulation. This is typical and it is caused by the tapering of $\dot{V}_{target}$. Figure 18 plots the control effort that was calculated and applied to the system. The drawbacks of the switched Lyapunov controller are that it tends to saturate and chatter the control inputs, similar to sliding mode control. In physical systems, chatter and saturation can produce excessive heat and higher cyclical stresses that lead to premature component failure. The chatter can be reduced with a less aggressive $\dot{V}_{\text{target}}$ and stress on the actuators can be reduced by diminishing the saturation limits, but these actions will have a negative effect on rise time. There is also a feature in the MATLAB GUI that applies a low-pass filter to u, which is another method of reducing chatter at the expense of a less responsive controller (Figure 19). Notice how the filtering reduces the fraction of

time spent at the saturation limits of the actuators (Figure 20). To create Figure 19 and Figure 20, a second-order Butterworth low-pass filter with a 50 $\frac{cycles}{time\ unit}$ cutoff was applied to u_1.

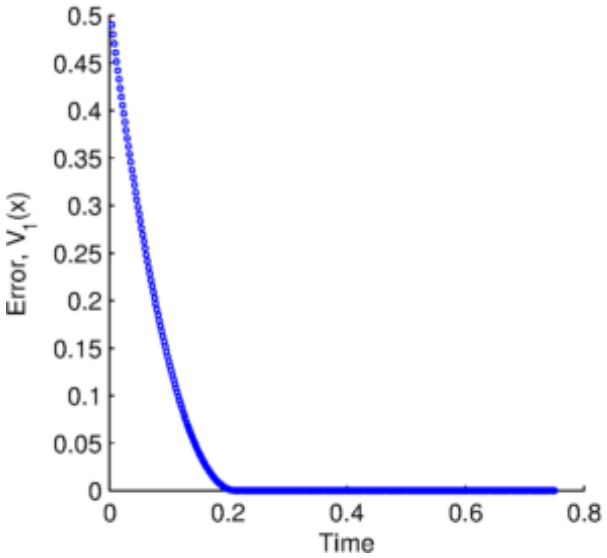


Figure 17. First-order simulation: value of the CLF V_1

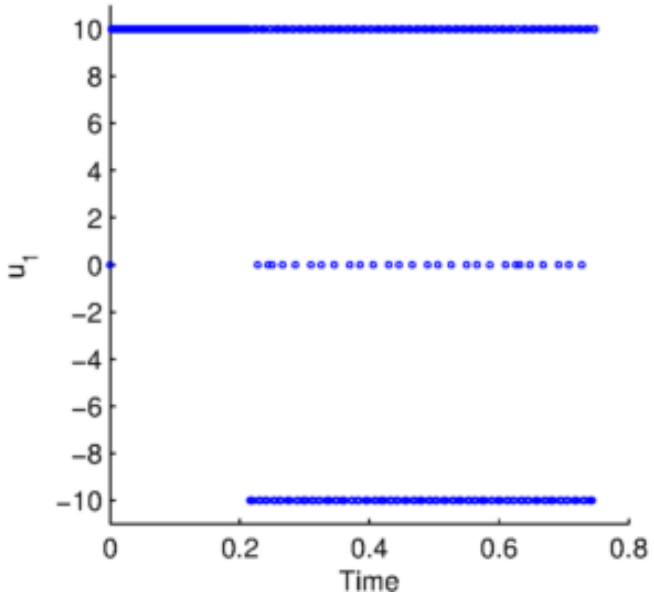


Figure 18. First-order system: control effort. The control effort is often saturated.

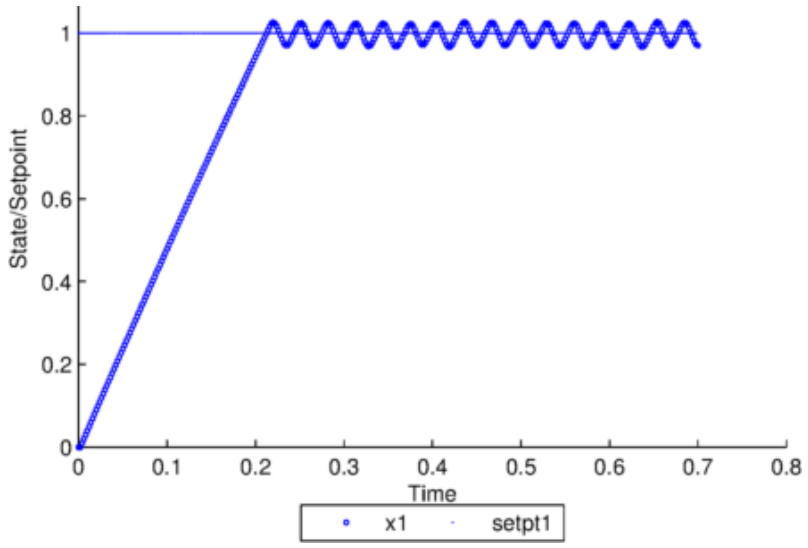


Figure 19. Trajectory of the system with a filtered control effort (compare to Figure 16). The magnitude of the chatter is greater but the frequency is less. The net effect is fractionally less time spent at the actuator saturation limits.

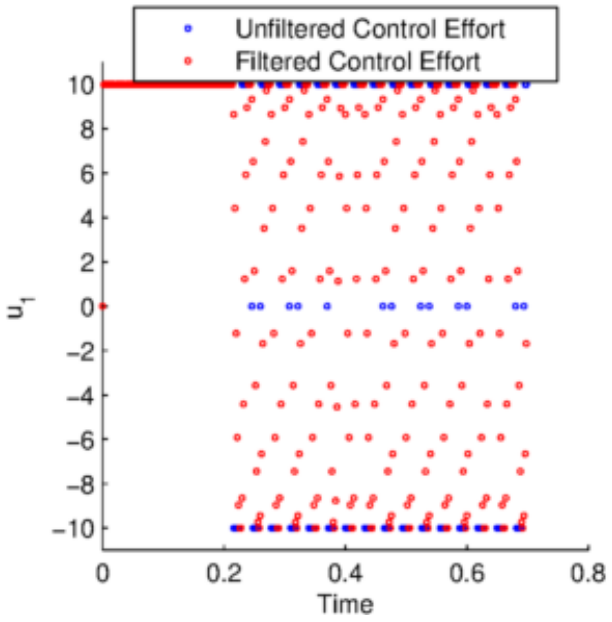


Figure 20. A low-pass filter reduces the fraction of time spent at the actuator's saturation limits.

Simulation 2: First-order sensitivity analysis

In this section, the sensitivity of the switched Lyapunov controller's performance to controller parameter variations is studied in a systematic way. The same first-order system is used but the coefficient is slightly different:

$$\dot{x} = \frac{1}{2}(u - x) \tag{4.6}$$

First, a step response simulation was run with the same controller parameters from Simulation 1. This was the control case and some baseline performance metrics {rise time, settling time, percent overshoot} were collected. Then the user-adjustable parameters of the MATLAB simulator were varied in turn by a factor of two and the simulation was re-run, collecting the same performance metrics. Finally, all of the parameters were changed by a factor of two simultaneously to gauge the cumulative effect. The baseline parameters

are shown in the center column of Table 4 and the perturbed parameters are shown in the right-hand column.

Table 4. Parameters of Simulation 2

	Baseline	Perturbed
Time step	10^{-3} units	$2 * 10^{-3}$ units
Control Effort Saturation	±10	±20
Aggressiveness, $\dot{V}_{target}(0)$	-120 dB	-114 dB
V_2 step sharpness, β	0.5	0.5
γ	0.1	0.1
Switching threshold, $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$	0.001	0.002
Initial condition	0	0
Setpoint	1	1

The graphical results of the baseline simulation are shown in Figure 21. Rise time (measured by the convention of 10%-90% of the setpoint) was 0.076 time units and the overshoot was 1.9%. Settling time is not a meaningful metric because there were no decaying oscillations about the setpoint.

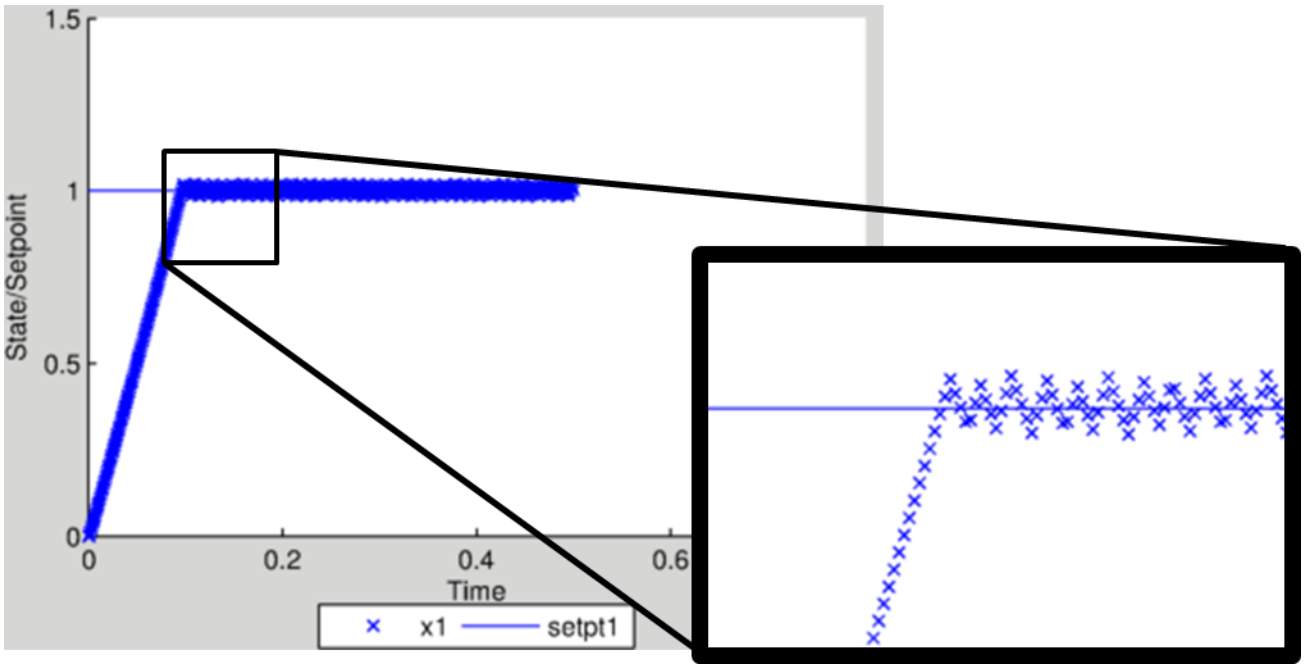


Figure 21. Simulation 2: baseline controller parameters.

The measurements of {rise time, overshoot, settling time} for all parameter sets are given in Table 5. A few observations can be made from this data:

- Doubling the time step caused the system to overshoot by approximately twice as much as the baseline case. This is intuitive and it ties into the discussion of the importance of small time steps (see *Implications of linearizations in the derivation*, Chapter 3).
- Since the baseline control effort was almost always saturated, doubling the saturation limits roughly halved the rise time. The system was 'driven harder' towards the setpoint. However, additional overshoot was caused due to a faster velocity with the same time step.
- Variations of the aggressiveness had almost no effect on the system because saturation of the input was the limiting factor. The controller was already

driving the system towards the setpoint as quickly as possible given the saturation limits.

- Adjustments to the switching threshold also had no effect because the system is controllable with V_1 over its entire trajectory. Recall from (*The trivial first-order case*, Chapter 2) that an alternative CLF V_2 is not useful for first-order systems like this. However, V_2 is important for some of the following simulations.
- Figure 22 shows the controller's performance after the cumulative parameter changes. Comparing to Figure 21, the rise time was noticeably faster but the system overshoot also increased. These effects were predictable and they were due to the combination of a larger time step and larger saturation limits (i.e. the system was driven harder).

Table 5. Numerical results of the first-order sensitivity analysis

	Baseline	2X Time Step	2X Saturation Limits	0.5X Aggressiveness	2X Switching Threshold	Cumulative Changes
Rise Time	0.077	0.076	.039	.077	.077	0.04
Overshoot	1.9%	3%	2.5%	1.9%	1.9%	6.8%
Settling Time	N/A	N/A	N/A	N/A	N/A	N/A
Summary of Effects	Baseline	Significantly more overshoot	Faster rise time but more overshoot	No effect	No effect	Faster rise time but more overshoot

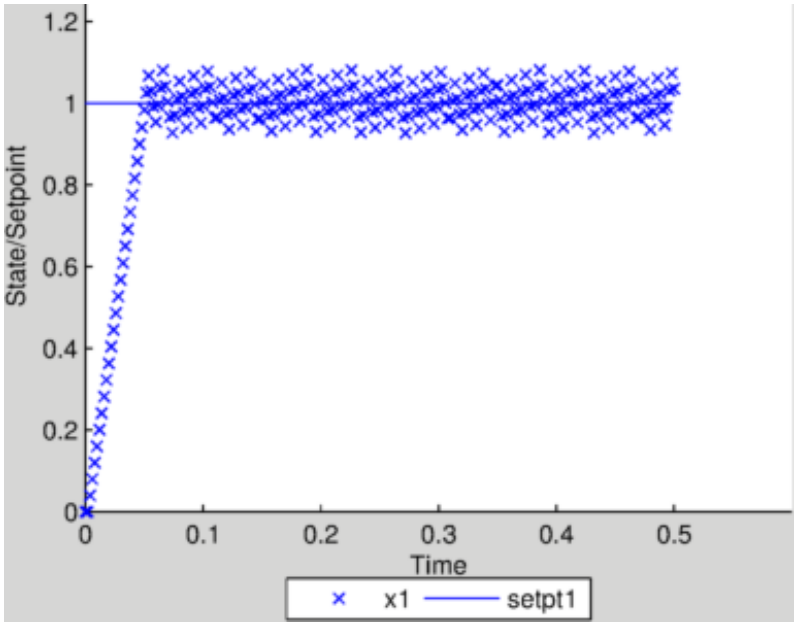


Figure 22. Simulation 2: the effect of cumulative controller parameter changes.

Although they are not applicable to the sensitivity analysis, the other plots from the baseline simulation are presented in Figure 23. As with Simulation 1, the Lyapunov value decreased exponentially due to the tapering of $\dot{V}_{target}$ and the controller saturated the input almost constantly due to the large magnitude of $\dot{V}_{target}$.

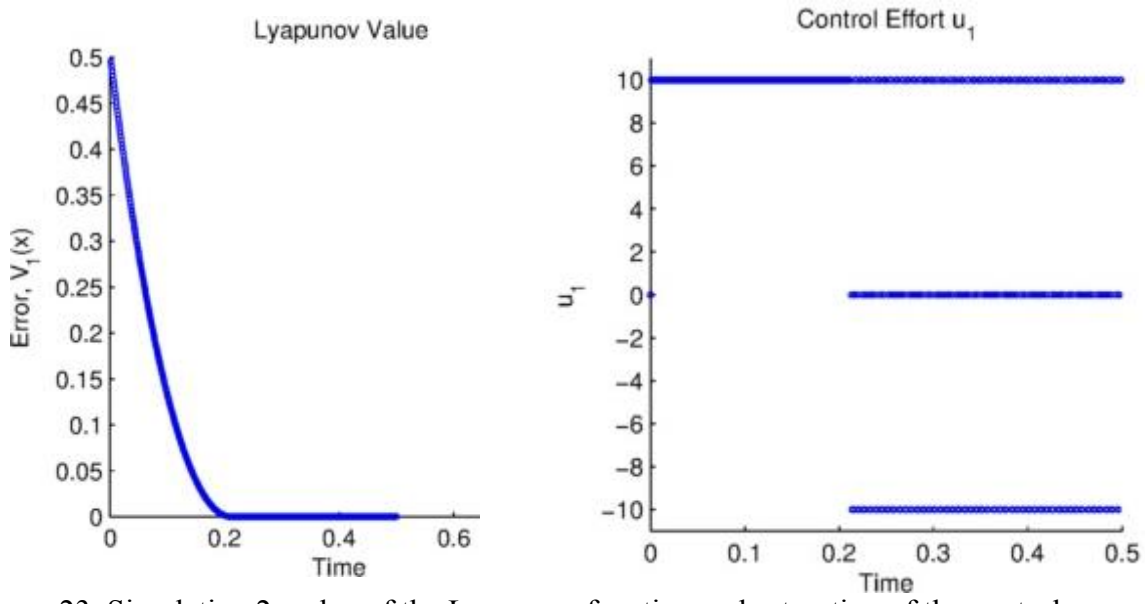


Figure 23. Simulation 2: value of the Lyapunov function and saturation of the control effort when simulating with the baseline parameter set.

Simulation 3: First-order robustness analysis

Slotine and Li [1991] define controller robustness as "sensitivity to effects which are not considered in the design, such as disturbances, measurement noise, unmodeled dynamics, etc." This section analyzes the robustness of the switched Lyapunov controller. Another first-order system was used; but, in this case, the system is open-loop unstable (i.e. the state grows without bounds unless a stabilizing control effort is applied). The coefficients of the state-space equations of the system were adjusted to varying degrees

without changing the dynamic model. This was equivalent to introducing measurement noise and/or parameter uncertainty. The baseline system was:

$$\dot{x} = x - u \tag{4.7}$$

Which is controllable with $C = [1 \ -1]$ (full row rank).

For comparison, trials were run with plant coefficients that were 100% and 400% larger:

$$\dot{x}_{plant} = 2(x_{plant} - u) \tag{4.8}$$

$$\dot{x}_{plant} = 5(x_{plant} - u) \tag{4.9}$$

While maintaining the dynamic model:

$$\dot{x}_{model} = x_{plant} - u \tag{4.10}$$

The simulation parameters from Simulation 1 were repeated (Table 3). For the control case (model matching the plant perfectly), the results were very similar to those from Simulation 1. There was no overshoot and the rise time was nearly identical at 0.077 time units. When modeling error was introduced, it caused more chatter (Figure 24) but the system remained stable. For this system and these simulation parameters, it can be concluded that the switched Lyapunov controller is quite robust.

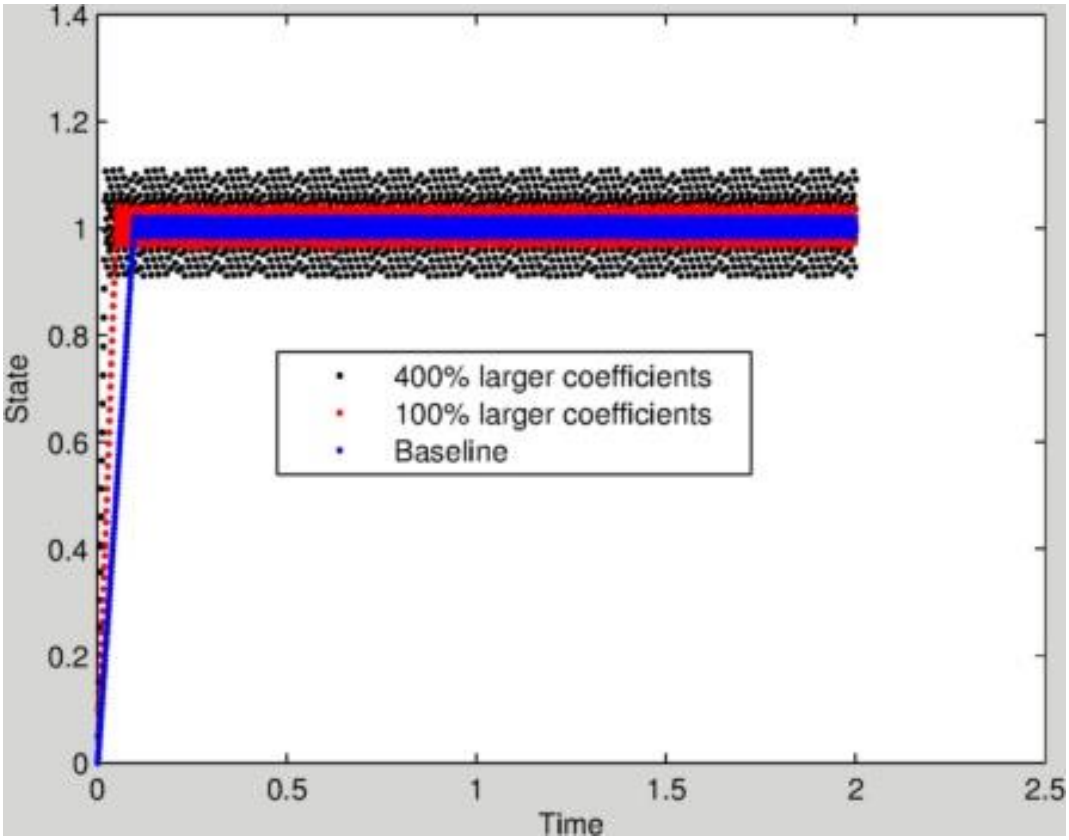


Figure 24. A plot of first-order robustness trials. The system is stabilized despite drastically "faster" dynamics but there is more chatter.

Simulation 4: Sensitivity analysis of a coupled third-order system

While the systems from previous simulations could have been controlled with a PID controller, the fourth simulation involves a coupled, over-actuated system that would likely be impossible to control globally with PID controllers. This system is much more sensitive to controller parameter variations than the first-order system that was analyzed in Simulation 3. The system definition is:

$$\dot{x}_1 = x_1 x_2 + u_1 + u_2 \tag{4.11a}$$

$$\dot{x}_2 = u_1 x_1 x_2 + u_3 \tag{4.11b}$$

$$\dot{x}_3 = -u_4 x_1 x_3 \tag{4.11c}$$

The parameters for this simulation are given in Table 10. The significant differences in the baseline case are twofold:

- The saturation limits were increased from ±10 to ±20; this was necessary to counteract the unforced dynamics, which are "faster" than the previous systems.
- The simulation time step was decreased from 0.001 to 0.0001 units. This simulation was very sensitive to the time step size, and the smaller time step was necessary to asymptotically stabilize the system.

Again, several of the controller parameters were perturbed individually in order to degrade the controller's performance. Evaluating the effects of these variations was not as straight-forward as it was in the first-order case because there are three states; the typical metrics of rise time, settling time, and percent overshoot of a setpoint do not apply. So, performance was evaluated on two criteria:

- The simulation time elapsed before the scalar Lyapunov value $V_1(\boldsymbol{x})$ fell and remained below one. (Recall that $V_1 = \frac{1}{2}\sum_{i=1}^{3}(x_i - x_{i,setpt})^2$, so it is a measure of the error across all three states.) This is similar to rise time on a first-order system.
- The value of $V_1(\boldsymbol{x})$ at the end of the simulation (smaller is better). This is similar to a steady-state error measurement on a first-order system.

Table 6. Parameters of Simulation 4

	Baseline	Perturbed
Time step	10^{-4} units	10^{-3} units
Control Effort Saturation	±20	±10
Aggressiveness, $\dot{V}_{target}(0)$	-120 dB	-100 dB
V_2 step sharpness, β	0.5	0.5
γ	0.1	0.1
Switching threshold, $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$	0.001	0.01
Initial condition	(-1 3 2)	(-1 3 2)
Setpoint	(4 -2 3)	(4 -2 3)

The results of this experiment are summarized graphically in Table 7 and numerically in Table 8. The performance for the baseline case was good. All three states were rapidly, asymptotically stabilized. It is interesting to note that, unlike the first-order systems, the individual actuators were rarely saturated. The controller tended only to saturate the actuator that will reduce $V_1(\boldsymbol{x})$ most rapidly at each moment.

When the controller was degraded, overshoot in states x_1 and x_2 were observed. There was more steady-state error for all three states and more chatter. An analysis of each individual parameter reveals the following:

- A larger time step was the largest contributing factor to steady-state error. In fact, if the time step is increased even further (from 0.001 units to 0.01 units),

the performance is erratic and marginally stable (Figure 25). Chapter Two explains why; it is due to the heavy use of linearization in the controller.

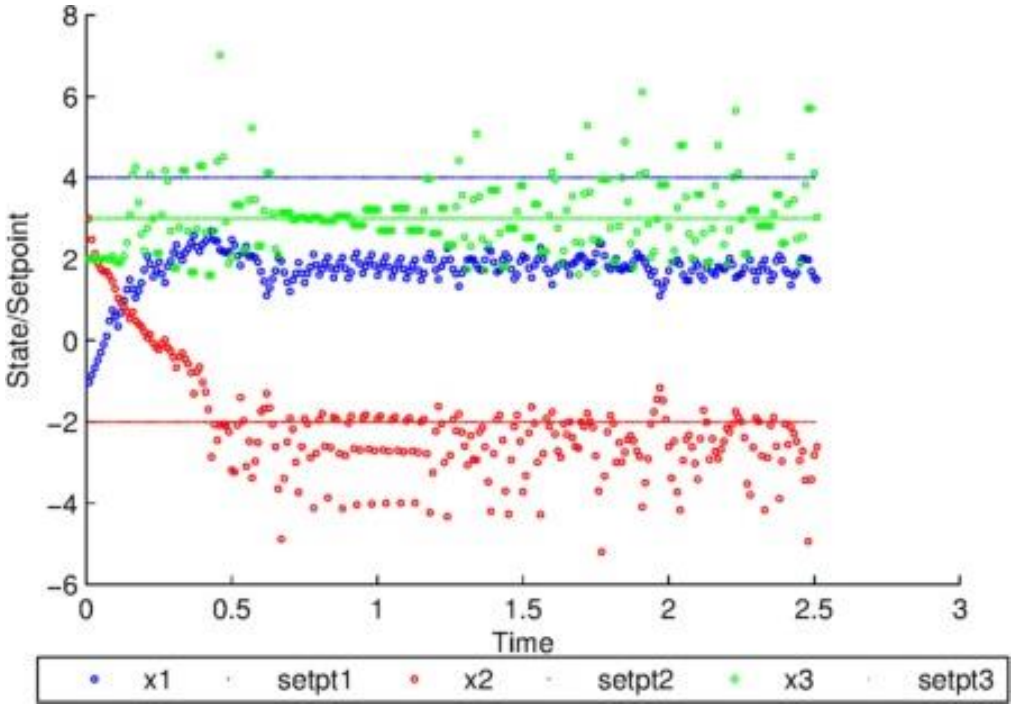


Figure 25. A larger time step leads to marginal stability for state x_1, in particular (blue).

- Decreased saturation limits were the largest contributing factor to a slower rise time. This makes sense on an intuitive level because it limits how hard the controller can drive the system.
- A less aggressive $\dot{V}_{target}(0)$ caused slightly more steady-state error but had very little effect on the rise time. This was expected because the baseline $\dot{V}_{target}(0)$ was quite large; it was driving the actuators to saturation. A less aggressive $\dot{V}_{target}(0)$ still saturates the actuators over most of the trajectory. As the states approach the setpoints and $\dot{V}_{target}$ is tapered according to

Equation 4.2, the larger $\dot{V}_{target}(0)$ performs better because it forces the actuators to continue working. It is possible to have a $\dot{V}_{target}(0)$ which is too large for the simulation time step and causes significant overshoot.

- A larger switching threshold had no effect because, for this simulation, the default Lyapunov function was able to control the system over its entire trajectory. The alternative Lyapunov function was never used.

Table 7. Graphical results of the coupled third-order sensitivity analysis

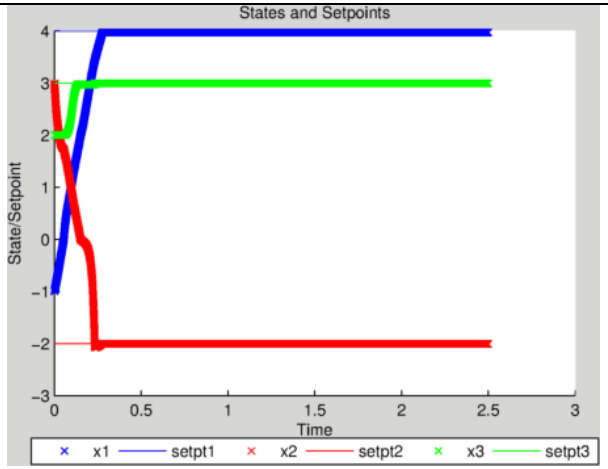
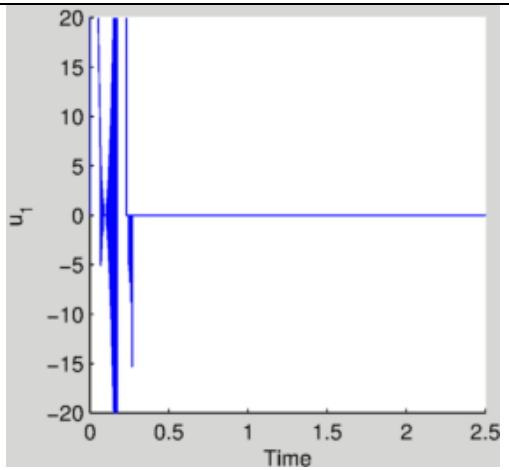
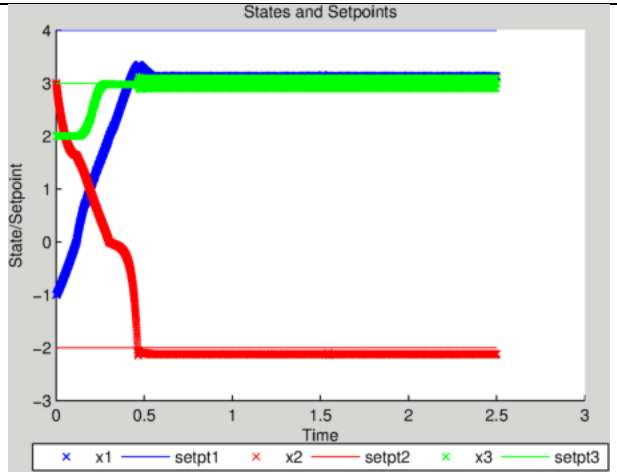
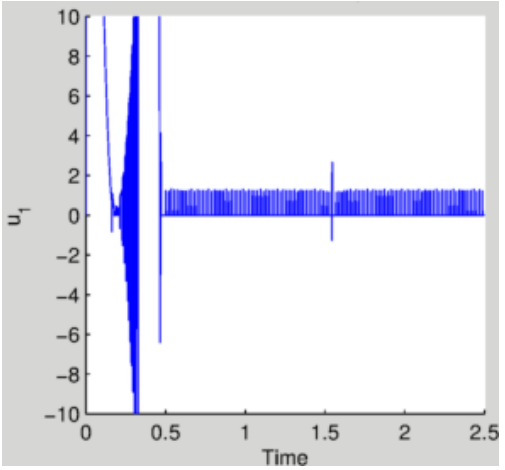
Baseline Case		
Cumulative Changes: Worst Performance		

Table 8. Numerical results of the coupled third-order sensitivity analysis

Perturbed Parameters	Time for $\boldsymbol{V_1(x)}$ to fall to one (analogous to rise time)	$\boldsymbol{V_1(x)}$ at t=2.5 (analogous to steady-state error)	Summary
None (baseline)	0.219 (t-best)	0.00038 (t-best)	
Time step	0.223	0.35	Slower stabilization, significantly more steady-state error
Saturation	0.437 (worst)	0.0025	Very slow stabilization, slightly more steady-state error
Aggressiveness	0.219 (t-best)	0.00069	Slightly more steady-state error
Threshold for switching	0.219 (t-best)	0.00038 (t-best)	No effect (the alternative Lyapunov function was not used)
Cumulative	0.436	0.38 (worst)	Very slow stabilization and significantly more steady-state error

Simulation 5: Tracking with seven motors

Simulation 5 extends the switched Lyapunov algorithm in two interesting ways. It is a large (seventh-order) system and it is a combination tracking/regulation problem. The state-space definition of the system is the equation of motion of a motor [Control Tutorials for MATLAB and Simulink, 2012]:

$$J\ddot{\theta} + b\dot{\theta} = Ki \tag{4.12}$$

$$\downarrow$$

$$J\dot{x} + bx = Ku$$

$$\downarrow$$

$$\dot{x} = \frac{1}{J}(Ku - bx) \tag{4.13}$$

Where

- J is the moment of inertia of the rotor [kg*m2]
- b is a viscous friction constant [N*m*s]
- K is a back-e.m.f. constant [Volts/(rad/s)]
- $x \equiv \dot{\theta}$ is the angular velocity of the rotor [rad/s]
- $u \equiv i$ is the armature current [Amps]

We extended the model to seventh order in a systematic way so it captures the simultaneous control of seven motors with different inertia, friction, and back-e.m.f. parameters (Equation 4.14). Note how the motors vary drastically; motor seven has a viscous friction constant 49 times greater than motor one, for example.

$$\dot{x}_j = j^{-1} * \left(u_j - j^2 x_j\right) \forall j \in \{1,2,\ldots,7\} \tag{4.14}$$

Having a dedicated input for each joint (i.e. a B matrix of full rank), the system is clearly controllable. (For the sake of brevity, the 7x49 controllability matrix is not shown.) This simulation was motivated by the control of a seven degree-of-freedom robotic manipulator such as the robot in Figure 26. Such a problem is familiar to the manufacturers of industrial robots, and they would typically use a separate PID controller for each joint.

A unique PID controller for every joint requires 21 total parameters to be tuned, and this reveals one of the nicest features of the switched Lyapunov controller: invariance of the number of tuning parameters with respect to system complexity. This can be critical since our experience shows that tuning parameters should vary depending on the payload. Thus PID controlled joints may unnecessarily restrict the flexibility of the system to perform a wide range of tasks. No matter the complexity of the system, the switched Lyapunov controller only requires the tuning of three quantities: time step, aggressiveness ($\dot{V}_{target}(0)$), and saturation limits. Sometimes it is very easy to specify these parameters: the time step should be as small as practicable and the saturation limits may be established by the hardware.

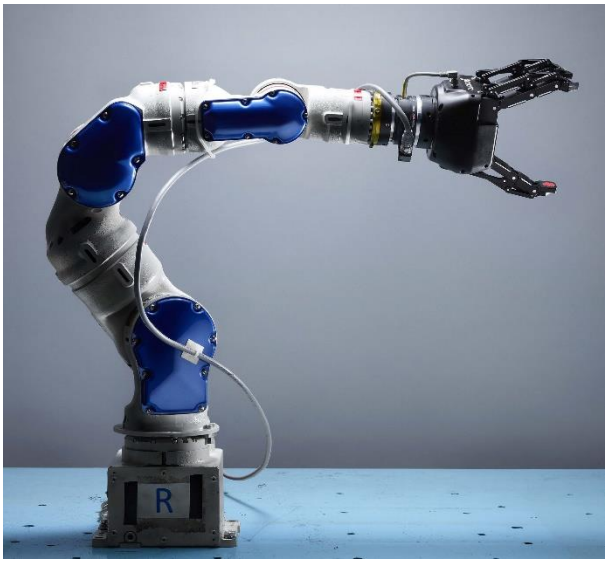


Figure 26. A seven degree-of-freedom robot (Motoman SIA5D).

The parameters of the simulation are shown in Table 9. This simulation required a bit more tuning than the rest because of the wide range of coefficient magnitudes, which

caused some states to change more rapidly than others. Notice that the saturation limits needed to be much larger for motors 5, 6, and 7 to compensate for the system's unforced dynamics. It is unlikely that even a large industrial robot would require such high joint current, but it is useful to test the controller on such a broad range of dynamics, simultaneously.

Motor one (state x_1) is most similar to the Motoman SIA5 manipulator we use, which has a maximum angular joint velocity of 9 rad/s [Schroeder, 2011] and a maximum motor current of 16A. (The other motor parameters are unknown.) To be realistic, the simulation time step was established at the maximum control rate of a Motoman SIA5 (1 kHz).

Table 9. Parameters of Simulation 5

Time step	1 ms
Control Effort Saturation	± [16 20 30 250 500 1000 5000] Amps
Aggressiveness, $\dot{V}_{target}(0)$	$-120\,\frac{rad^2}{s^2}$
V_2 step sharpness, β	0.5
γ	0.1
Switching threshold, $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$	0.001
Initial condition	[-1 -2 -3 -4 -5 -6 -7] rad/s
Setpoint	[1+2*cos(2t) 2 3 4 5 6 7+sin(t)] rad/s

As seen in Figure 27, the controller performed well. All states were stabilized rapidly and without overshoot, although there is a noticeable amount of steady-state error for states 3-6. The Lyapunov value, a measurement error across all states, dropped rapidly and exponentially, as it had done with the lower-order systems (Figure 28). Finally, it is interesting to observe how the control effort related to state one (u_1) flipped signs to track the sinusoidal setpoint (Figure 29).

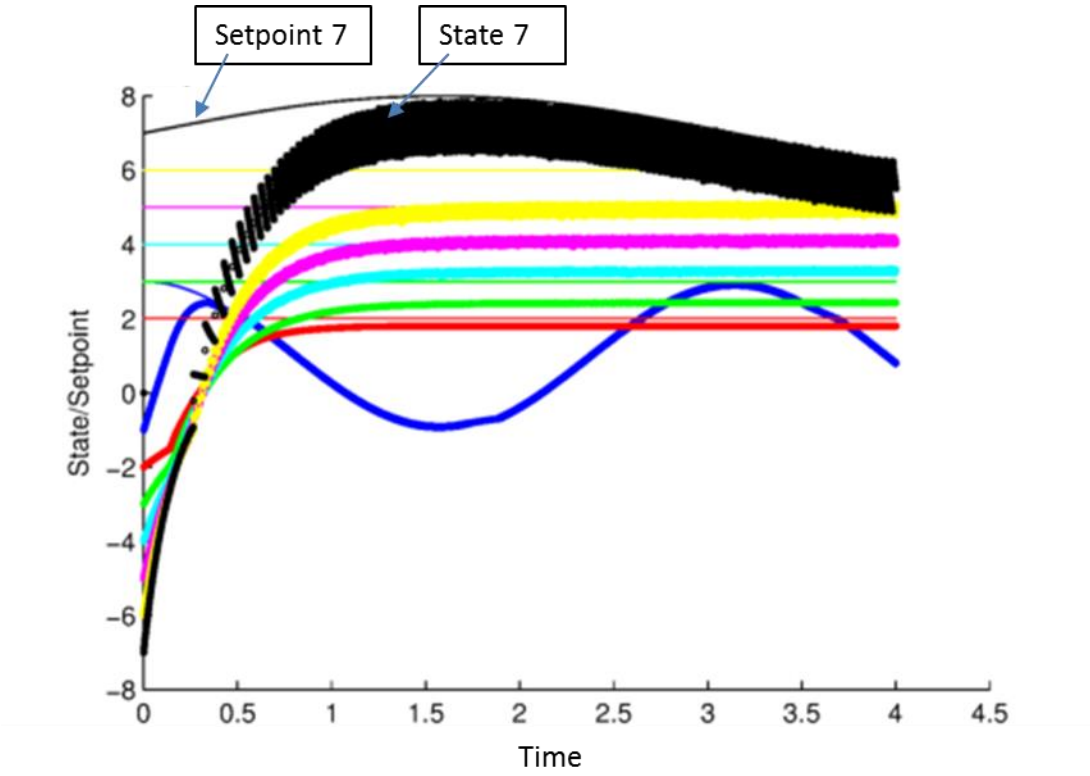


Figure 27. Velocity tracking with seven motors. There are seven state/setpoint colored pairs; the pair in black corresponding to x_7 is labeled.

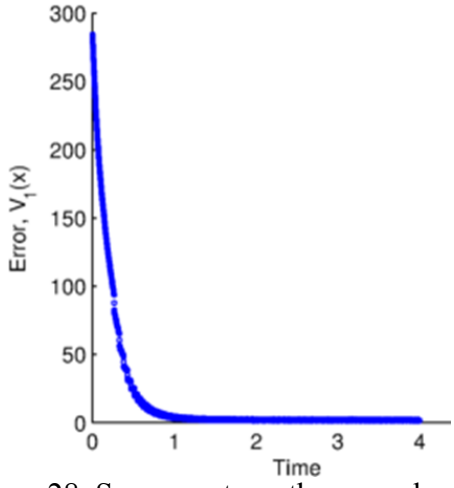


Figure 28. Seven motors: the error drops exponentially, as it was designed to do.

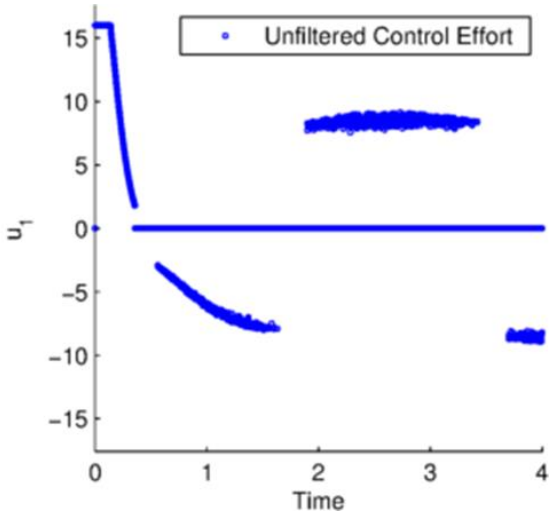


Figure 29. Seven motors: control effort u_1. Again, the control effort chatters as the system approaches the setpoint.

Simulation 6: Alternative Lyapunov function improves performance

In the previous simulations, the alternative Lyapunov function was never used because the linearizations based on $V_1(\boldsymbol{x})$ were always capable of controlling the system. A singularity in the denominator did not become an issue. For Simulation 6, the switching threshold has been adjusted upwards from 0.001 to 1 so that $V_2(\boldsymbol{x})$ becomes active occassionally. Otherwise, the simulation was identical with the 7-motor Simulation 5. Doing so revealed that the system was often nearly-uncontrollable using the linearization of $V_1(\boldsymbol{x})$. The positive effects are shown in Figure 30. There was a marked reduction in chatter, rise time, and steady-state error for almost every state when both Lyapunov functions were active, and Figure 31 displays that the average magnitude of the control

effort for every state (except x_1) was greater. This explains the faster rise times. Figure 32 shows the effect of switching from $V_1(\boldsymbol{x})$ to $V_2(\boldsymbol{x})$ in greater detail. It is clear that the behavior of the system immediately improves. Finally, Figure 33 shows the much better performance that can be achieved with an even smaller time step (10^{-4} s, no low-pass filter needed). Notice the lack of chatter.

For the present simulations, the 0.001 and 1 switching thresholds were chosen by trial-and-error, but an analytical technique for specifying a switching threshold would be an interesting topic for future research.

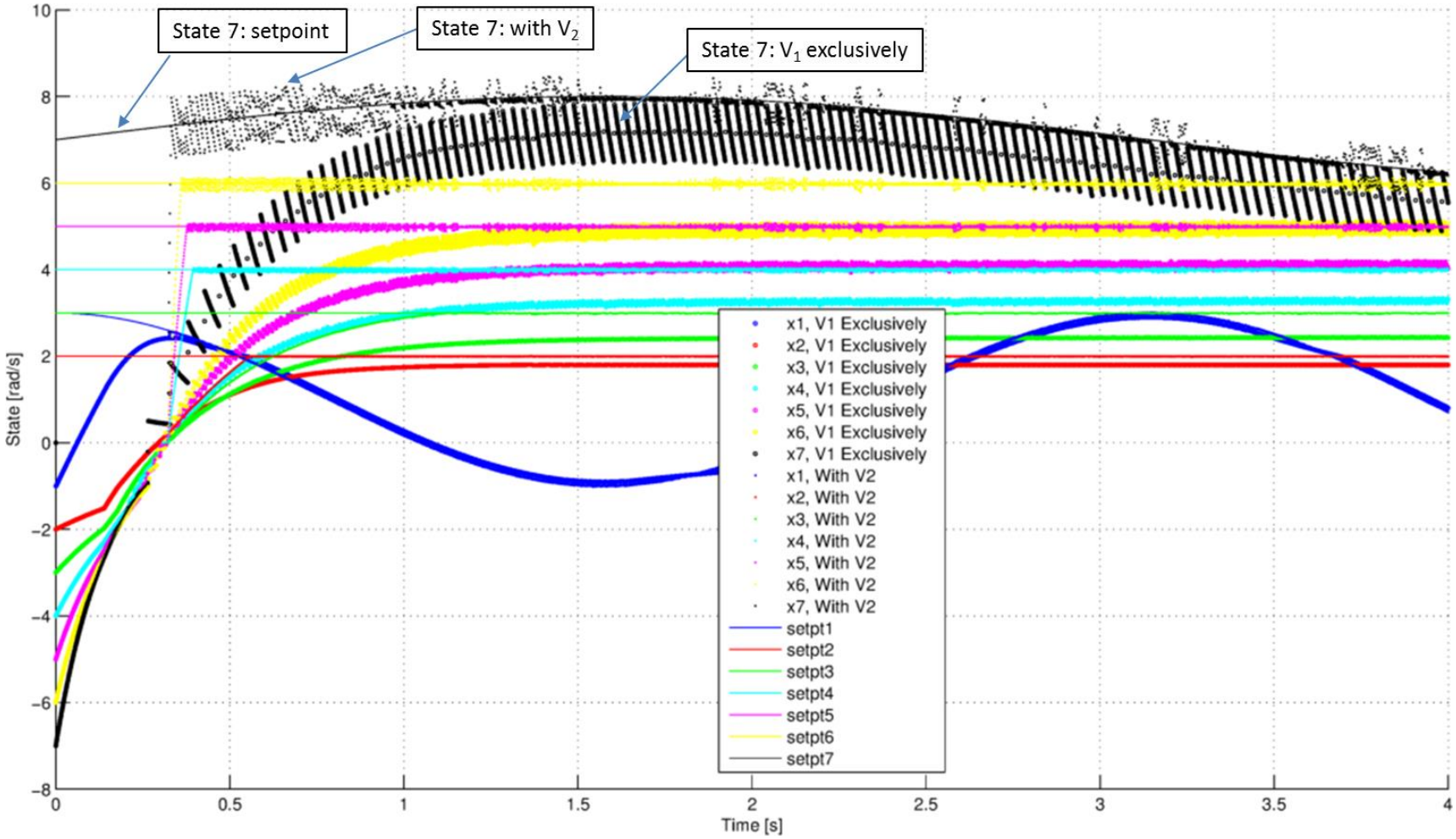


Figure 30. Performance improvements when $V_2(\boldsymbol{x})$ is applied.

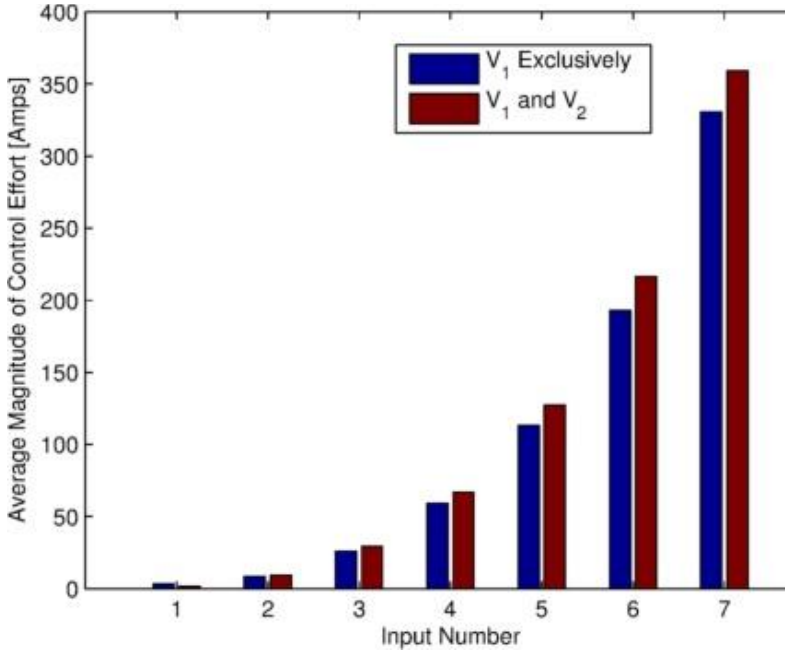


Figure 31. Average control efforts are higher when both CLF's are used.

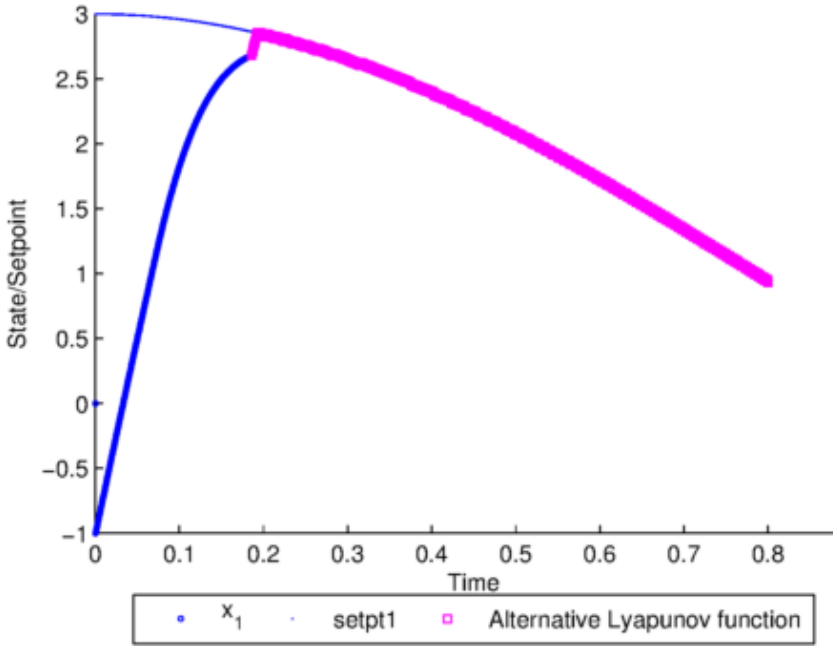


Figure 32. Switching CLF's has an immediate effect.

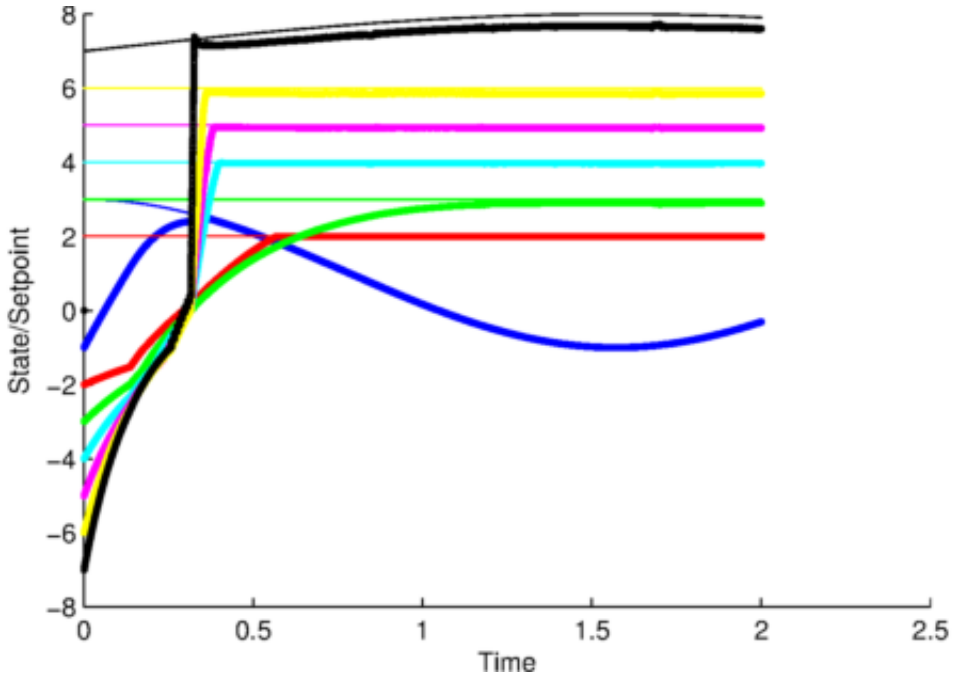


Figure 33. Greatly improved performance with a smaller time step.

Simulation 7: Comparison with McCourt

As mentioned in the literature review, McCourt [2015] recently published a switched CLF control technique which is conceptually similar to the presented method. This section compares the performance directly on an example which McCourt gave. The system definition is:

$$\dot{x}_1 = x_1^3 + x_2 + u_1 \tag{4.15a}$$

$$\dot{x}_1 = -x_1 + x_2^2 + u_2 \tag{4.15b}$$

The performance of McCourt's controller is quantified in Figure 34. Notice how the states initially move farther from the setpoint as V_1 and V_2 spike (around t=2s). This is undesirable and McCourt gave no explanation for it. Comparing against Figure 35 and Figure 36, Zelenak's controller stabilized the system much faster and the Lyapunov value

decreased monotonically (as it should). The simulation did not require a particularly small time step nor large saturation limits; these controller parameters are shown in Figure 38.

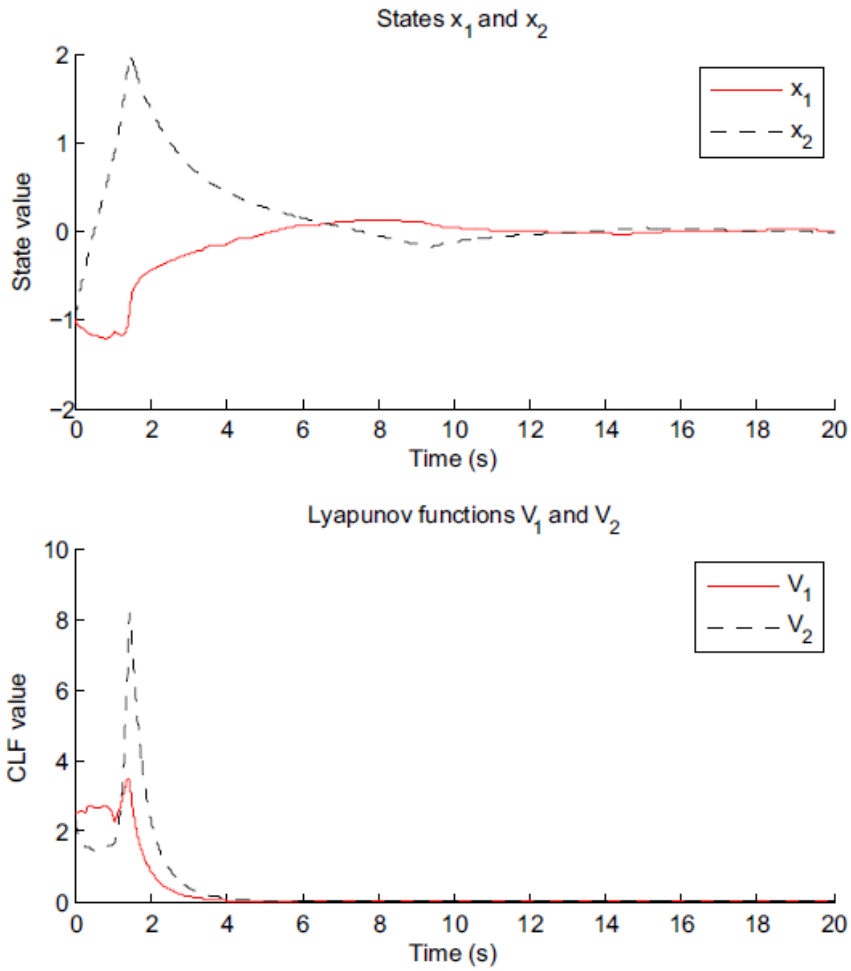


Figure 34. McCourt's controller. Note the unstable spike early in the simulation.

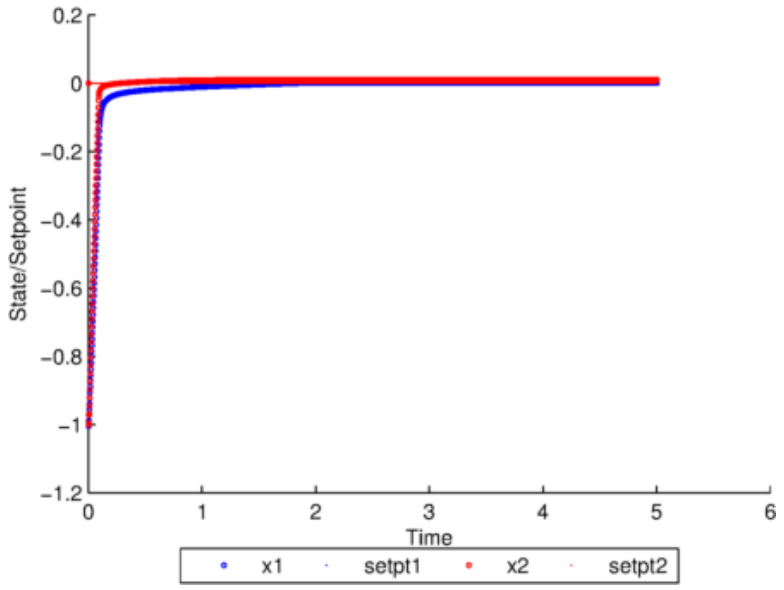


Figure 35. Zelenak's switched Lyapunov controller. Both states are stable throughout.

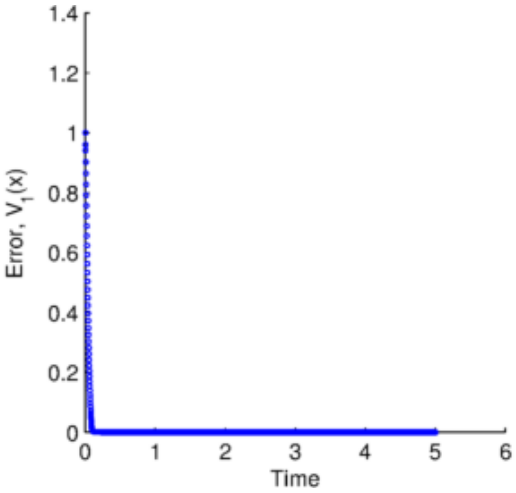


Figure 36. Zelenak's error.

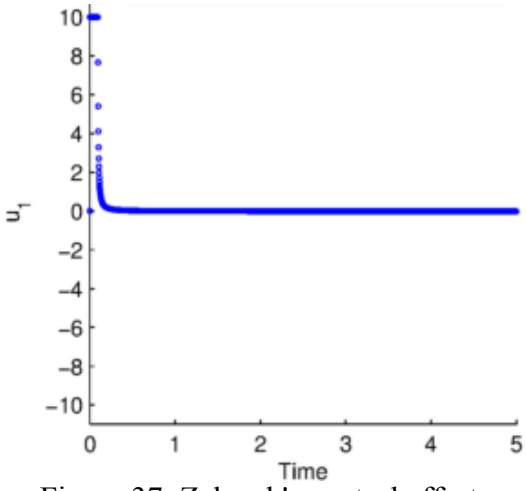


Figure 37. Zelenak's control effort.

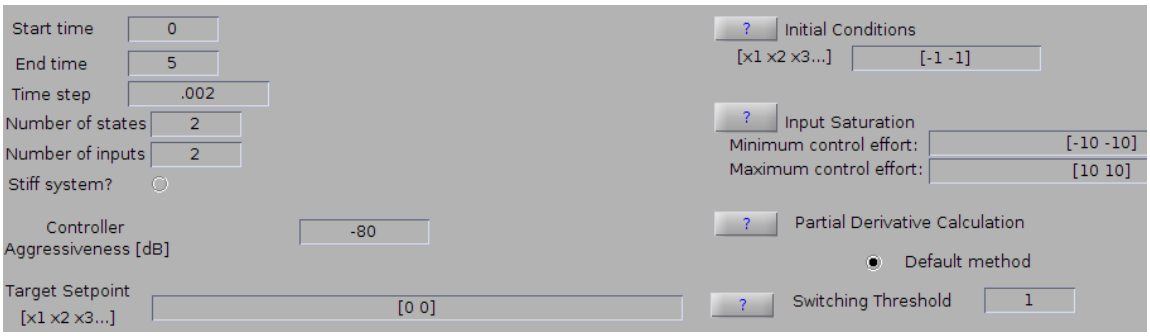


Figure 38. Parameters of Zelenak's controller for the comparison against McCourt's switched Lyapunov controller.

The likely reason why McCourt's switched Lyapunov controller was briefly unstable was explained by Hespanha and Morse [2002]. Switching between individual regions of stability does not guarantee that the overall system is stable. However, if there is a single, continuous Lyapunov function, then stability is assured [Liberzon and Morse, 1999]. In the case of the presented algorithm, $\boldsymbol{V_1}$ serves as a continuous Lyapunov function that assures stability, even if $\boldsymbol{V_2}$ is temporarily used to make calculations. Figure 39 and Figure 40 make the distinction clear.

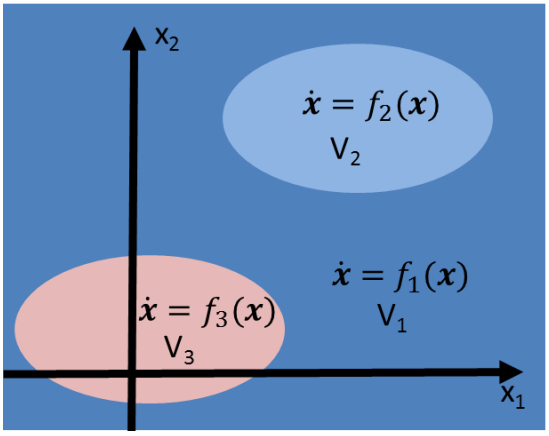


Figure 39. Three distinct Lyapunov functions ==> Possibly unstable.

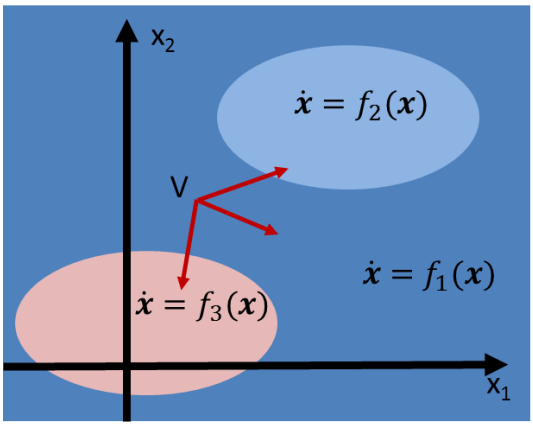


Figure 40. A single common Lyapunov function ==> Guaranteed stability.

4.4 MARGINALLY STABILIZED SYSTEMS

Simulation 8: van der Pol oscillator

The next system of interest is the van der Pol oscillator [van der Pol, 1920], which was one of the first nonlinear systems to receive significant research attention. It describes an electrical circuit which produces cyclical current flows. These oscillations came to be known as *limit cycles* and their discovery opened a new topic of research. The oscillator is typically studied as an open-loop system, but we've added a control input to see if it could be controlled. The form of the system for this simulation was:

$$\dot{x}_1 = x_2 + u \tag{4.16a}$$

$$\dot{x}_2 = -x_1 + (1 - x_1^2)x_2 + u \tag{4.16b}$$

The local, nonlinear form of controllability (*accessibility)* can be analyzed with the formula for input-affine nonlinear systems from Chapter Two (refer also to [Hedrick, 2005] for a similar example). For this system, $f = \begin{bmatrix} x_2 \\ -x_1 + (1 - x_1^2)x_2 \end{bmatrix}$ and $g = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

The form of the accessibility distribution is

$$C = [g\ [f, g]] \tag{4.17}$$

$[f, g]$, a Lie bracket, is calculated as follows:

$$[f, g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g = \mathbf{0} * f - \begin{bmatrix} 0 & 1 \\ -1 - 2x_1x_2 & 1 - x_1^2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ x_1^2 + 2x_1x_2 \end{bmatrix} \tag{4.18}$$

Thus, the controllability matrix for the system is

$$C = \begin{bmatrix} 1 & -1 \\ 1 & x_1^2 + 2x_1x_2 \end{bmatrix} \tag{4.19}$$

Which has full row rank and therefore is locally accessible unless

$$x_1^2 + 2x_1x_2 = -1 \tag{4.20}$$

The simulation parameters are given in Table 10. The most significant difference from previous simulations is that the saturation limits are increased from ±10 to ±100. This was necessary to overcome the unforced dynamics of the system.

Table 10. Parameters for Simulation 8 (Van der Pol oscillator)

Time step	$3 * 10^{-4}$ units
Control Effort Saturation	±100
Aggressiveness, $\dot{V}_{target}(0)$	-120 dB
V_2 step sharpness, β	0.5
γ	0.1
Threshold for switching, $\lvert D_1^2 + D_2^2 + \cdots + D_m^2 \rvert_{threshold}$	0.001
Initial condition	(-1,5)
Setpoint	(0,0)

The limit cycle (which occurs under open-loop conditions) is apparent in Figure 41.

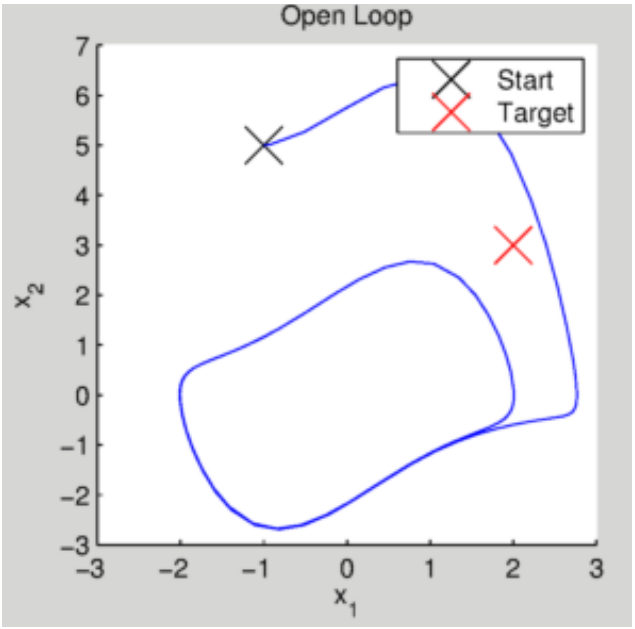


Figure 41. The open-loop van der Pol limit cycle.

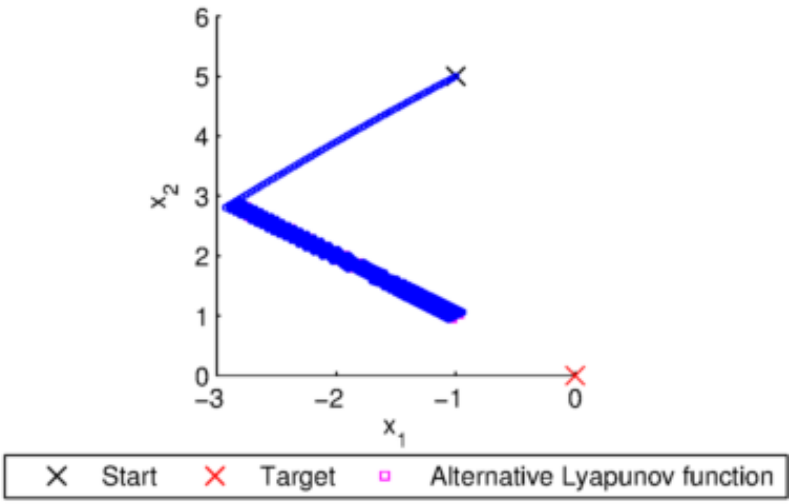


Figure 42. Van der Pol closed-loop performance. The system is not asymptotically stabilized because it is inaccessible at (-1,1).

Unfortunately, it is clear from Figure 42 that the system is only marginally stabilized. It becomes "stuck" at (-1,1). According to the controllability matrix, the system is inaccessible at this point because $x_1^2 + 2x_1x_2 = -1$. So, it is not the controller's fault that the system was not asymptotically stabilized. In fact, the controller switches to V_2 towards the end of its trajectory, but that cannot help in this scenario. Equation 3.28 is satisfied, which explains the failure from the standpoint of the proof provided in Section 3.2 *Extension to systems of any order and any number of inputs*.

Sub-study: Fully Actuated van der Pol Oscillator

The underactuated van der Pol system in the previous example was not asymptotically stabilized, but if the system is modified slightly, it becomes globally accessible and the switched Lyapunov controller can asymptotically stabilize it. A second input is added to the system:

$$\dot{x}_1 = x_2 + u_2 \tag{4.21a}$$

$$\dot{x}_2 = -x_1 + (1 - x_1^2)x_2 + u_1 \tag{4.21b}$$

With a second input, the controllability matrix has full row rank globally:

$$C = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 + x_1^2 & 1 + 2x_1x_2 \end{bmatrix} \tag{4.22}$$

The controller asymptotically stabilizes the system almost immediately (Figure 43). After the system has been stabilized, chatter of the actuators ceases (Figure 44). A more aggressive $\dot{V}_{target}$ would cause the system to stabilize even faster but the actuators would chatter more severely.

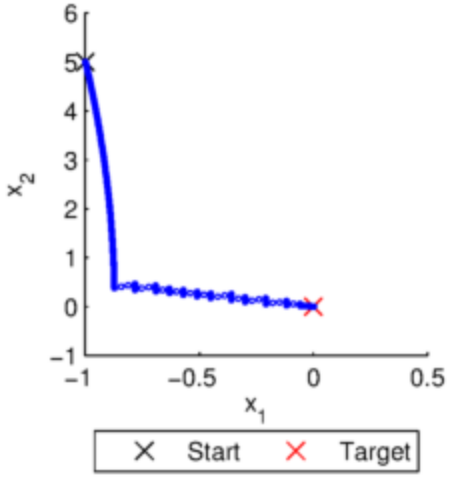


Figure 43. When it is modified to be globally accessible, the switched Lyapunov controller asymptotically stabilizes the fully actuated Van der Pol oscillator

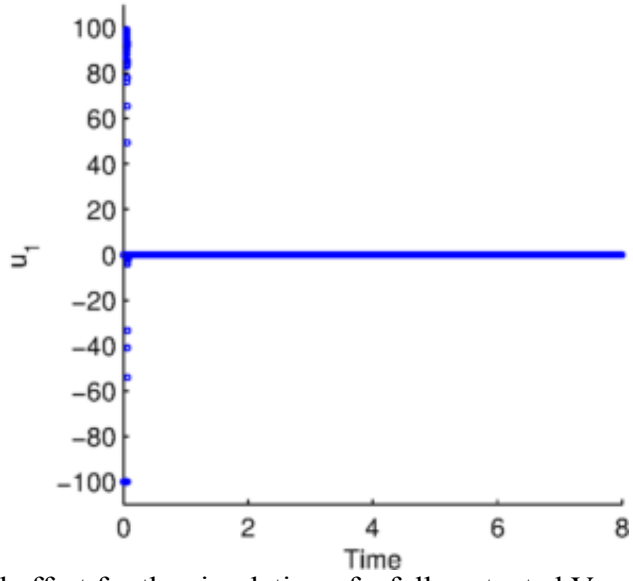


Figure 44. Control effort for the simulation of a fully actuated Van der Pol oscillator.

4.5 Comparison with PID controllers

In this section, the switched Lyapunov controller is compared against the industry-standard PID controller for three different cases. When applicable, the PID controllers were tuned with the Ziegler-Nichols (ZN) method, which is a standard heuristic-based technique. For these step response tests, the ZN method yields a fast rise time and oscillations that subside approximately according to the "quarter decay ratio," i.e. each successive overshoot is 75% smaller than the previous. Downsides to the ZN method are significant overshoot and long settling times [Visioli, 1999].

A summary of the results is that the switched Lyapunov controller performs equally well or better on first-order systems (by most metrics) but is not a good choice for higher-order integrators where a PID controller performs well. The poor performance on

integrators arises because the switched Lyapunov controller does not anticipate the future behavior of the system like the derivative term of a PID controller.

Simulation 9: First-Order PID Controller

The first comparison was made on a very simple first-order system:

$$\dot{x} = x - u \tag{4.23}$$

The ZN approach does not apply for this system, so a variety of PI and PD controllers were compared against (Figure 45). The PI/PD simulations and the switched Lyapunov simulation used the same time step (10^{-4} units).

Table 11. Parameters of the first-order Lyapunov controller

Parameter	Switched Lyapunov Controller
Simulation time step	10^{-4} units
Control effort saturation	±50
Aggressiveness, $\dot{V}_{target}(0)$	-40 dB
V_2 step sharpness, β	0.5
γ	0.1
Threshold for switching, $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$	0.01

The switched Lyapunov controller had the fastest rise time and there was practically zero overshoot. The only significant drawback to the switched Lyapunov controller (on this first-order system) was chatter. While the PI controller with a very high proportional gain had comparable performance, it commands a larger actuator effort and it will not be robust (Figure 46). It might be dangerous to use a PI controller with such high gains if, for

example, the system parameters drift over time or there is a signal delay. The switched Lyapunov controller's largest-magnitude actuator efforts were noticeably smaller than those of the high-gain PI controller, indicating that it is likely more robust. The situation would have been even less favorable for the PI/PD controllers if the error were larger than one. Furthermore, cranking the proportional gain higher would provide minimal performance improvements as most of the lag occurs in the "knee" region where the error is small.

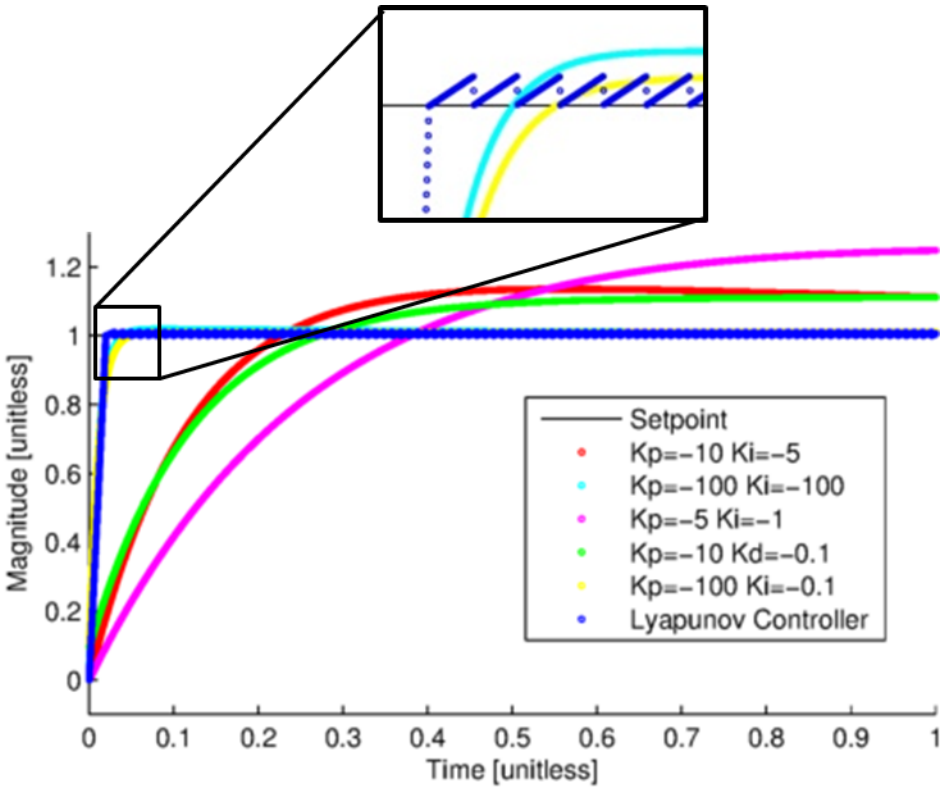


Figure 45. A comparison with various PID controllers on a first-order system. Note that the switched Lyapunov controller has the fastest rise time yet its overshoot is negligible.

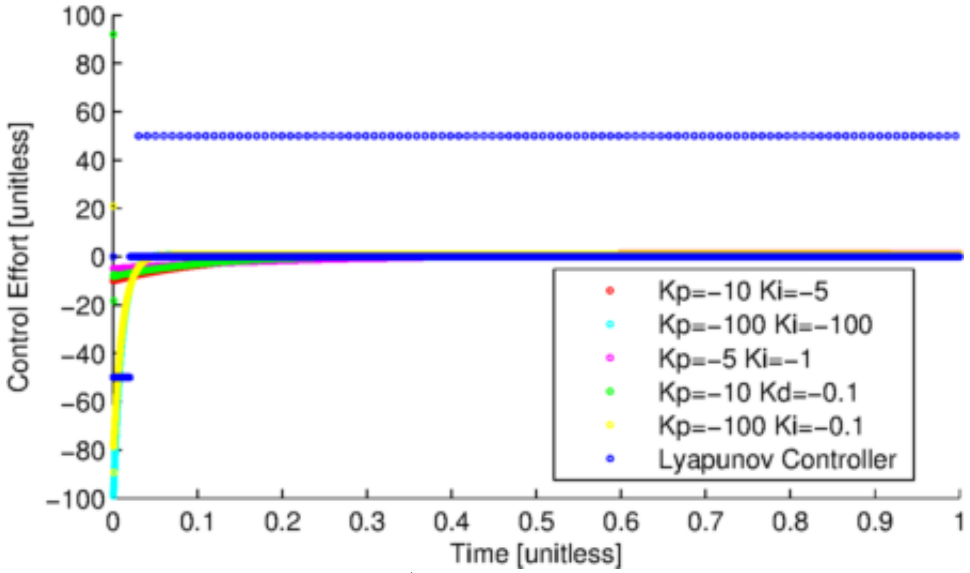


Figure 46. Control efforts during the 1st-order PID comparison. The switched Lyapunov controller chatters significantly but the magnitude of its largest control effort (50) is less than that of the PID controllers where Kp=-100.

This simulation is another example that benefits greatly from low-pass filtering of *u* to reduce chatter, as shown in Figure 47. The cutoff frequency for this filter was 1kHz.

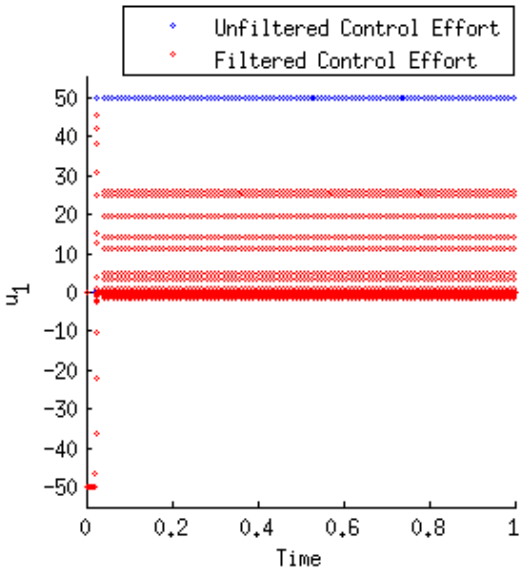


Figure 47. A low-pass filter (red) smooths the chatter.

Simulation 10: Second-Order PID Controller

PID controllers are commonly applied to first-order systems. However, single-input integrators are a special case where PID control can be used on higher-order systems. A second-order example is a mass-spring-damper system:

$$\dot{x}_1 = x_2 \tag{4.24a}$$

$$\dot{x}_2 = \frac{1}{2}(-4x_2 - 3x_1 + u) \tag{4.24b}$$

The ZN method was used for PID tuning, and the parameters for each controller are shown in Table 12. Tuning the switched Lyapunov controller was much more difficult for this integrator. The reason is that the Lyapunov controller does not anticipate the future behavior of the system like the derivative term of a PID controller. It seeks to reduce the error in x_1 in the short term, building up a large x_2 which integrates and causes a delayed overshoot of the setpoint. The concept is similar to integral windup. Because of this challenge, a very small time step had to be used.

Despite the significant challenge of tuning the switched Lyapunov controller, its performance was on-par with the PID controller (Figure 48). The rise time was much faster but the overshoot also increased.

The poor performance is predicted by Corollary 2 (Chapter 3), which requires $\frac{\partial f_1}{\partial u} \neq 0$ and $\frac{\partial f_2}{\partial u} \neq 0$. That is not the case here because $\frac{\partial f_1}{\partial u} = 0$; the system is only stabilized by the inherent damping of the system.

Table 12. Parameters of second-order PID and Lyapunov controllers

Parameter	**PI Controller**	**Switched Lyapunov Controller**
Simulation time step	0.01 units	10^{-5} units
K_p	120.6	
K_i	385.92	
K_d	9.42	
Control effort saturation		±10,000
Aggressiveness, $\dot{V}_{target}(0)$		-100 dB
V_2 step sharpness, β		0.5
γ		0.1
Threshold for switching, $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$		1
Setpoint	$x_1 = 1$	$x_1 = 1,\ x_2 = 0$

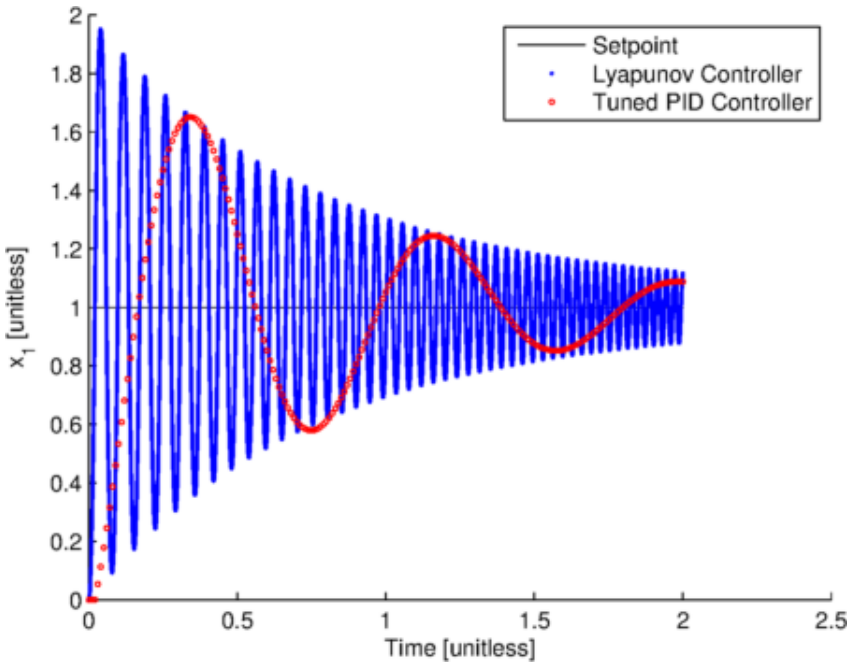


Figure 48. Comparison with a second-order PID controller.

Simulation 11: Third-Order PID Controller

If the integrator is increased from second- to third-order, the switched Lyapunov controller fails to stabilize the system (Figure 49) due, again, to integrator windup. As a rule of thumb, the switched Lyapunov controller should not be used on higher-order integrators. The system for this simulation was:

$$\dot{x}_1 = x_2 \tag{4.25a}$$

$$\dot{x}_2 = x_3 \tag{4.25b}$$

$$\dot{x}_3 = -x_1 - x_2 - x_3 + u \tag{4.25c}$$

Again, the PID controller was tuned with the Ziegler-Nichols method and the failure of the switched Lyapunov controller could have been predicted by Corollary 2 (since $\frac{\partial f_1}{\partial u} = 0$ and $\frac{\partial f_2}{\partial u} = 0$).

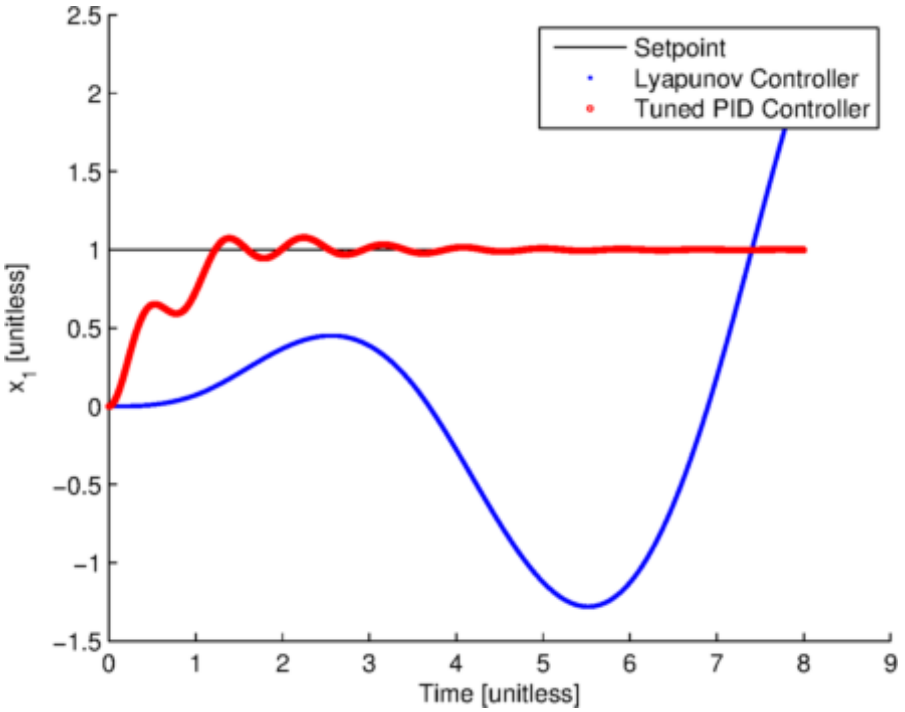


Figure 49. Comparison with third-order PID controller. The switched Lyapunov controller (blue) is unstable.

4.6 Previous simulations lumped into a high-dimensional system

The final simulated test of the switched Lyapunov controller was to lump previously asymptotically stabilized system into one very large, complex, high-dimensional system and verify that the controller can stabilize every state simultaneously. It was a 14th-order, 15-input system. Refer to Chapter Five for details; it was too large to simulate with the MATLAB simulator, so the C++ software package was the only way to evaluate the controller's performance.

4.7 Insight into controller tuning

After these extensive simulations, several insights into tuning a switched Lyapunov controller emerged. Some rules of thumb are:

- Use the smallest practical time step.
- Use the most aggressive (i.e. most negative) $\dot{V}_{target}(0)$ that the time step allows. A smaller time step allows a more negative $\dot{V}_{target}(0)$, which means faster stabilization and less steady-state error. On the other hand, a $\dot{V}_{target}(0)$ which is too negative for the time step causes more overshoot and more chatter.
- If one or more states need to be stabilized more rapidly and with less steady-state error, a "unit conversion" or weighting factor can be used to emphasize the important states (see 5.3 *Inverted pendulum simulation*).

Excessive chatter can be reduced with a smaller time step, a less negative $\dot{V}_{target}(0)$, or smaller saturation limits.

4.8 CHAPTER SUMMARY

A brief summary of the MATLAB simulations from Chapter Four follows:

- Simulations One and Eight showed that the switched Lyapunov controller performs very well on first-order systems. With a small tuning effort, it does not overshoot and rises and settles quickly. Furthermore, it can outperform a well-tuned PI/PD controller with less actuator effort.
- Simulation Two showed that the effects of tuning the controller parameters are predictable and the controller's performance was good despite varying all of the controller parameters by a factor of two. Thus the proposed

method has the potential to alleviate the burden on a controls developer from a tuning perspective.

- Simulation Three proved that the controller can be robust as it stabilized a first-order plant despite significant modeling errors.
- Simulation Four was more challenging as the plant was a coupled third-order system that is likely impossible to control with PID controllers. Nevertheless, the switched Lyapunov controller stabilized the plant rapidly and with minimal rise time, settling time, and overshoot. The sensitivity analysis revealed that a small time step was crucial on this more challenging simulation. Furthermore, the system was conceived for this effort, and no comparable systems were found in the literature.
- Simulation Five demonstrated that the controller can perform well even on a large, seventh-order system.
- Simulation Six demonstrated why the second Lyapunov function is important as it improved the controller's performance by reducing chatter.
- Simulation Seven verified the fact that an uncontrollable plant cannot be controlled, so it is important to check controllability of the plant beforehand.
- Simulation Nine was a second-order integrator that the switched Lyapunov controller was able to stabilize after careful tuning. Its performance was similar to a well-tuned PID controller.
- Simulation Ten revealed that the switched Lyapunov controller could not stabilize a third-order integrator because it does not anticipate the future

behavior of the system like the derivative term of a PID controller would. Thus, it suffers from a phenomenon similar to integral windup. This is a particular weakness of the algorithm.

In summary, the algorithm was shown to perform well on dynamic systems ranging from simple to complex. On simple systems, it performs similarly to a well-tuned PID controller. It also performs well on large or coupled systems that would be impossible or tedious to control with a PID controller. One particular weakness identified is higher-order integrators.

This concludes Chapter Four, which evaluated the proposed control algorithm on a variety of systems using an intuitive MATLAB GUI. The next chapter describes how the developed algorithm was implemented in C++ to improve its computational performance. The high-performance version is validated on hardware and on a more demanding, highly nonlinear simulation.

Chapter 5: ROS Demonstrations

5.1 Description of the ROS control package

The previous chapter explored the capabilities of the developed algorithm using a MATLAB GUI which was intended for rapid testing and tuning of the switched Lyapunov control algorithm. The MATLAB GUI was able to stabilize a wide variety of dynamic systems; however, it is computationally slow and generally would not be acceptable for hardware control. The main reasons for its (relatively) slow speed are the data collection that occurs in the background and that MATLAB is an interpreted language. There are also the difficulties of MATLAB's memory constraints and data-entry boxes of limited width, which make the MATLAB GUI unsuitable for the simulation of large or complex systems. While MATLAB provides advanced tools such as C-language cross-compilation [Ananthan, 2011] which could have solved this problem, it seemed logical to take the opportunity to provide the switched Lyapunov controller on a different platform.

To address these deficiencies and ensure the proposed controller is readily available for use in the motivating application domain (robotic force control), a second switched Lyapunov controller was written in C++ [Zelenak, 2015, "lyap_control Package Summary"] for use with the Robot Operating System (ROS). C++ is a compiled language which is much faster than MATLAB. While the measurement of a program's execution speed depends on many factors, Figure 50 [The Computer Language Benchmarks Game] shows that C++ programs are some of the fastest. The boxes-and-whiskers of Figure 50 show the speed range of the various programs that users have submitted for evaluation of each programming language. For example, even the slowest C++ implementation of the

ten benchmarks was faster than the fastest from Ruby. (Ruby is an interpreted language like MATLAB.) The ten benchmarks that the figure refers to included, for example, a calculation of the digits of pi. This C++ implementation is very fast and should be more than adequate for online control of any hardware on the market for years to come. To the best of our knowledge, this is the first nonlinear control package for ROS and the first available via the ROS web site.

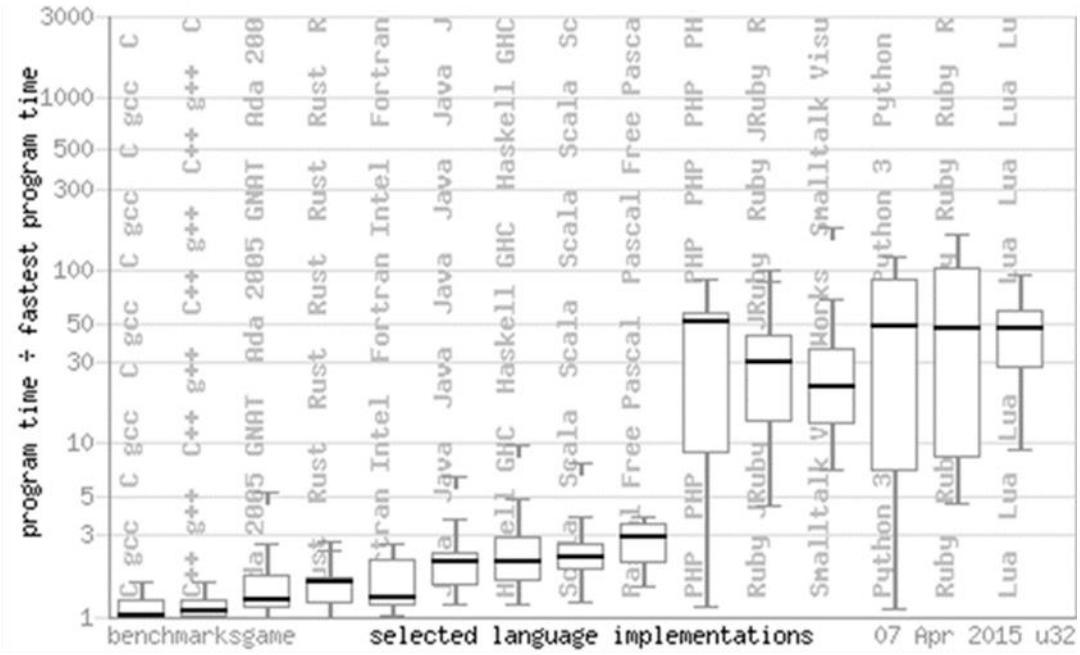


Figure 50. Speed comparison between programming languages [The Computer Language Benchmarks Game]

The second software package has very few features because it was designed to execute at the fastest possible loop rate. It was written as a *node* in ROS, which has become the dominant open-source software package for roboticists. Integration with ROS ensures it can be used with:

- Many of the most common industrial robots, including manipulators from Yaskawa-Motoman, Kuka, ABB, Fanuc, etc.
- Sensors such as the Microsoft Kinect RGB-D sensor, various force/torque sensors, laser range finders, etc.
- Several excellent simulation environments such as RVIZ and Gazebo
- Assorted hardware such as grippers from Robotiq and mobile robots from the Open Source Robotics Foundation
- A wide variety of open-source libraries that greatly simplify tasks such as object recognition

In general, the availability of the software package in ROS should greatly reduce the software development effort that is required from others who want to use the switched Lyapunov controller.

In 2014, the "wiki" website that hosts tutorials for ROS, among other things, had 976,431 visitors representing 48% annual growth. Subscriptions to various ROS-related email lists and user groups saw annual growth ranging from 12% to 48%. Finally, the quantity of binary ROS downloads grew 343%! [Open Source Robotics Foundation, 2014] Thus, it seems likely that ROS is rapidly becoming the most common software for robot-related development. The release of the switched Lyapunov control package on ROS was a calculated step towards improving its acceptance and visibility among the robotics community. While most ROS users were traditionally academics, the software is also used by the automation industry [ROS-industrial] [Yaskawa, 2013] [Edwards, 2012] [Hoorn, 2012].

Since its release, the package has been gaining popularity. Figure 51 shows the daily page views of the ROS wiki documentation [Open Source Robotics Foundation, 2015, "Info for 'lyap_control'"] compared with the views of a PID controller that was released simultaneously [Open Source Robotics Foundation, 2015, "Info for 'pid'"]. Initially the PID controller was much more popular, but the Lyapunov nonlinear controller is now getting more hits.

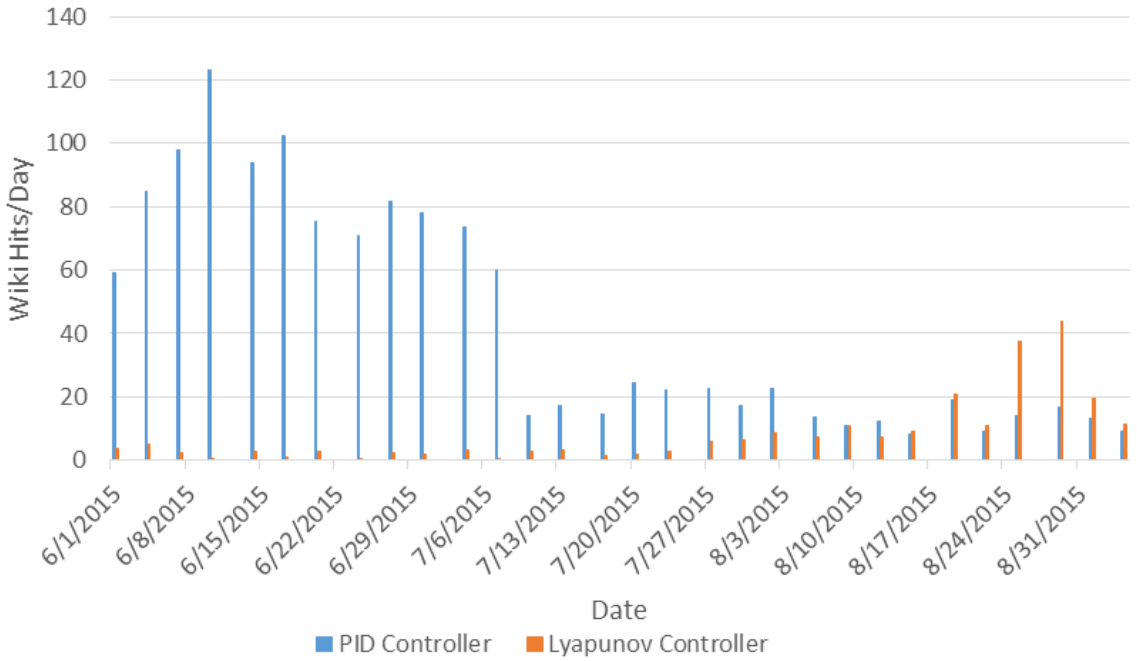


Figure 51. Daily page views of the wiki documentation for two types of ROS controller. [Open Source Robotics Foundation, 2015]

This chapter covers two demonstrations where the switched Lyapunov controller was integrated and tested in the ROS environment. The first is the simulation of an inverted pendulum, which was performed using the Gazebo physics simulator. The second is trajectory tracking with a Motoman SIA-10 industrial manipulator. These demonstrations

serve to validate the switched Lyapunov software package in two ways. First, they proved that the switched Lyapunov controller is capable of performing useful tasks. Second, they provided an opportunity to get feedback on the software package in order to make it easier to use.

5.2 C++ VS. MATLAB: COMPUTATION TIME COMPARISON

To verify that the C++ implementation is faster than its MATLAB equivalent, a comparison was run on the seven-motor Simulation 5 (with the same controller parameters and time step of 1 ms). The MATLAB version completed the four-second simulation in 24.1s while the C++ implementation finished in 4.35s, so the C++ implementation was roughly 5.5 times faster. Figure 52 shows the performance of the controllers, which was similar except extra chatter from the MATLAB controller. We are unsure what causes the extra chatter; it could be the different ordinary differential equation integrators. The MATLAB implementation defaults to a variable-step ode23 solver while the C++ implementation uses a fixed-step Euler method. It is likely that MATLAB is taking extra-large time steps and performance could be improved with a narrower accuracy tolerance or a fixed-step solver.

Notice that approximately 0.3s of data is not captured for the ROS simulation. This is due to the latency of data collection with the ROS method "*rosbag*."

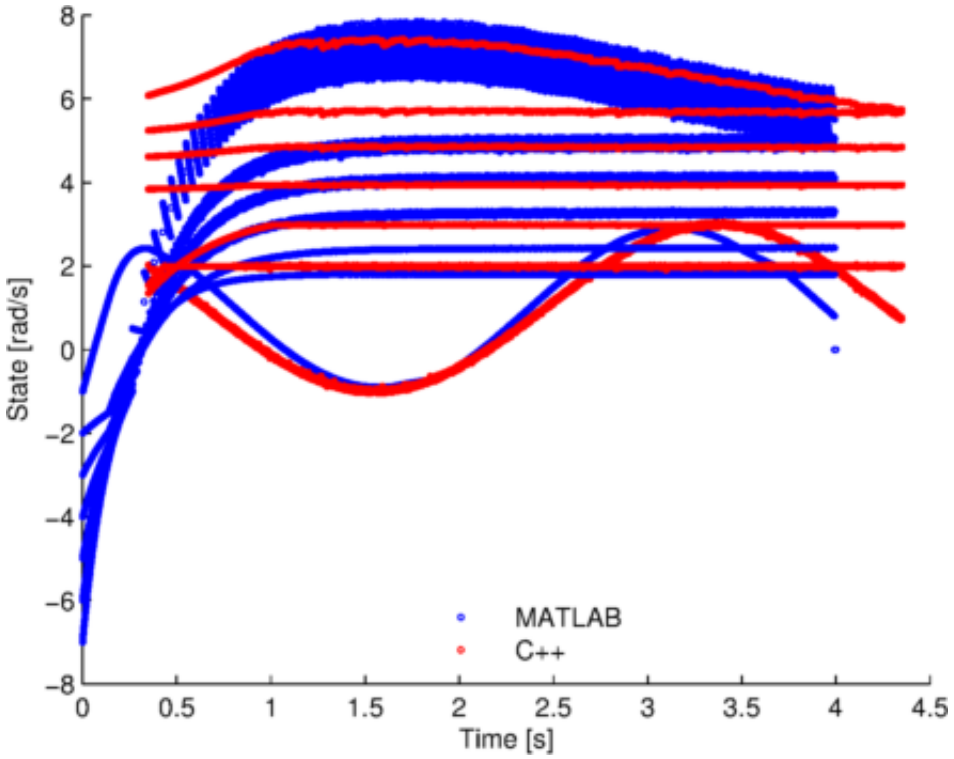


Figure 52. Comparison of C++ and MATLAB implementations of the switched Lyapunov controller. The performance is similar but the MATLAB controller chatters a bit more. The difference is likely due to different types of integrators.

5.3 Inverted pendulum simulation

Parameters of the inverted pendulum simulation in Gazebo

A schematic of the dynamic system for the first demonstration is given by [Widnall, 2009] and shown in Figure 53, along with its representation in Gazebo. Gazebo is a rigid-body physics simulator often used with ROS. When supplied with the key parameters of the system (mass, inertia, joint configuration, and basic geometries), Gazebo simulates the motion of the system. It accounts for gravity, motor torques, and other forces applied to a joint or inertial link and it is also possible to represent damping and nonlinear dynamics (such as nonlinear springs). Hence, Gazebo is often used to simulate robot dynamics before

a control algorithm is used on hardware. In this case, the goal was to start with the second link oriented downwards (θ_1=0°) then regulate it to an upright position.

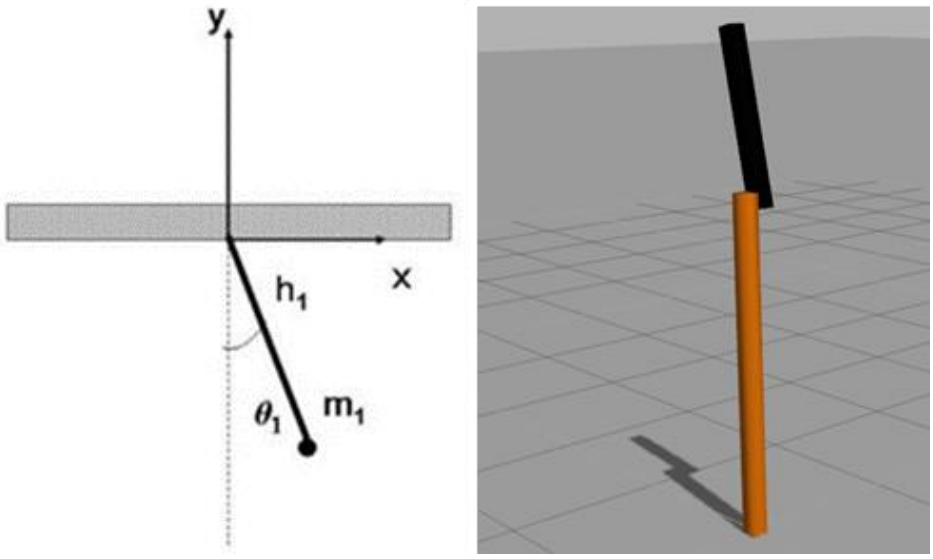


Figure 53. An inverted pendulum and its Gazebo representation.

Although Gazebo is a good simulator, it is interesting to note that a pendulum starting in the upright position (θ_1=180°) in the absence of a feedback controller will eventually fall due to the accumulation of numerical error. Table 13 shows the parameters of the inverted pendulum Gazebo simulation.

Table 13. Parameters of the inverted pendulum simulation

Link length, h_1	1.0 m
Link mass, m_1	1.0 kg
Velocity damping ratio*	0.7 [uncertain units]
Starting position	θ_1=0°
I_{xx}, I_{yy}, I_{zz}	1.0 kg*m2
I_{xy}, I_{xz}, I_{yz}	1.0 kg*m2
*Note: the author was uncertain how this Gazebo parameter was implemented, so it was neglected in the controller model	

Dynamics

Widnall [2009] uses the Lagrangian to derive the equation of motion for the system. The Lagrangian is:

$$\mathrm{L} = T - V = \frac{1}{2} m_1 h_1^2 \dot{\theta}_1^2 - m_1 g h_1 cos\theta \tag{5.1}$$

Where the first term (T) represents the kinetic energy of the system and the second term (V) represents its potential energy. The formula to calculate the equations of motion from the Lagrangian is:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_i}\right) - \frac{\partial L}{\partial x_i} = Q_i^{NC} \tag{5.2}$$

Where x_i is a "generalized coordinate" and Q_i^{NC} is a non-conservative "generalized force." In this case, the only generalized coordinate is θ and the only generalized force is the torque on the joint, τ. After applying Equation 5.2, the equation of motion for the system is obtained:

$$m_1 h_1^2 \ddot{\theta}_1 + m_1 g h_1 sin\theta_1 = \tau \tag{5.3}$$

Faking a First-Order System with a Second-Order System

Equation 5.3 can be re-arranged into state-space form:

$$\ddot{x} = \frac{\tau - m_1 g h_1 sin(x)}{m_1 h_1^2} \tag{5.4}$$

Where $x \equiv \theta$. However, the goal is an angle, not an angular velocity. It is possible to capture the dynamics of the system while allowing for an angular setpoint by reformulating as a second-order system:

$$\dot{x}_1 = x_2 \tag{5.5a}$$

$$\dot{x}_2 = \frac{\tau - m_1 g h_1 sin x_1}{m_1 h_1^2} \tag{5.5b}$$

Where

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \equiv \begin{bmatrix} \theta_1 \\ \dot{\theta}_1 \end{bmatrix} \tag{5.6}$$

Note how the system is a single-input double integrator. As discussed in Section 4.5 *Comparison with PID controllers*, Corollary 2 does not apply to this type of system (since, in this case, $\frac{\partial f_1}{\partial u} = 0$). We use the switched Lyapunov controller anyway, making use of a unit weighting trick and a particular method of partial derivative calculation that is suitable for single-input integrators. Section 7.2 discusses mixed units in greater detail.

This second-order system requires both an angular velocity and an angle setpoint. We de-emphasize the angular velocity setpoint and place extra emphasis on the angular setpoint by weighting the states. In this case, we took the measurement of angles in millirads and the angular velocity measurements in rad/s. This makes a velocity error of ~1rad/s almost negligible in comparison to an angular error of ~1000millirads, for example, so the angular error is corrected much more quickly and the controller behaves similarly to a first-order controller.

Simulation Architecture

The inverted pendulum demonstration can be cloned from a repository [Zelenak, 2015, "Lyap_Pend_Demo"] along with instructions on its usage [Zelenak, 2015, "lyap_control Package Summary"]. Figure 54 was auto-generated by the ROS rqt_graph command and shows the program architecture. Nodes are represented as rectangles that contain ovals, while ROS topics are represented as stand-alone rectangles whose name begins with a '/'. The demonstration is based on a Gazebo demonstration that introduces PID controllers [Open Source Robotics Foundation, 2014]. That original demonstration comprised the *gazebo*, *rrbot*, and *robot_state_publisher* nodes. The *gazebo* node starts the simulator and reads the parameters of the pendulum from a collection of xacro and urdf files. The *rrbot* node launches the PID controllers and monitors the joint angles in Gazebo. Finally, the *robot_state_publisher* makes the joint angles available to other nodes in the ROS ecosystem. The other nodes in Figure 54 are ancillary; the *rosout* node is almost always present for any ROS program because it handles console logging and the *rqt_gui* node is responsible for generating the figure.

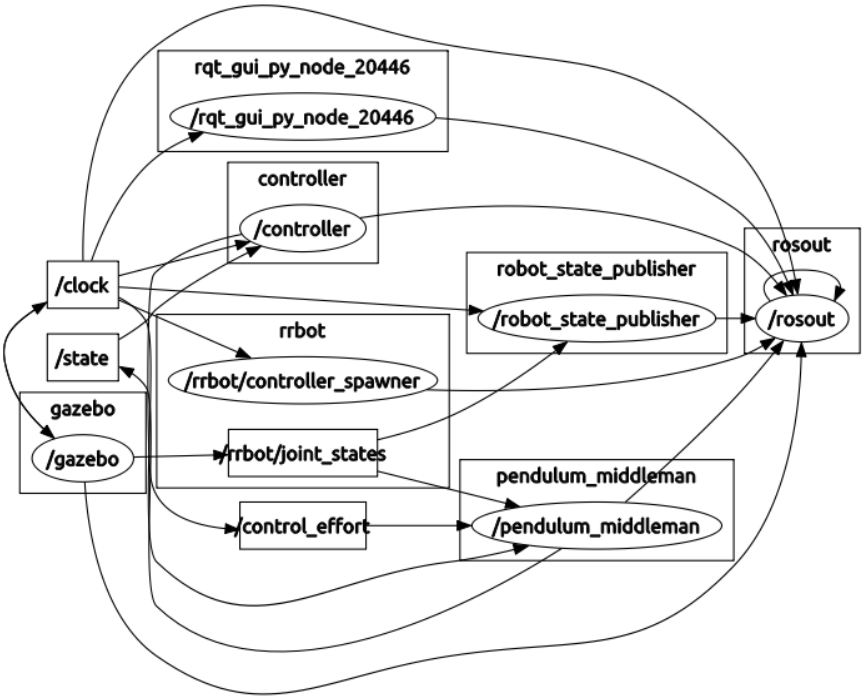


Figure 54. Graph of the pendulum controller.

To modify this simulation to use a switched Lyapunov Controller, two additional nodes were added. The controller node encapsulates the dynamic model of the system (Equation 5.5) and it performs the control effort calculations that stabilize the system. In order to feed this controller information on the state of the system, the pendulum_middleman node was added. It gathers information on the state of the system from two topics (/clock & /rrbot/joint_states) and converts them to a /state message type that the controller can understand. It also receives the /control_effort message that the controller has calculated and converts it to a message type that is compatible with the rrbot node.

The entire system is asynchronous, meaning the various nodes may run at different loop rates. However, the controller and pendulum_middleman nodes should run at an equal or faster rate than the other nodes. Thus, there is always a fresh /control_effort message

waiting to be sent to the simulation. Although Figure 54 appears convoluted, ROS provides many tools (such as *rostopic*, *rqt*, and *roswtf*) that make it easier to break down what is occurring within the ROS ecosystem. The task of understanding this system architecture without those tools would have been extremely difficult.

The parameters of the switched Lyapunov controller are shown in Table 14.

Table 14. Parameters of the pendulum controller

$\dot{V}_{target,initial}$	-140 dB
Motor torque saturation	±100 N*m
Nominal control rate	10 kHz
Lyapunov switching threshold	0.1

Simulation Results

It was noted previously that the switched Lyapunov controller's performance is highly dependent on the simulation time step and this inverted pendulum simulation was no exception. It was found that a Gazebo time step of 0.1ms was sufficiently small to yield good performance, yet large enough to run close to real time (real time factor of ~0.7). Simulation time steps larger than 1ms were not stable. The sequence in Figure 55 captures how the pendulum swung up, overshot the angular setpoint by approximately 85°, oscillated about the setpoint for several cycles, then settled with a steady-state error of approximately 5°. In this case, it behaved like an underdamped system. Figure 56 quantifies its behavior.

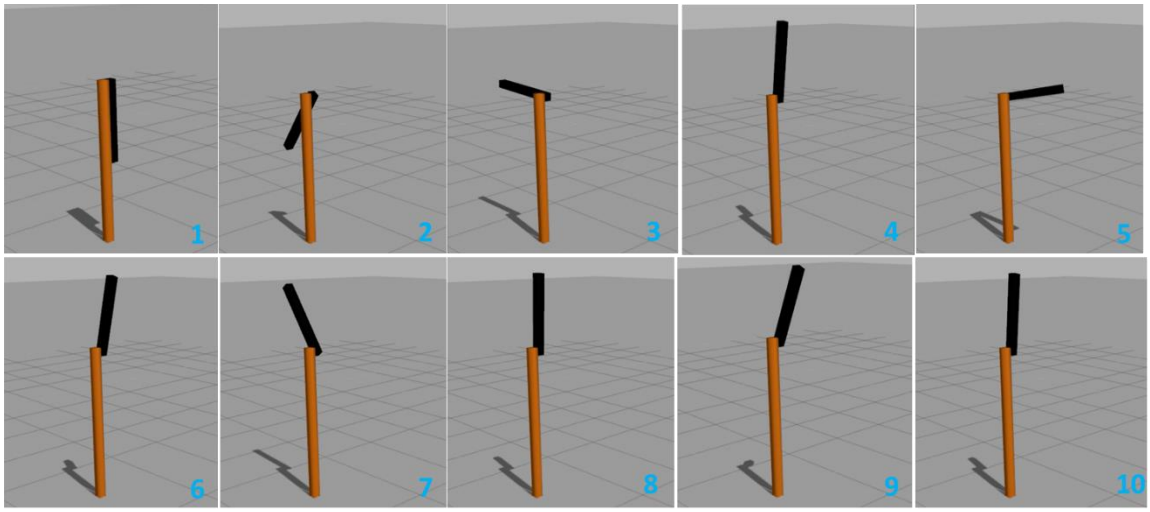


Figure 55. Pendulum swing-up sequence.

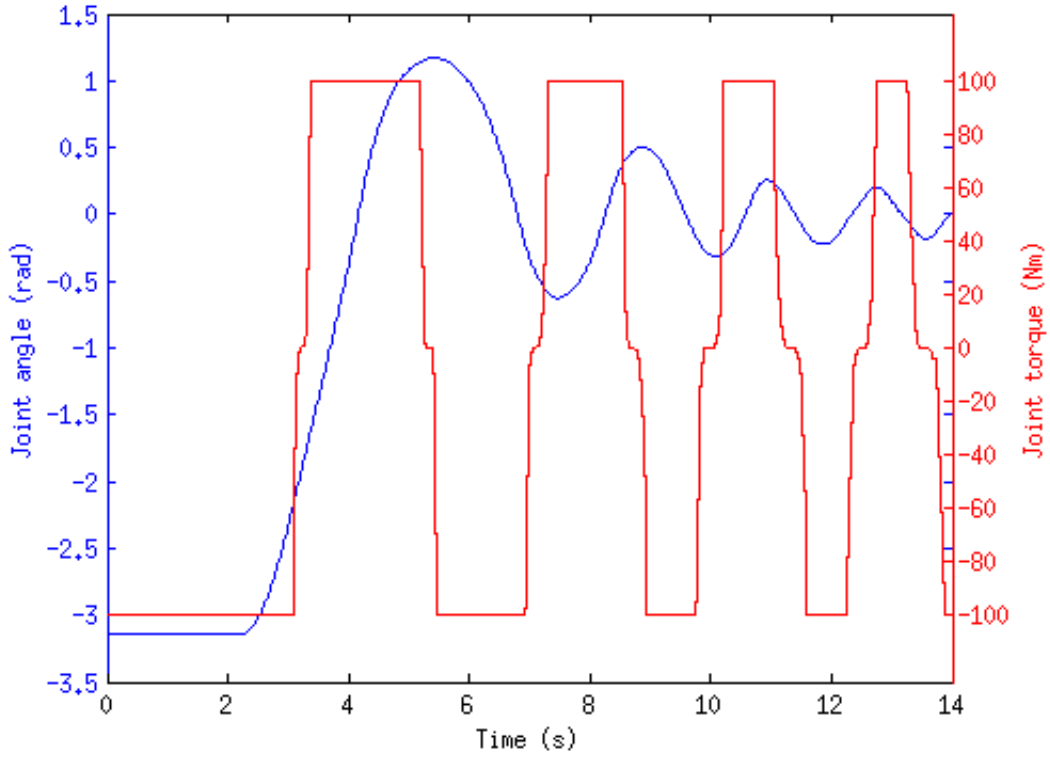


Figure 56. Pendulum axis angle and control effort plots.

Subsequent to the simulation that is shown in Figure 55, it was discovered that the publishing rate of the joint angles from the Gazebo simulator was only 50 Hz, although it was set at 10kHz. The 'rrbot' node appears to limit this publishing rate to a maximum of 100Hz. Increasing the joint angle publish rate from 50 to 100 Hz reduced the angular overshoot to 57°, which is a significant performance improvement relative to what is shown in Figure 55. The three most likely reasons why the controller's performance was not better are:

- The slow joint angle refresh rate due to the 'rrbot' node, and
- The controller's aforementioned difficulty with integrators (see Chapter Four, *Simulation 10: Third-Order PID Controller*), and
- Neglect of velocity damping in the controller's model of the system.

Running the same simulation in the MATLAB GUI (with identical controller parameters) yields the results shown in Figure 57 and . (Compare to Figure 56.)

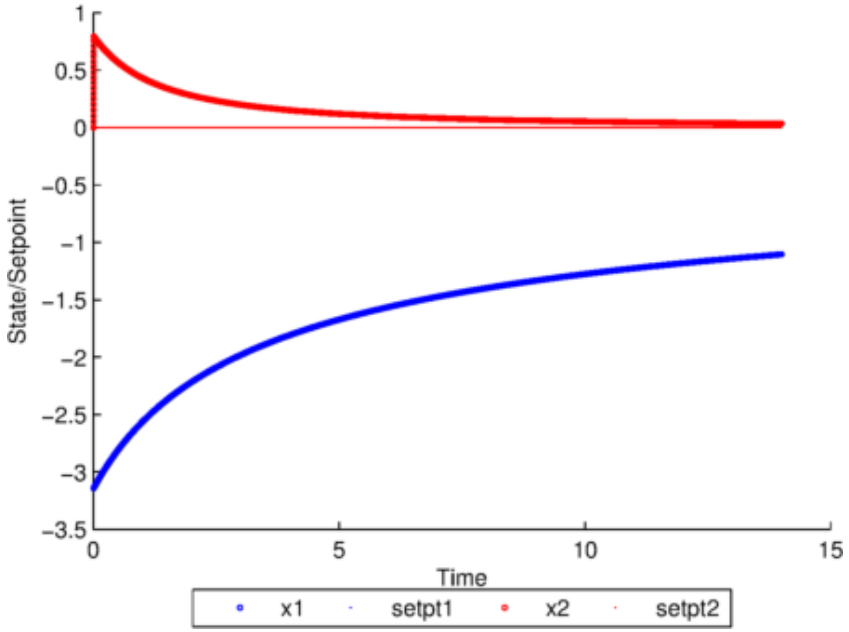


Figure 57. Pendulum stabilization with MATLAB controller.

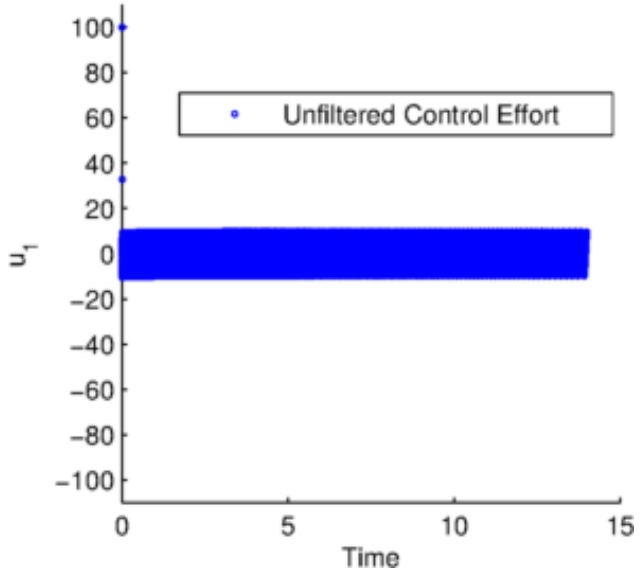


Figure 58. Control effort for the inverted pendulum.

By most metrics, the MATLAB controller performed better. The Lyapunov value tapers exponentially, as it was designed to do, and there is no chatter. The differences were possibly due to three factors:

- The MATLAB algorithm has an option for a more sophisticated calculation of partial derivatives. With this option enabled, it "looks ahead" for one extra time step, compensating for the double integrator (i.e. for $\frac{\delta f_1}{\delta u} = 0$). This option was not implemented for the ROS controller.
- The Gazebo simulator included viscous damping. Since it was unclear how the damping was implemented, the ROS controller model did not include it. However, the MATLAB plant simulation did not include viscous damping, so the plant matched the model perfectly.
- The MATLAB controller was not limited to a 100 Hz rate like the ROS implementation was.

Due to the small time step, the MATLAB controller ran 38.6 times slower than real time. The alternative CLF was never required.

5.4 COMPLIANT ROBOT DEMONSTRATION

A more challenging application of the controller was the motion control of a software-compliant robot. In robotics parlance, compliance is the term for physical flexibility [Mason, 1979] [Hogan, 1984]. A compliant robot will deviate from its path when an external force is applied to the robot; this is important for the alleviation of large force errors that arise from small positional inaccuracies in stiff systems. A classic example is the insertion of a peg into a hole. If the peg is slightly misaligned during insertion, excessive forces could damage the parts or the robot itself. The software implementation of

compliance is one approach that could allow humans to work safely in the proximity of large, stiff industrial robots. Compliance is often implemented by forcing the robot to behave as a spring.

This demonstration was performed on a Motoman SIA10D industrial robot as shown in Figure 59. Having seven joints, this manipulator is similar to a human arm in size and dexterity, although it has a larger payload than most human arms (10kg), moves faster, and is much more repeatable. An end-effector-mounted ATI Gamma force-torque (F/T) sensor monitors the externally-applied wrench on the robot and the controller acts to move the end-effector in a manner that reduces the applied wrench. (A wrench is a spatial generalization of forces that includes torques.)

Figure 59. Motoman SIA10D.

The six-dimensional wrench measurement from the F/T sensor is converted to a displacement with a spring law. This displacement is added to the nominal trajectory of the robot then sent to the controller as an updated setpoint. It calculates a Cartesian velocity

vector that will stabilize the system. The controller's dynamic model is a very simple sixth-order, uncoupled system:

$$\dot{x}_i = u_i \; \forall \; i \in \{1,2, \ldots 6\} \tag{5.7}$$

Where the translational components of $\dot{x}_i$ and u_i are measured in [cm/s] and the the rotational components are measured in [μrad/s]. Then a seven-dimensional joint velocity vector is calculated via multiplication with the pseudoinverse of the robot's Jacobian. Finally, the joint velocity command is sent to the robot. This process is illustrated in Figure 60. A ROS framework [Sucan, 2014] handled the more difficult calculations such as the Jacobian and the transformation of end-effector forces into the world frame. The parameters of the simulation are given in Table 15.

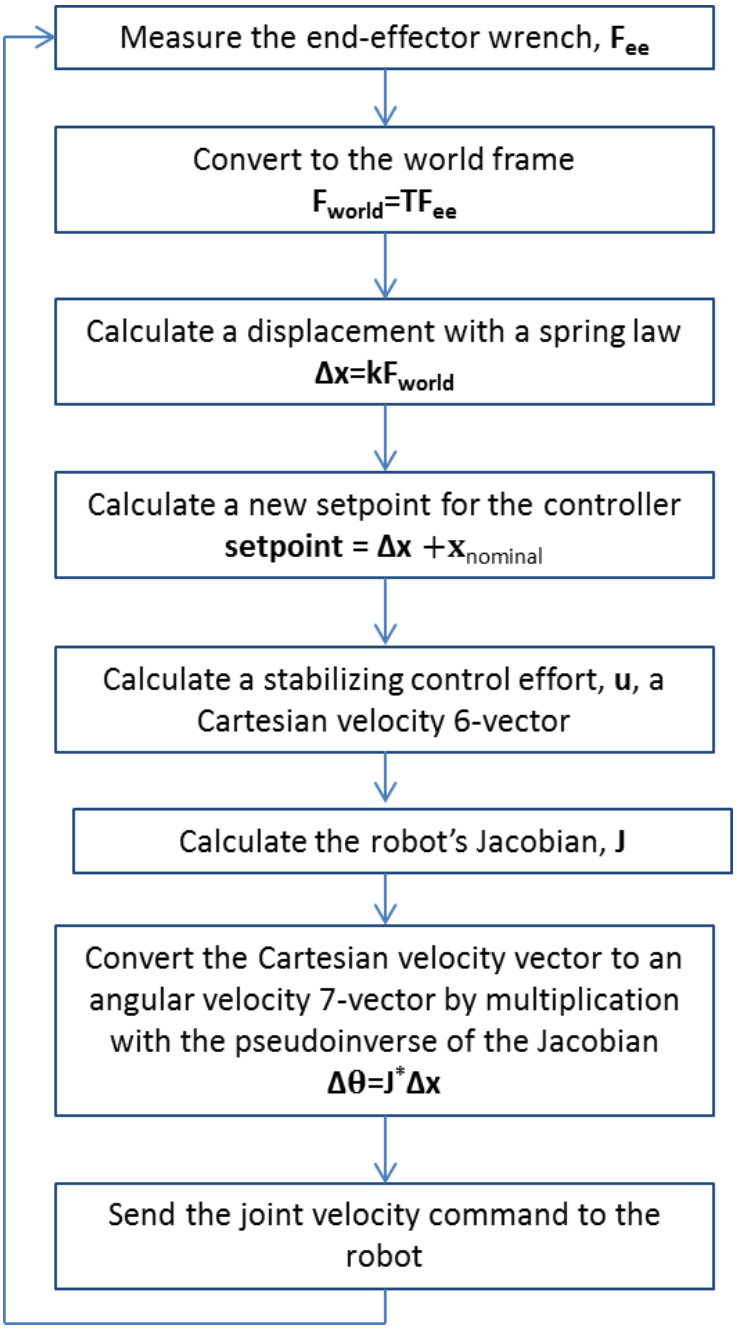


Figure 60. Flowchart of the compliance controller.

Table 15. Parameters of the compliance demonstration

Compliance Parameters	
Translational Spring Constants	$[0.01\ 0.01\ 0.01]\ \frac{s^2}{kg}$
Rotational Spring Constants	$[0.1\ 0.1\ 0.1]\ \frac{mrad*s^2}{kg*m^2}$
Nominal Trajectory	Stationary
Controller Parameters	
Time Step	0.05s
Translational Control Effort Saturation	±[1 1 1] cm/s
Rotational Control Effort Saturation	±[175 175 175] μrad/s
Aggressiveness, $\dot{V}_{target}(0)$	-10^5 rad^2
V_2 Step Sharpness, β	0.5
Switching Threshold	0.1

As Table 15 shows, the communication interface with the robot was limited to just 20 Hz. Section 3.1 discusses how linearization makes the switched Lyapunov controller susceptible to poor performance at slower loop rates. Regardless, the system model was very simple and uncoupled in this case, so the controller's performance was adequate. Its performance in three representative dimensions is captured in Figure 61.

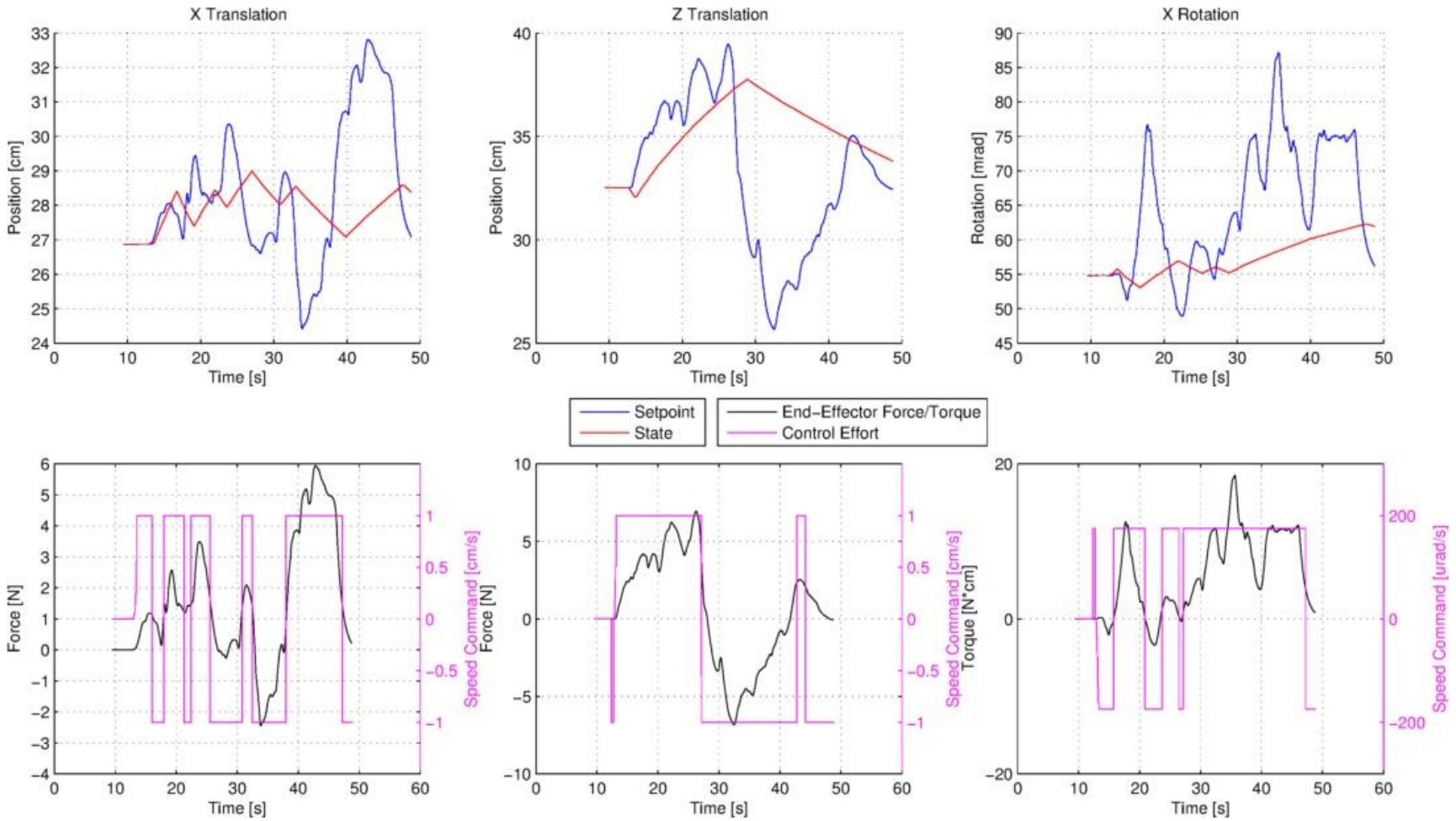


Figure 61. Plots from the compliance demonstration.

From Figure 61, it is clear that the controller is tracking the displacement calculated from the compliance law and the control efforts are almost constantly saturated. This is a consequence of the slow control frequency and a rather large $\dot{V}_{target}(0)$.

5.5 Simulation 12: Very Large System

As mentioned in Section 4.6, the final test of the switched Lyapunov control algorithm was a large and highly nonlinear dynamic system. This system was composed of all the asymptotically stabilized systems from Chapter Four, lumped into a 14^{th}-order, 15-input system (Figure 62). The system includes significant coupling and nonlinear terms as well as a broad range of coefficients, yielding a range of "slow" and "fast" dynamics. The parameters of the controller are given in

Table 16. A very small time step (1 µs) was used to ensure good results, and the resultant large quantity of data analyzed was one reason why the MATLAB GUI was not appropriate for this simulation. Over the 80ms duration of the simulation, 59 MB of data was collected.

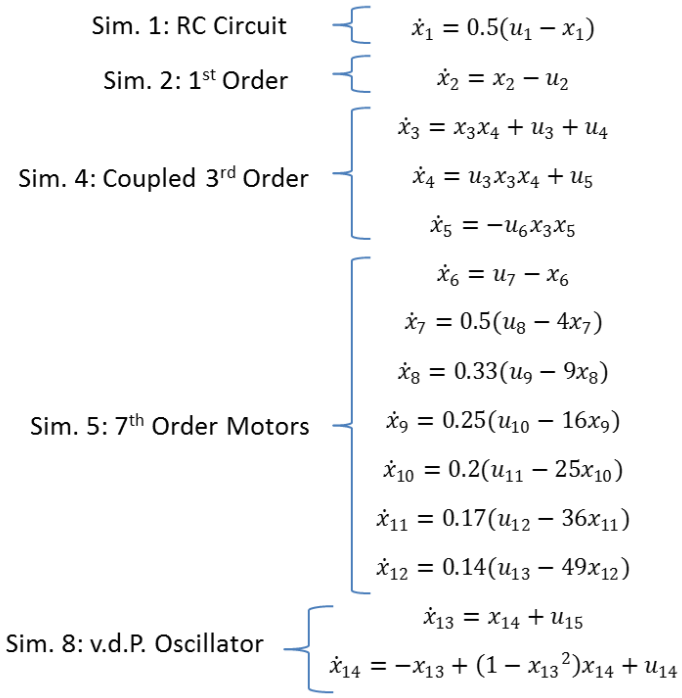


Figure 62. Annotated state-space definition of the 14th-Order System.

Table 16. Parameters of Simulation 12

	Baseline
Time step	1 μs
Control Effort Saturation	±50
Aggressiveness, $\dot{V}_{target}(0)$	-10^4
V_2 step sharpness, β	0.5
γ	1.0
Switching threshold, $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$	0.01
Initial conditions	(0.3,0.4,0.5,0.6,0.7,0.8, 0.9,1.1,1.2,1.3,1.4,1.5, 1.6,1.7)
Setpoint	1 (for all states)

The results of the simulation were good. Every state was stabilized, although there was a small steady-state errors for three of the states (10,11,12). The other eleven states were asymptotically stabilized. Both the settling times and the rise times were fast and the overshoot was less than 10% for all states. (Figure 63)

The plot of the control efforts (Figure 64) is interesting because it is clear that the control efforts were distributed across every state (as they were designed to be per Section 3.2). Unlike the smaller simulations, the control efforts were rarely saturated. Another interesting feature was the spike in control efforts around 76ms, which occurs with a switch in the dominant control effort.

Simulation 12 validates the switched Lyapunov control algorithm. The system was complex enough that a more computationally-intensive control algorithm such as Model Predictive Control or Optimal Control would typically be required, and the simulation has demonstrated that the Switched Lyapunov Controller is an alternative to such approaches.

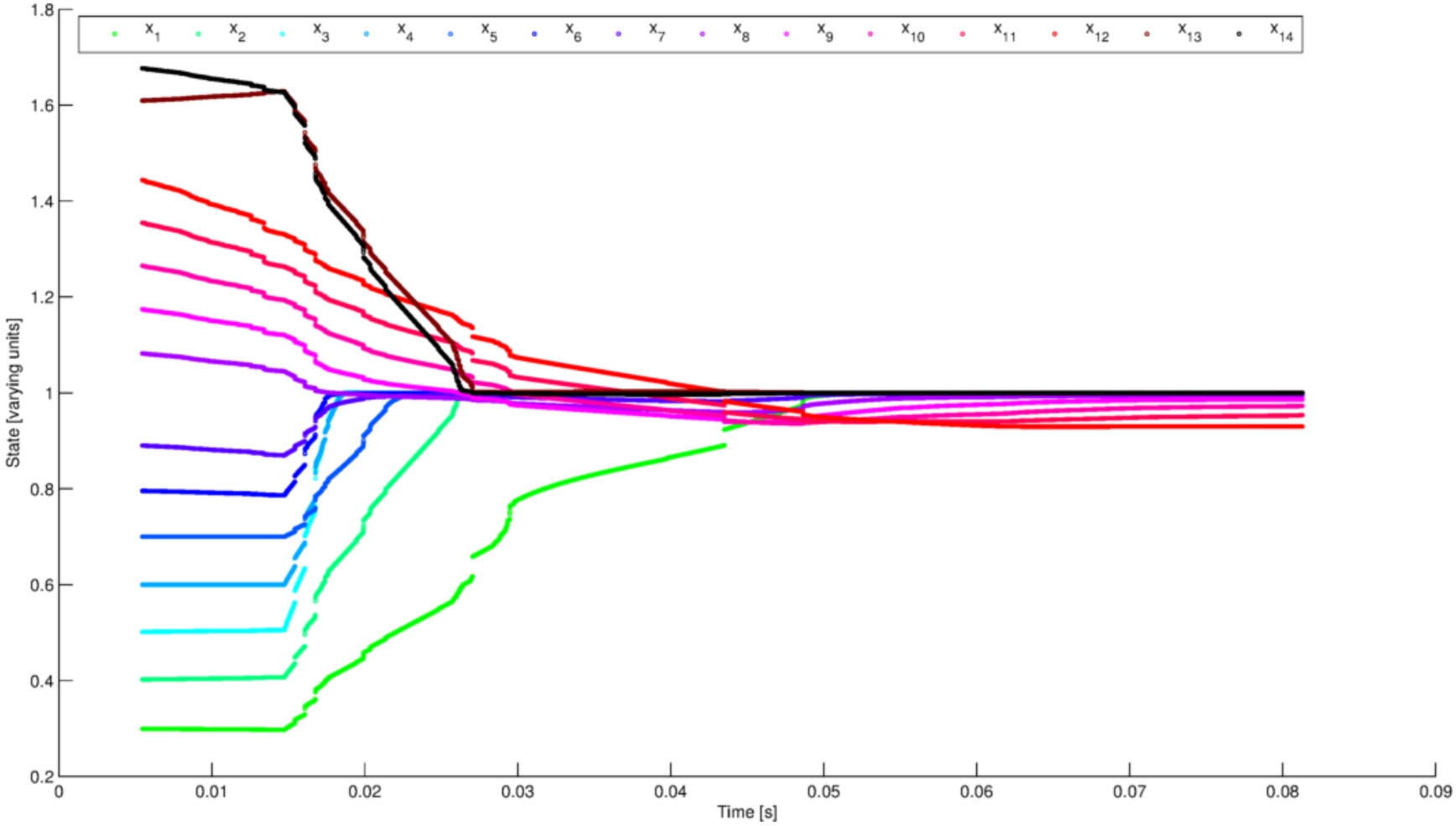


Figure 63. Trajectories of the states during the very large simulation.

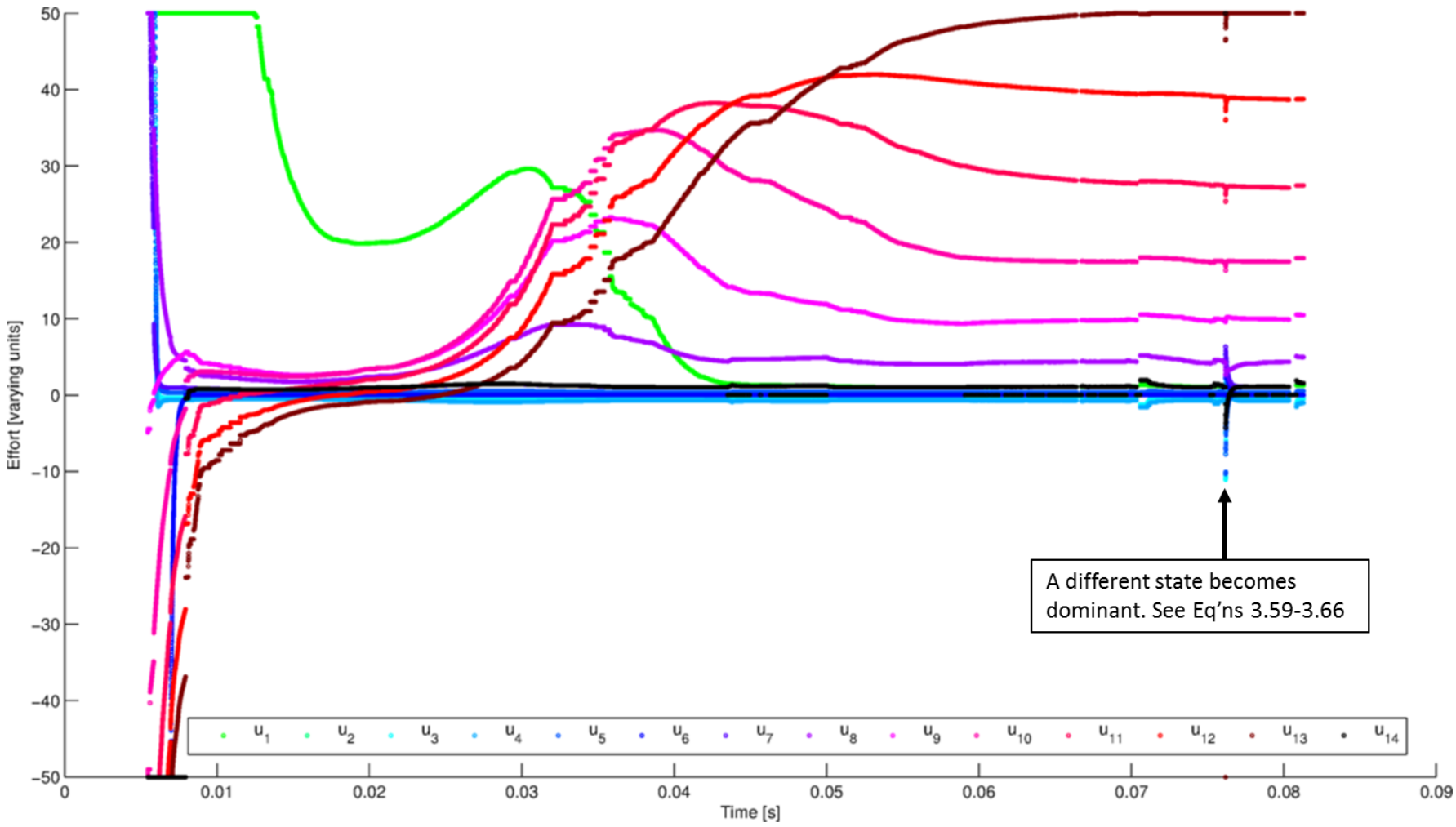


Figure 64. Control efforts during the very large simulation.

5.7 CHAPTER SUMMARY

This concludes Chapter Five which validated the switched Lyapunov controller on several practical applications and a rigorous, highly nonlinear simulation. The chapter also described the C++ software package which yielded a 550% speed improvement over its MATLAB counterpart. Our original premise from Chapter 1 suggested the switched Lyapunov controller should be easier to use than the most common techniques in industry today, so the next chapter examines that claim in a preliminary fashion: four engineering students who were recently presented to the switched Lyapunov algorithm are asked to compare it against PID control and we analyze the results. The study motivates several improvements to the algorithm.

Chapter 6: Comparison with PID Control

One of the premises of this dissertation was that the current controls curriculum is unnecessarily confusing for engineering students. To test that hypothesis, four students who had recently completed a typical introductory undergraduate controls class were asked to perform a comparison between the switched Lyapunov controller and PID controllers. The intention was to compare the ease of use and the performance against the industry-standard, linear PID controller which the students had recently learned. The exercise revealed some interesting strengths and weaknesses of the switched Lyapunov controller and motivated several improvements. It also revealed unexpected deficiencies in the students' understanding of the PID algorithm.

6.1 Description of the Study

The four students (one undergraduate and three Master's students) had recently completed an undergraduate controls class at The University of Texas at Austin (a top ten mechanical engineering program). With a several-paragraph introduction to the switched Lyapunov algorithm, they were first asked to tune a switched Lyapunov controller on the second-order system:

$$\dot{x}_0 = x_0 + u_0 \tag{6.1a}$$

$$\dot{x}_1 = x_1 - 100u_1 \tag{6.1b}$$

Then they were asked to tune a PID controller for each individual state. Notice that the u-coefficient is positive in 6.1a while it is negative and much larger in 6.1b. This was intentional; it was designed to test that the students truly understood the function of each

PID parameter. The reference trajectories were a chirp and a linearly increasing displacement:

$$x_{0,ref}(t) = \sin(4t^2) \tag{6.2a}$$

$$x_{1,ref}(t) = t \tag{6.2b}$$

The comparison was peformed with ROS packages [Zelenak, "lyap_control Package Summary"] [Zelenak, "PID Package Summary"] and the written guidelines for the students and their responses are available in Appendixes A/B/C, but the most insightful questions included:

- What were the final, tuned parameters for each controller (Lyapunov, PID #1, and PID #2)?
- How long was spent tuning each controller?
- Did you notice any performance differences between the Lyapunov controller and the PID controllers?
- Was it easier to stabilize both states at once with the Lyapunov controller, or was it easier to stabilize each state individually with two PID controllers?
- If you had a system with 5 states, would you rather tune five separate PID controllers? Or would you rather stabilize all five simultaneously with a Lyapunov controller?
- Was it easy to recall how to tune a PID controller?

The brief description of the switched Lyapunov controller they were given was:

The Lyapunov controller is based on the 1892 work of a Russian mathematician. The idea is to move along a bowl-like surface where the setpoint is

at the bottom of the bowl (see Figure). You will define the parameters that determine how quickly the system moves towards the bottom of the bowl.

Open catkin_ws/src/pid_lyap/src/lyap_controller.cpp in a text editor. Notice lines 14-16. These are the only 3 lines that you're allowed to change for this exercise. $\dot{V}_{target,initial}$ *determines how quickly the system moves towards the bottom of the bowl. It must be a negative value. A very large negative value moves faster down the bowl, but if it is too negative, the system might overshoot or become unstable. You need to specify a good value. (The default -1.0 is not negative enough.)*

The high_saturation_limit defines the largest possible u_i*'s. In the motor example, this is like specifying a maximum current to a motor. You need to specify good values for* u_0 *and* u_1*. Again, the default values will not work very well.*

The low_saturation_limit defines the smallest possible u_i*'s. In the motor example, this is like specifying the most negative current to a motor. You need to specify good values for* u_0 *and* u_1*. Again, the default values will not work very well.*

6.2 RESULTS OF THE STUDY

Table 17 provides a concise summary of the numerical results. Typical graphical results are given in Figure 65-Figure 67.

Table 17. Controller comparison results

	Switched Lyapunov Controller	PID #1 and PID #2		
Average Tuning Time	1.78h	0.45h (for PID #1 and #2 combined)		
Mean Controller Parameters	$\overline{\dot{V}_{target}}(0) = -3.2E8$ $\bar{u}_{sat} = \{13000, 3.5\}$		PID #1	PID #2
		$\bar{K}_P$	360	-6.0
		$\bar{K}_I$	0.41	-1.9
		$\bar{K}_D$	0.13	0.00
Range of Controller Parameters	$\dot{V}_{target}: -1E6\ to\ -1E9$ $u_{sat}: \pm\{100, 0.3\} to\ \pm\{5E4, 5\}$		PID #1	PID #2
		K_P	30 to 1000	-1.0 to -20
		K_I	-0.15* to 1.7	-8 to 0.5*
		K_D	-0.45* to 0.9	-0.005 to 0.0005*
*Indicates an incorrect sign.				

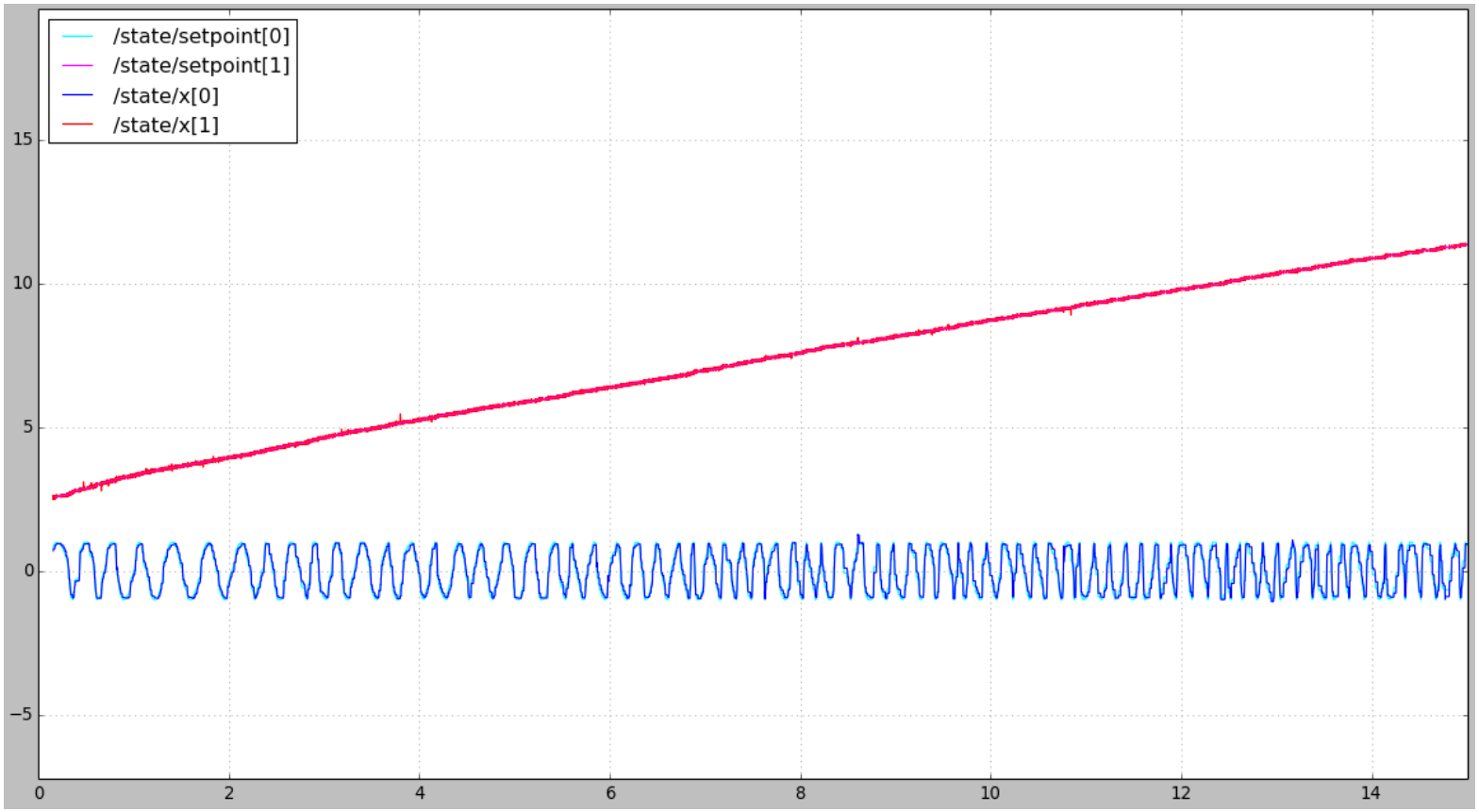


Figure 65. Tracking both states with a switched Lyapunov controller.

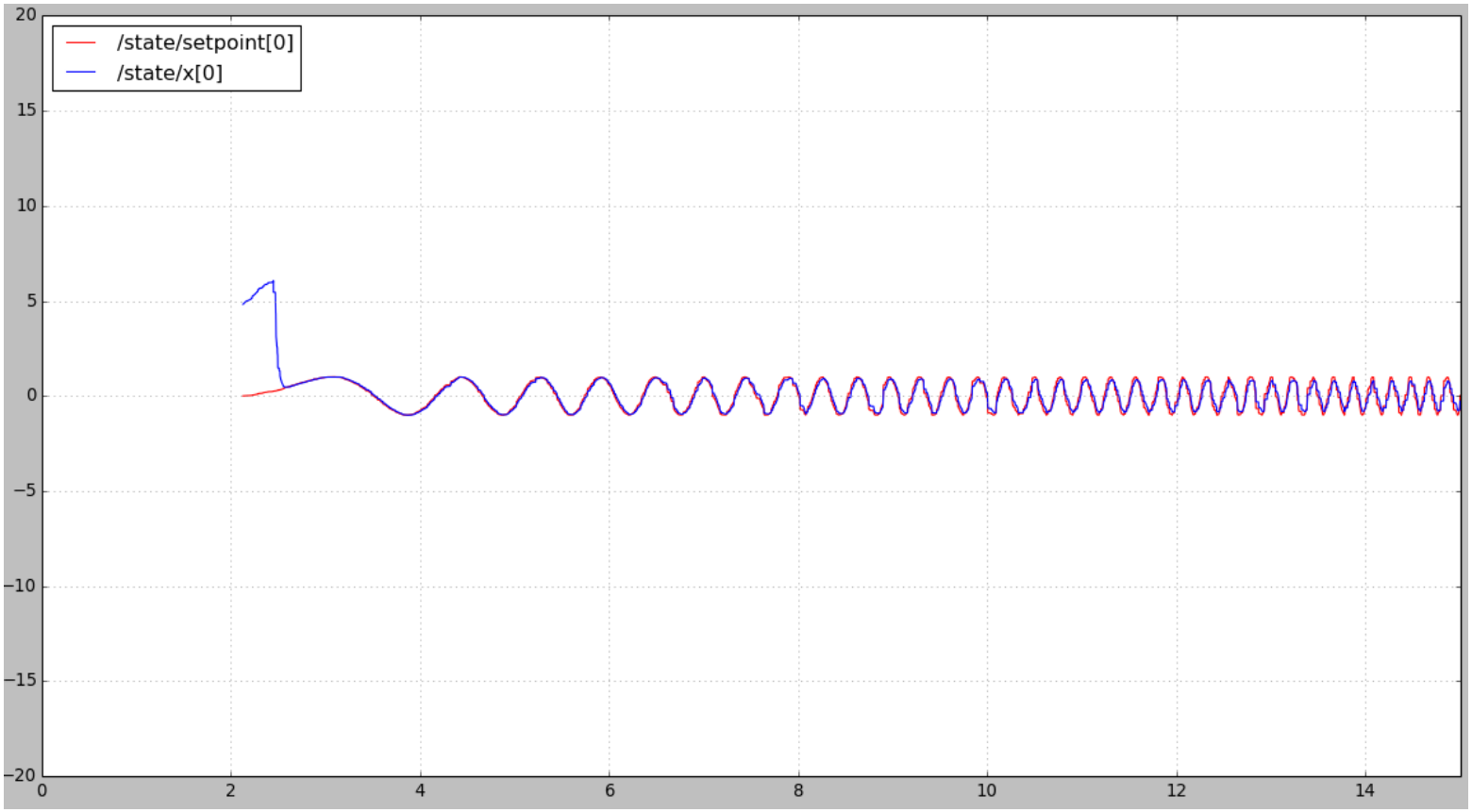


Figure 66. Tracking state 0 with a PID controller.

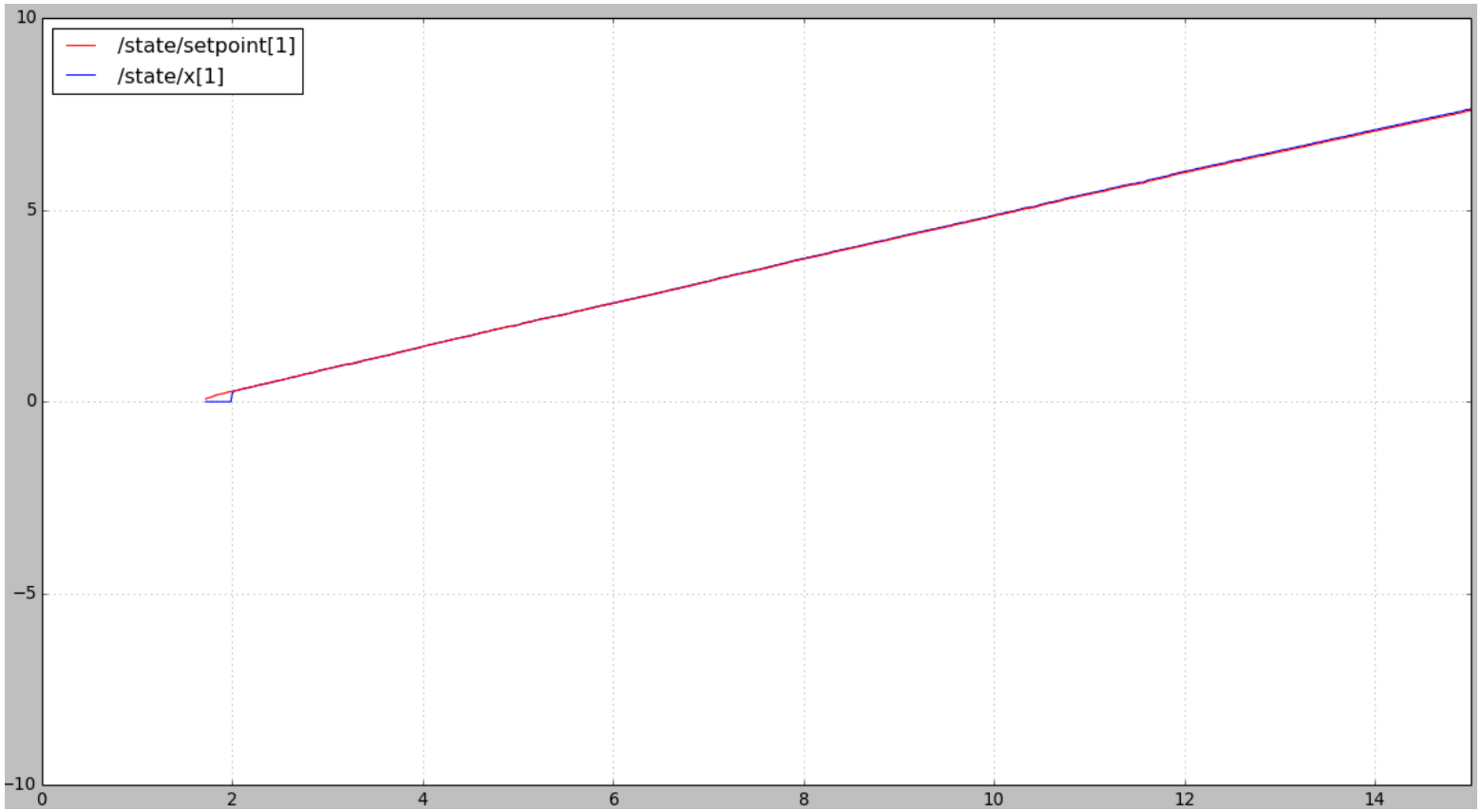


Figure 67. Tracking state 1 with a PID controller.

Observations on the Switched Lyapunov Controller

For all four students, tuning the switched Lyapunov controller was more time-consuming than tuning two separate PID controllers. Even for a fifth-order system, two of four said they would prefer five separate PID controllers. It is difficult to say whether that opinion would change if they had an equal amount of prior experience with each type of controller. Two students expressed the expected opinion, i.e. that the Lyapunov controller would be preferred for higher-order systems:

> *I think that with more and more states, the Lyapunov controller would become more appealing. And it felt like, with practice, the Lyapunov became a little more intuitive. So with five states I would probably rather tune a Lyapunov controller.*

And,

> *It was easier to tune both states at once versus each state independently. It was difficult to know the correct range numbers needed for tuning the parameters due to lack of experience with this controller.*

One student was particularly confused by the Lyapunov controller while the other three figured it out from a several-paragraph explanation. The crux of the confusion was the very large $\dot{V}_{target}$ which was required for asymptotic stability. This large magnitude is required because of the tapering of Equation 4.2. It seems this confusion would have been resolved with a better explanation of the parameters beforehand:

> *Because I have never worked with this controller, I lacked the intuition needed to understand how altering the parameters would affect the graph. I had to ask another student (after being stuck for 1.5 hours) why state x_0 was not following the setpoint and he basically explained $\dot{V}_{target}$ had to be ridiculously high.*

And,

I didn't really understand what each parameter did based on your description of them.

As for performance, two students did not notice any performance differences between the Lyapunov controller and the PID controllers. Another student made the insightful observation that, since the Lyapunov controller included saturation limits but the PID controllers did not, the Lyapunov controller will eventually fail to track the linearly increasing setpoint of x_1. We argue this is not a practical concern because a hardware PID controller would require saturation limits as well. In other words, the simulation was not realistic in this regard. Finally, the fourth student noted, "The PID controllers tended to track more smoothly once they were tuned correctly." (Refer to Figure 65 vs. Figure 66.)

6.3 A GAP IN COMPREHENSION OF THE PID ALGORITHM

A surprising conceptual deficiency was observed regarding signs of the PID gains. K_i or K_d gains with incorrect, destabilizing signs were used four times and only one student avoided this type of error. Furthermore, it did not matter whether the coefficients of the state-space equation were positive or negative, as the error was made with equal frequency on Equation 6.1a (which has a positive u-coefficient) and Equation 6.1b (which has a negative u-coefficient).

Examining Equation 6.1b in particular, all four students realized the proportional gain must be negative but two did not realize that K_I and K_D should also be negative. One student, in particular, noted how the sign change was confusing:

"I was seriously thrown off by the negative proportional gain in the last section because I thought that gains were always positive."

The other student who made the sign error was misled by intuition:

"For the most part, I had an intuition on how to tune it."

This is a particularly dangerous mistake because it could lead to rapid and unexpected destabilization. For example, the error integral could grow slowly larger until the destabilizing integral feedback suddenly overcomes the stabilizing proportional feedback. Furthermore, this type of error could be difficult to detect in simulation; the study guidelines only required the simulation to last ten time units, which was not long enough to accumulate a destabilizing error integral.

Stability Analysis of the Tuned PID Controllers

A formal mathematical analysis reveals the consequences of integral and/or derivative terms with the incorrect sign. This analysis is based on the roots of the characteristic equation, i.e. the "poles" of the system's closed-loop transfer function. A

linear closed-loop system is stable for arbitrary, bounded reference inputs if all poles have negative real parts [Dorf & Bishop, 2008, pg. 359]. From the student responses, the poles closest to the origin (including the two unstable poles) are shown in Figure 68. Clearly two of the students' controllers might potentially destabilize the system.

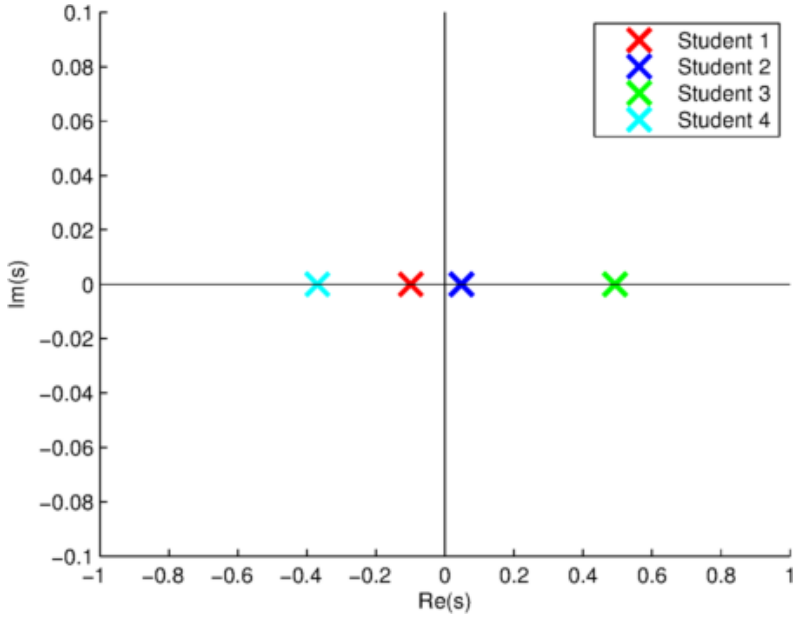


Figure 68. Two unstable poles in the right half-plane would lead to potentially unstable controllers.

None of the students took time to analyze the poles of their PID transfer functions. If they had, it is likely that the time savings versus the Lyapunov approach would have been nullified. Figure 69 is an example of how Student Three's PID controller can become unstable if the integral of the error becomes large enough. In this case, the large error integral develops because the initial error (setpoint minus $x_1(0)$) is very large.

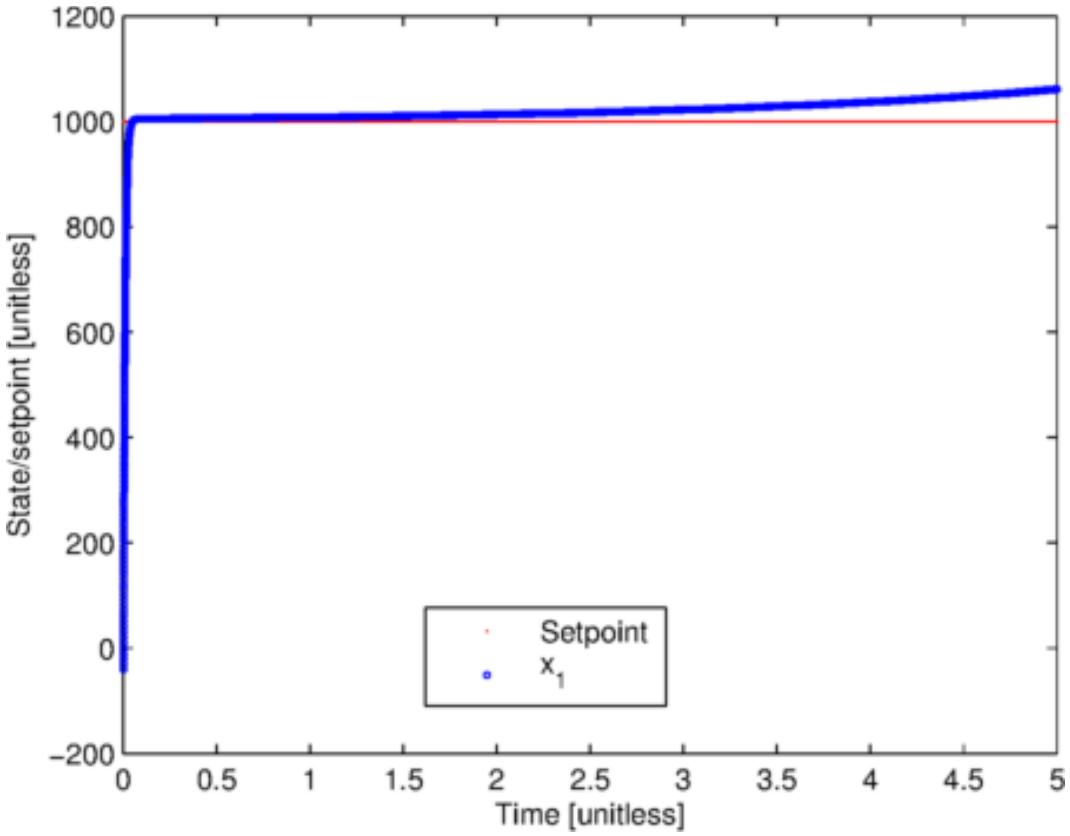


Figure 69. An example of unstable PID control.

To avoid this issue in the future, controls educators should ensure their students are familiar with dynamic systems having both positive and negative coefficients. Additionally, validation of any PID algorithm should include long-running simulations, simulations with large error terms, and the simulation of abrupt setpoint swings that would detect destabilizing errors in the integral and derivative terms. A simple rule of thumb that would help students avoid sign errors is, "All three gains should have the same sign." Furthermore, PID software packages should include sign checking of the gains.

By contrast, it is impossible to induce such latent instability with the switched Lyapunov controller. The tunable parameter $\dot{V}_{target}$ must always be negative and both software packages that we have released include error checking on the parameter signs.

6.4 CHAPTER SUMMARY

Major conclusions from the preliminary PID-Lyapunov feedback are:

- The switched Lyapunov algorithm was sometimes confusing for students who had no previous exposure to it.
- PID control is powerful when it is applied appropriately, but inexperienced engineers are likely to trust intuition over theory.
- Careless usage of the PID algorithm can introduce "latent instabilities" which are difficult to observe in simulation. A switched Lyapunov controller does not have the same drawback.
- Tuning a PID controller by intuition was faster than tuning a switched Lyapunov controller, but intuition lead to dangerous parameter sets in the majority of cases.

The next chapter examines several improvements that were proposed for the switched Lyapunov control algorithm. Several of these improvements were directly motivated by the feedback from this chapter.

Chapter 7: An Examination of Real-World Issues

The students' comments upon completion of the comparison in Chapter Six, along with the discussion in Chapters Four and Five and discussion with two experts in the field (Dr. Maruthi Akella and Dr. Benito Fernàndez), motivated the following modifications to the algorithm. The modifications make the algorithm easier to use or improve its performance in certain practical situations.

7.1 Specify aggressiveness in dB

As they worked through a tutorial based an early version of the Lyapunov control software, students expressed confusion related to the tuning of $\dot{V}_{target}$. When a very large magnitude was required (like -10^6), it was especially difficult for new users to intuit the appropriate magnitude. So we have modified the nonlinear controller to accept $\dot{V}_{target}$ in dB units, i.e. $\dot{V}_{target}(0) = -10^{\left(\frac{\dot{V}_{target,dB}(0)}{-20}\right)}$. This change was therefore applied retroactively so the entire document now reflects $\dot{V}_{target}$ in dB units. For example, a $\dot{V}_{target}$ that would have been specified originally as -10^6 is now specified as -120dB.

7.2 Isotropic error measurements

Although it was not relevant for the simple comparison against PID controllers, we observed in Sections 4.7 *Insight into controller tuning* and 5.3 *Inverted pendulum simulation* that mixed units can be troublesome for the algorithm. Roboticists face a similar problem when translational and rotational units are mixed within a matrix, or when force and torque units are mixed. Stocco [1998] has developed a technique that overcomes this obstacle (for square matrices) by converting units to percentages, assuring an isotropic matrix. Stocco's technique does not translate directly to our application since we are concerned with a scalar function *V*, but we have adjusted the MATLAB controller to implement a similar method.

There were a few methods which could have been used to convert absolute error values into unitless fractions:

- Aström [2011] suggests dividing all unit-bearing parameters in the state-space equations with an intrinsic, unit-bearing property of the system (such as natural frequency in a mechanical system). Similar to Stocco's approach, this change of variables yields a unitless state-space equation. However, this approach does not generalize well because the user must select the dividing parameters. Furthermore, it may be impossible to cancel units easily on highly nonlinear systems.
- The error for each state could be measured as a unitless fraction of the initial error. The issue with this approach is near-zero initial errors, which will be weighted extremely heavily in comparison to the other states.
- The error for each state could be measured as a unitless, positive fraction of a user-specified error, which we refer to as "nominal error." Sometimes the selection of nominal error is obvious, as in the case of the inverted pendulum (maximum angular error = 2π). In other cases, the user needs to select a reasonable value. For example, a nominal angular velocity error might be specified as the hardware's maximum. Furthermore, it is possible to select the initial error as the "nominal error" if the user deems it reasonable. This approach is similar to Stocco's method.

The third option seems the most practical, so it was implemented on the most challenging, mixed-unit simulation (5.5 Simulation 12: Very Large System). The simulation parameters were identical to those of Simulation 12 and the results from Figure 70 can be compared against Figure 63. The control efforts are plotted in Figure 71.

"Nominal error" for each state was specified as the initial error: E_{nom}= [0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]. After each state is scaled by E_{nom}, all states had the same initial error of one. This leads to isotropic weighting in $V(x)$ which is the control theory equivalent of Stocco's isotropic robot Jacobian.

Normalization produced several significant effects: the gaps in Figure 63 are missing which suggests the dominant control effort no longer switches at those points. All states still approach the setpoint asymptotically near the end of the simulation; they did not do so in Figure 63. Finally, $V(x)$ decreases more slowly. That is likely a consequence of the nonlinear effect of increasing the scale of the the error measurements while maintaining the saturation limits on the actuators.

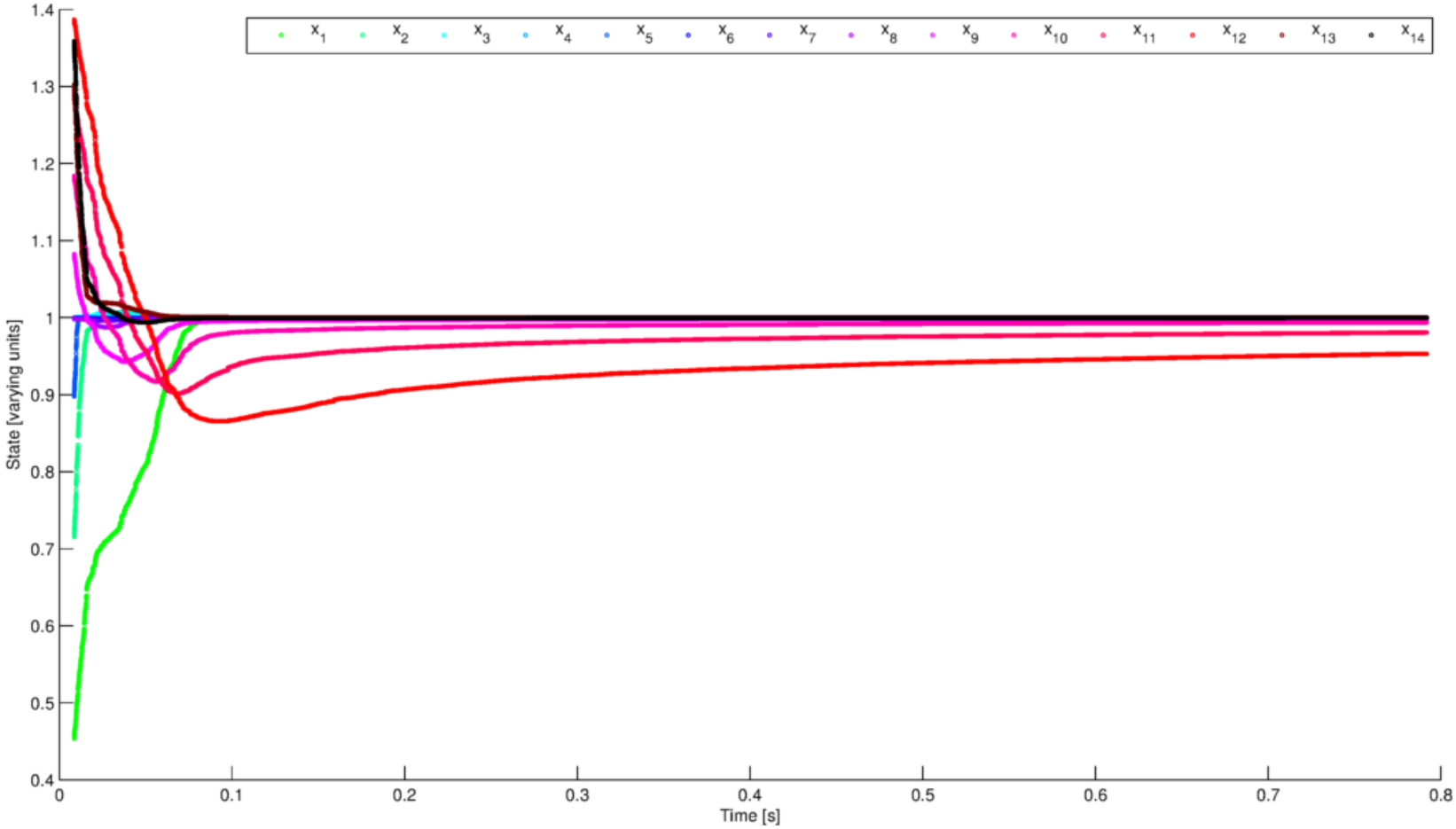


Figure 70. Trajectories of the states of the Very Large Simulation after normalization.

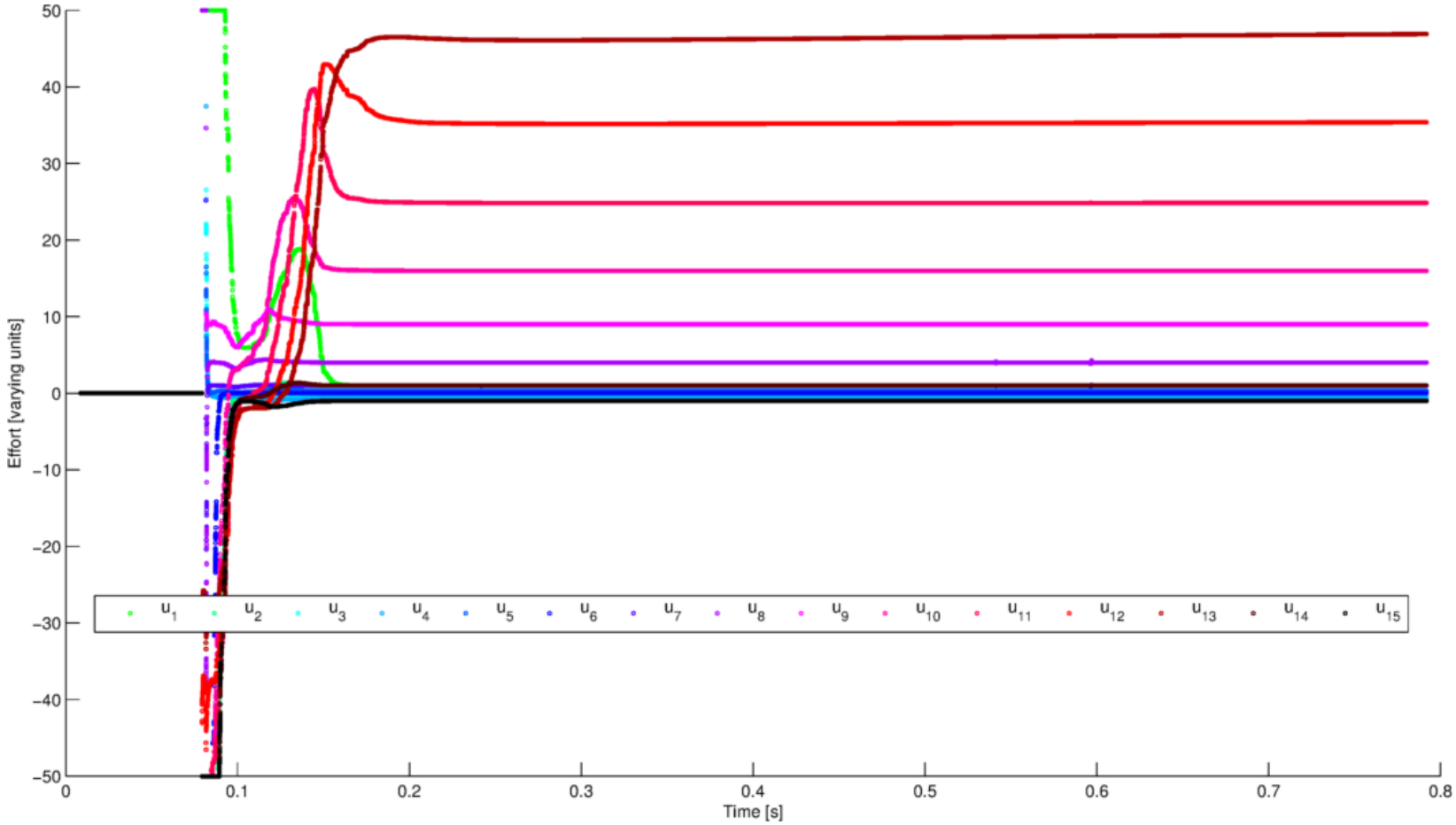


Figure 71. Control efforts after normalization.

7.3 INTEGRATION ACCURACY

It was noted that the switched Lyapunov controller, especially the MATLAB implementation, often chatters (see e.g. Section 5.2 C++ vs. MATLAB: computation time comparison). To test the hypothesis that the chatter is caused by MATLAB's adaptive integrator, the accuracy tolerances of the o.d.e. solver were tightened ("RelTol" decreased from 1e-1 to 1e-6, "AbsTol" decreased from 1e-4 to 1e-8). In the adaptive MATLAB ode23 solver we use, the results from a higher-accuracy third-order and a lower-accuracy second-order Runge-Kutta (RK) integrator are compared after every time step [Bogacki, 1989]. If they agree within the range specified by "absolute tolerance," the integration time step is maintained. If a nonlinearity causes the two RK methods to diverge, the integration time step is reduced to improve accuracy. When the integrated values are very small, the relative tolerance becomes more restrictive than absolute tolerance so it guides the time step adaptation [MathWorks, 2012]. This explains why both tolerances must be specified.

Tightening the tolerances increased the run time of Simulation 1 by 10.4% but there was a negligible performance difference (Figure 72). The blue dots appear to fill the red circles because the time series are almost identical. We conclude that the chatter is not caused by the MATLAB integrator settings.

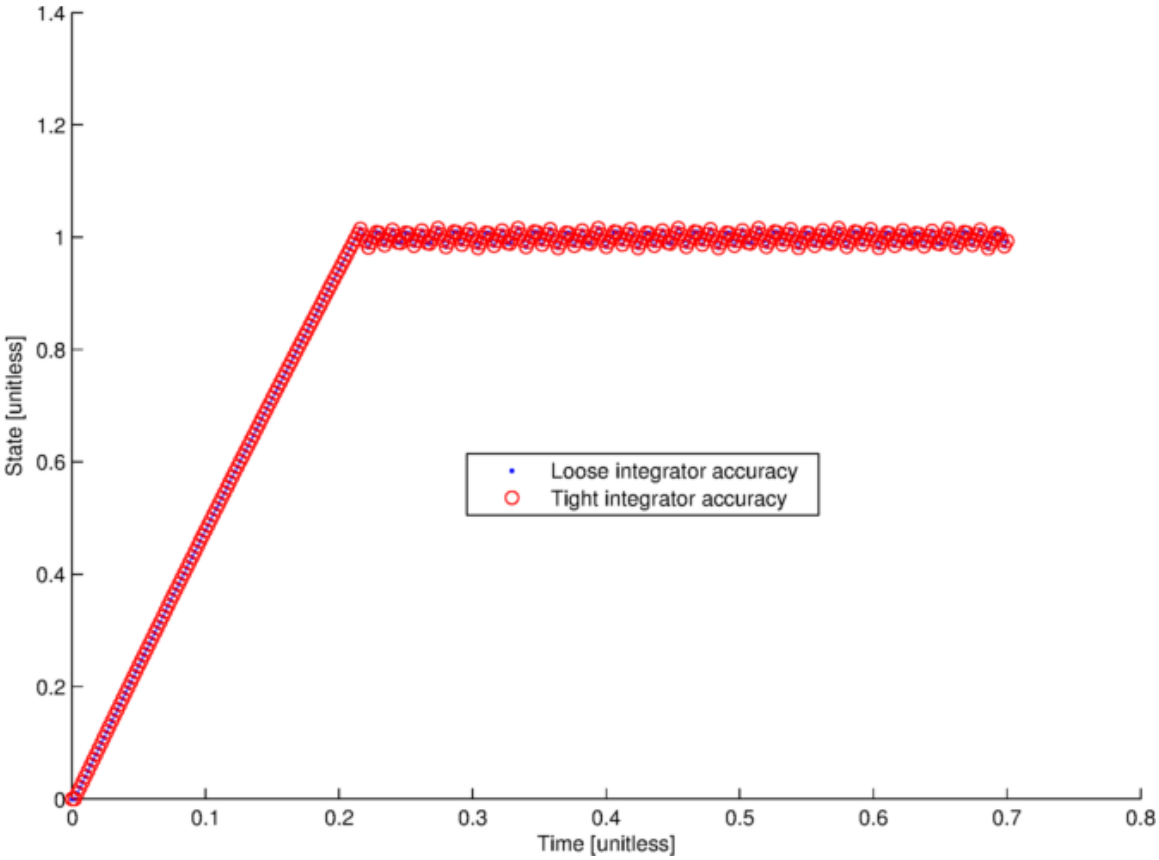


Figure 72. O.D.E. integrator settings had a negligible effect.

7.4 LINEARIZATION ABOUT THE PREVIOUS U

This section investigates a modification to the control algorithm which (it was hypothesized) would improve its performance. The modification involves linearization about u=0 (as seen in Equation 3.4 and trickling down to Equations 3.8, 3.12, 3.33, 3.36, etc.). To ensure the linearization was accurate, we required "small" control efforts. A better approach might be to linearize about the previous control effort u_{prev}, eliminating the need for a small control effort requirement. The only drawback to linearization about u_{prev} is that it negates the proof of stability in Chapter 3 and a corrected proof is – at a minimum – not trivial. (see Equation 3.21).

To test this modification, performance on another very large system was compared with and without the modification. The system is given in Equations 7.1a-o and is similar to the system in Figure 62 (with the addition of another state and another control input). Table 18 lists the controller parameters from the study. The conclusion (as visually apparent from Figure 73-Figure 74) is that the modification had a negligible effect on the controller's performance. The likely reason is that the controllers were often saturated regardless of the linearization technique. Of course, this result is not necessarily extensible to all systems.

$$\dot{x}_1 = 0.5(u_1 - x_1) \tag{7.1a}$$

$$\dot{x}_2 = x_2 - u_2 \tag{7.1b}$$

$$\dot{x}_3 = x_3 x_4 + u_3 + u_4 \tag{7.1c}$$

$$\dot{x}_4 = u_3 x_3 x_4 + u_5 \tag{7.1d}$$

$$\dot{x}_5 = -u_6 x_3 x_5 \tag{7.1e}$$

$$\dot{x}_6 = u_7 - x_6 \tag{7.1f}$$

$$\dot{x}_7 = 0.5(u_8 - 4x_7) \tag{7.1g}$$

$$\dot{x}_8 = 0.33(u_9 - 9x_8) \tag{7.1h}$$

$$\dot{x}_9 = 0.25(u_{10} - 16x_9) \tag{7.1i}$$

$$\dot{x}_{10} = 0.2(u_{11} - 25x_{10}) \tag{7.1j}$$

$$\dot{x}_{11} = 0.17(u_{12} - 36x_{11}) \tag{7.1k}$$

$$\dot{x}_{12} = 0.14(u_{13} - 49x_{12}) \tag{7.1l}$$

$$\dot{x}_{13} = x_{14} + u_{15} \tag{7.1m}$$

$$\dot{x}_{14} = -x_{13} + (1 - {x_{13}}^2)x_{14} + u_{14} \tag{7.1n}$$

$$\dot{x}_{15} = -x_{15}{u_{16}}^2 \tag{7.1o}$$

Table 18. Parameters for the linearization comparison

	Baseline
Time step	2E-5 units
Control Effort Saturation	±[2 2 2 2 2 3 2 10 20 30 50 70 100 5 2 2]
Aggressiveness, $\dot{V}_{target}(0)$	$-2E4$
V_2 step sharpness, β	0.5
γ	1.0
Switching threshold, $\lvert D_1^2 + \cdots + D_m^2 \rvert_{threshold}$	0.01
Initial conditions	[0.3,0.4,0.5,0.6,0.7,0.8, 0.9,1.1,1.2,1.3,1.4,1.5, 1.6,1.7,1.8]
Setpoint	1 (for all states)

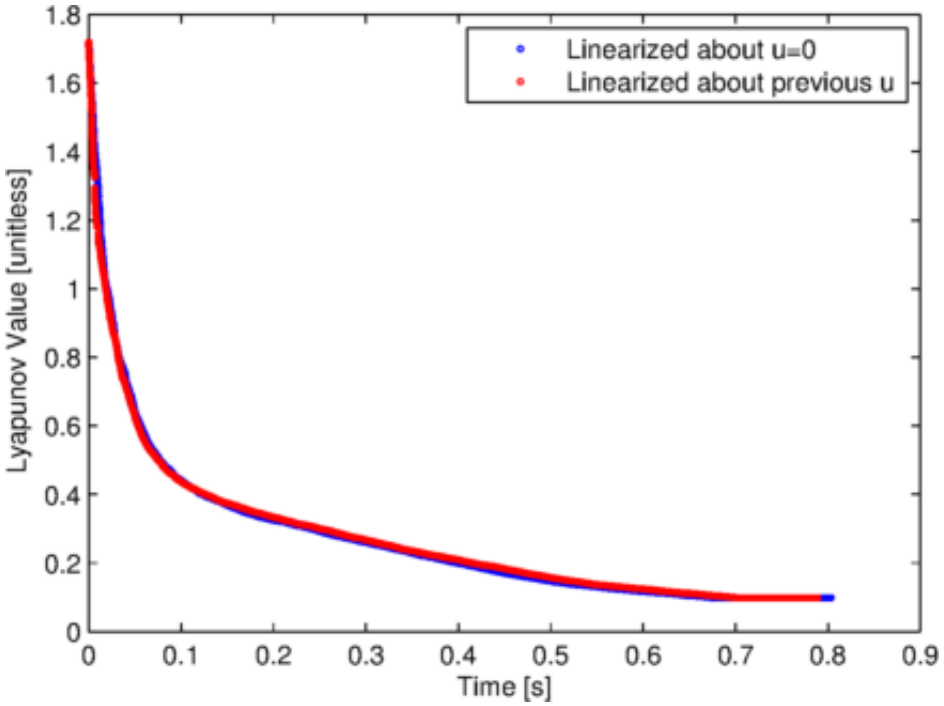


Figure 73. Comparison: Lyapunov value using two linearization techniques.

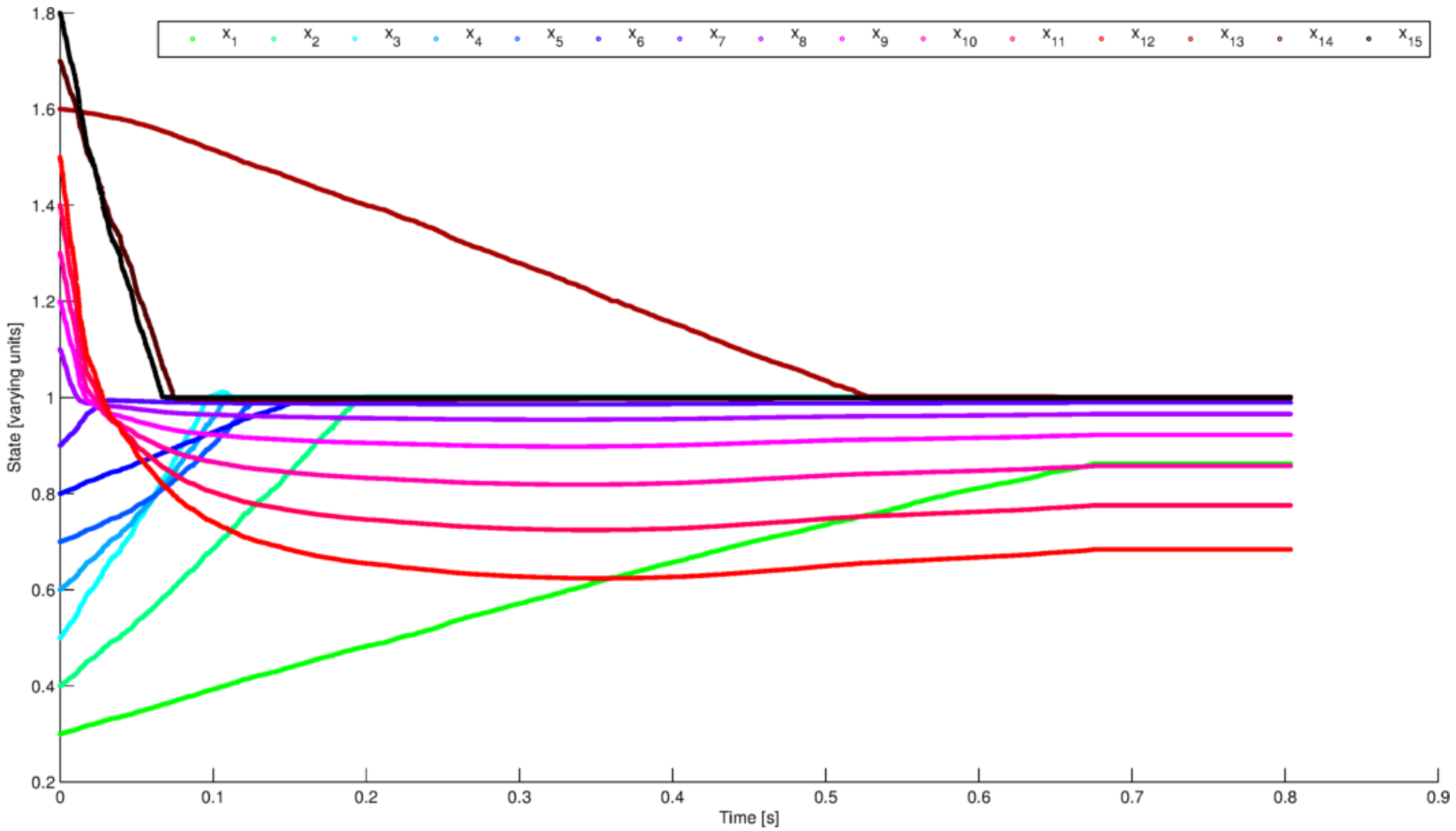


Figure 74. Trajectories of the states when linearized about u=0.

7.5 INTEGRAL ACTION

A nice feature of PID control is integral gain, which is often used to eliminate steady-state error. The same technique can be useful with the switched Lyapunov controller. Equation 4.2 described how the "aggressiveness" of the controller is tapered as it approaches its setpoint. Note that $V(0)$ is a measure of the error in the system across all states. Integral action is introduced by integrating error over time:

$$\dot{V}_{target}(t) = \dot{V}_{target}(0) * \left(\frac{V(t)}{V(0)}\right)^2 + K_I \int_0^t V(t)dt \tag{7.2}$$

Figure 75 shows how integral action can improve steady-state error. Figure 75 is based on the seven-motor simulation of Simulation Five (Chapter Four) with the parameters in Table 19. It compares a snapshot of the system performance towards the end of the simulation. When integral action was used, K_I was set at 100.

Table 19. Parameters of the Integral Action Case Study

Time step	1 ms
Control Effort Saturation	±10, 50, 100, 250, 500, 1000, 5000
Aggressiveness, $\dot{V}_{target}(0)$	-10 dB
V_2 step sharpness, β	0.5
γ	0.1
Switching threshold	0

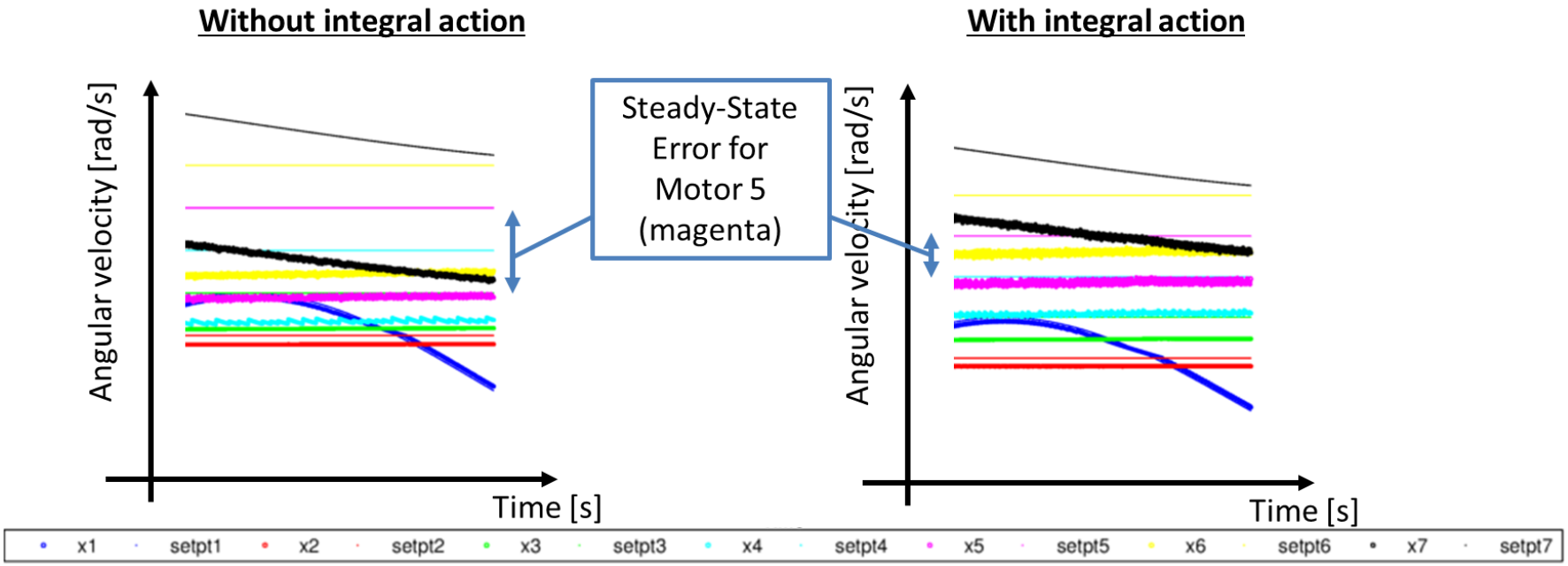


Figure 75. Integral action improved the tracking accuracy of the motor controller.

Figure 76 and Figure 77 show the benefit of integral action from the perspective of total error across all states. The error is significantly less, especially towards the end of the simulation. This resembles what one would expect from PID control; however, it is worth noting that integral action should not be applied blindly, as it can lead to integral windup and overshoot.

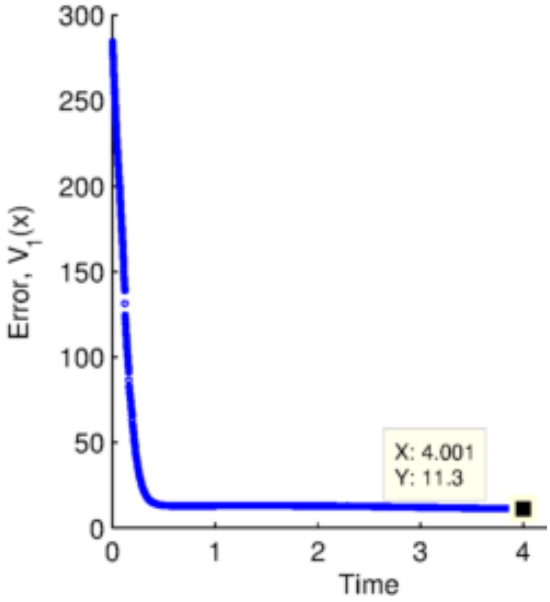


Figure 76. Error when integral action is not applied.

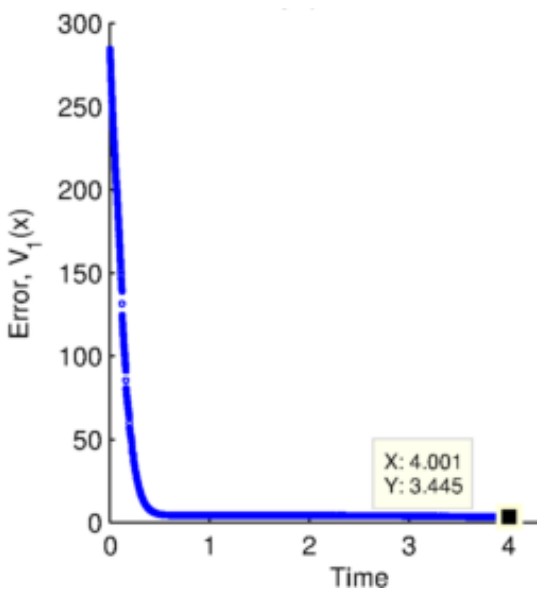


Figure 77. Less error when integral action is applied.

7.6 A STRATEGY FOR SYSTEMS WITH RELATIVE DEGREE GREATER THAN ONE

In Chapter 4 "Comparison with PID controllers", we described how the switched Lyapunov controller, as presented in Chapter 3, had difficulty controlling integrator systems. This section looks at a modified controller for single-input single-output systems with relative degree of two or more (including integrators).

It is common to put dynamic systems with a single control input and single output into "normal form." This is a canonical representation; it contains all available information

while decomposing the system into "internal" and "external" components [Khalil, 1996]. We assume here that the internal components (a.k.a. the zero dynamics, if they exist) are stable when the external components achieve equilibrium, i.e. the system is "minimum phase." We modify the switched Lyapunov controller based on this "normal form" because it is useful for underactuated systems, integrators, and other systems with relative degrees greater than zero.

For a single-input, single-output system of the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})u, \;\; \boldsymbol{x}\epsilon\boldsymbol{R}^n \tag{7.3}$$

$$y = h(\boldsymbol{x}) \tag{7.4}$$

The system is decomposed into external components ($\boldsymbol{\xi}$) and internal components ($\boldsymbol{z}$) as follows:

$$\begin{bmatrix} \boldsymbol{\xi} \\ \hline \boldsymbol{z} \end{bmatrix} = \begin{bmatrix} h(\boldsymbol{x}) \\ L_f(h(\boldsymbol{x})) \\ \vdots \\ L_f^{r-1}(h(\boldsymbol{x})) \\ \hline \boldsymbol{M}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \\ \vdots \\ y^{r-1} \\ \hline \boldsymbol{M}(\boldsymbol{x}) \end{bmatrix}, \boldsymbol{\xi} \in \mathbb{R}^{\boldsymbol{r}}, \boldsymbol{z} \in \mathbb{R}^{\boldsymbol{n-r}} \tag{7.5}$$

Where $L_{\emptyset}(\gamma(\boldsymbol{x}))$ is the directional *Lie derivative* of the scalar function $\gamma(\boldsymbol{x})$ in the direction of the vector field $\emptyset(\boldsymbol{x})$:

$$L_{\emptyset}(\gamma(\boldsymbol{x})) = \frac{\partial\gamma(x)}{\partial \boldsymbol{x}} \cdot \emptyset(\boldsymbol{x}) \tag{7.6a}$$

$$L_{\emptyset}{}^{k}(\gamma(\boldsymbol{x})) = \frac{\partial}{\partial \boldsymbol{x}}\left\{L_{\emptyset}{}^{k-1}(\gamma(\boldsymbol{x}))\right\} \cdot \emptyset(\boldsymbol{x}) \tag{7.6b}$$

$\boldsymbol{\xi}(\boldsymbol{x})$ is a nonlinear transformation of the state variables; it is derived by differentiating y with respect to time until the control effort u appears in $\xi_{\boldsymbol{r}}$ (the last element of $\boldsymbol{\xi}$). The control effort appears in the r^{th} derivative. The reduced-order manifold $\boldsymbol{M}(\boldsymbol{x})$ is present if r (the relative degree of the system) is less than n (the order of the system). The

calculation of $\boldsymbol{M}(\boldsymbol{x})$ is beyond the scope of this study; we assume the system is minimum phase so it can be neglected.

The time derivatives of ξ_1 through ξ_{r-1} are given in Equation 7.4. The time derivative of ξ_r is particularly important because that is where the control effort has an effect. It is:

$$\dot{\xi}_r = L_f^r(h) + L_g\left(L_f^{r-1}(h)\right)u \tag{7.7}$$

The default CLF V_1 will be a function of the transformed state variables, $\boldsymbol{\xi}(\boldsymbol{x})$:

$$V_1(\boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{\xi}\cdot\boldsymbol{\xi} \tag{7.8}$$

Expressing $\dot{V}_1$ as an explicit function of u:

$$\dot{V}_1(\boldsymbol{\xi}) = \xi_1\dot{\xi}_1 + \cdots + \xi_r\dot{\xi}_r = \xi_1 L_f(h) + \cdots + \xi_r\left[L_f^r(h) + L_g\left(L_f^{r-1}(h)\right)u\right] \tag{7.9}$$

Solving for a u that will produce the desired motion:

$$u = \frac{\dot{V}_1(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r}\xi_i L_f^i(h)}{\xi_r L_g\left(L_f^{r-1}(h)\right)} = \frac{\dot{V}_1(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r}\xi_i L_f^i(h)}{\frac{\partial\dot{V}_1}{\partial u}} = \frac{\dot{V}_1(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r-1}\xi_i\xi_{i+1} - \xi_r L_f^r(h)}{\frac{\partial\dot{V}_1}{\partial u}} \tag{7.10}$$

As with the similar technique that was proposed in Chapter 3, an issue arises if $\frac{\partial\dot{V}_1}{\partial u} \approx 0$, i.e. if the denominator is nearly singular. The alternative CLF to use in that case:

$$V_2(\boldsymbol{\xi}) = V_1\left[0.9 + 0.1\left|\xi_r - 1\right|\right] \tag{7.11}$$

$V_2(\boldsymbol{\xi})$ was carefully chosen to have $sign\left(\dot{V}_2\right) = sign(\dot{V}_1)$ while avoiding the singularity caused by $\xi_r = 0$ in $\frac{\partial\dot{V}_1}{\partial u}$ (see Equation 7.8). The second CLF is useful in canceling the singularity when $\xi_r \neq \xi_1$, i.e. for systems with a relative degree of two or more ($r > 1$). For a system with a relative order of two, $V_2(\boldsymbol{\xi})$ and its level curves are shown in Figure 78

and Figure 79, respectively. At each point, the controller uses the CLF with the denominator of greatest magnitude to calculate the stabilizing control effort.

The intermediate calculations related to the time derivative of $V_2(\boldsymbol{\xi})$ are:

$$\dot{V}_2(\boldsymbol{\xi}) = \frac{\partial V_2}{\partial \xi_1}\dot{\xi}_1 + \cdots + \frac{\partial V_2}{\partial \xi_r}\dot{\xi}_r \tag{7.12a}$$

$$\frac{\partial V_2}{\partial \xi_{i\neq r}} = \xi_i\left(0.9 + 0.1\left|\xi_r - 1\right|\right) \tag{7.12b}$$

$$\frac{\partial V_2}{\partial \xi_r} = \xi_r\left(0.9 + 0.1\left|\xi_r - 1\right|\right) + 0.1V_1 * sign(\xi_r - 1) \tag{7.12c}$$

Expressing $\dot{V}_2$ as an explicit function of u then solving for u:

$$u = \frac{\dot{V}_2(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r}\frac{\partial V_2}{\partial \xi_i}L_f^i(h)}{\frac{\partial V_2}{\partial \xi_r} * L_g\left(L_f^{r-1}(h)\right)} = \frac{\dot{V}_2(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r}\frac{\partial V_2}{\partial \xi_i}L_f^i(h)}{\left[\xi_r\left(0.9+0.1\left|\xi_r-1\right|\right)+0.1V_1 * sign(\xi_r-1)\right] * L_g\left(L_f^{r-1}(h)\right)} \tag{7.13}$$

The new $0.1V_1 * sign(\xi_r - 1)$ term in the denominator of Equation 7.12 avoids the singularity caused by $\xi_r = 0$.

Do $\boldsymbol{V_1(\xi)}$ and $\boldsymbol{V_2(\xi)}$ ensure stability?

As discussed in Chapter Three, the switched Lyapunov controller must ensure that:

- At every point along the system's directory excluding the origin, $\frac{\partial \dot{V}_1}{\partial u} \neq 0$ and/or $\frac{\partial \dot{V}_2}{\partial u} \neq 0$.
- If $\frac{\partial \dot{V}_1}{\partial u}$ is singular at a point, then $\dot{V}_2$ is regulated to be negative. Otherwise $\dot{V}_1$ is regulated to be negative.

Clearly V_1 meets the other requirements from Chapter Three: it is continuous, radially unbounded, and positive definite. The proof of $sign(\dot{V_2}) = sign(\dot{V_1})$ follows.

Proof of $\boldsymbol{sign(\dot{V_2}) = sign(\dot{V_1})}$

The partial derivative of V_1 with respect to any state variable $\xi_i, i \in \boldsymbol{R}^r$ is $\frac{\partial V_1}{\partial \xi_i} = \xi_i$, so clearly $sign\left(\frac{\partial V_1}{\partial \xi_i}\right) = sign(\xi_i)$. The partial derivative of V_2 with respect to any state variable $\xi_j,\ j \in \boldsymbol{R}^{r-1}$ is $\frac{\partial V_2}{\partial \xi_j} = \xi_j \left(0.9 + 0.1\left|\xi_j\right|\right)$. The quantity in parantheses is always positive, hence $sign\left(\frac{\partial V_2}{\partial \xi_j}\right) = sign\left(\xi_j\right)$. Finally,

$$\frac{\partial V_2}{\partial \xi_r} = \xi_r\left[0.9 + 0.1 * \left|\xi_r - 1\right|\right] + 0.1 V_1 * sign(\xi_r - 1) \qquad (7.14)$$

The left-hand term is larger and dominates the sign when $|\boldsymbol{\xi}|_2 < 4.24$. Its sign is determined by $sign(\xi_r)$. For values of $|\boldsymbol{\xi}|_2 \geq 4.24$, $sign(\xi_r) = sign(\xi_r - 1)$. Hence $sign\left(\frac{\partial V_2}{\partial \xi_r}\right) = sign(\xi_r)$ and $sign\left(\frac{\partial V_2}{\partial \boldsymbol{\xi}}\right) = sign\left(\frac{\partial V_1}{\partial \boldsymbol{\xi}}\right)$. Therefore a move in $\boldsymbol{\xi}$-space causes V_1 and V_2 to increase or decrease together.

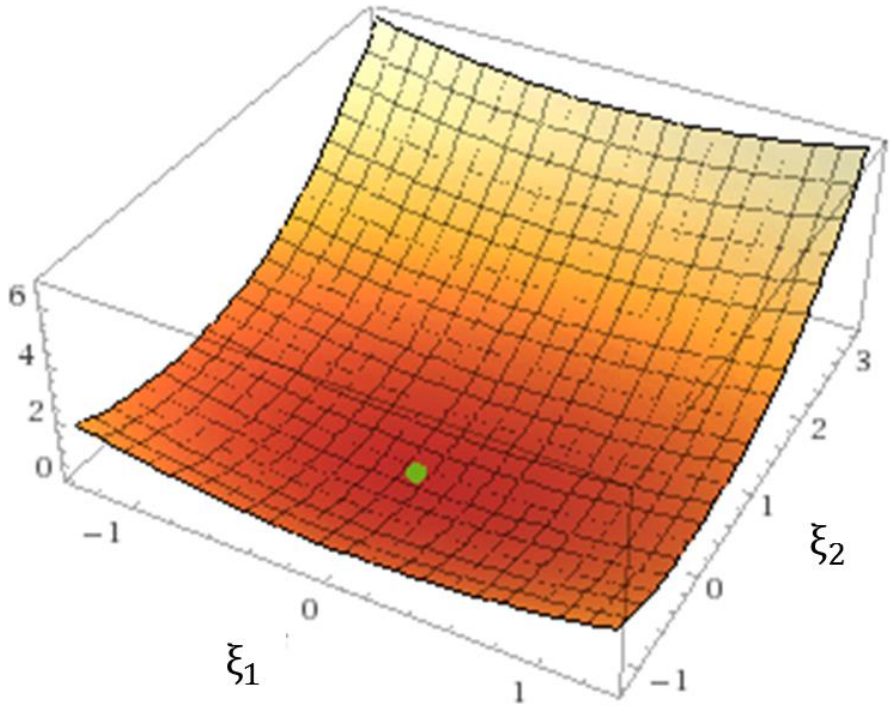


Figure 78. Plot of $V_2(\boldsymbol{\xi})$. The global minimum at (0,0) is marked with a green dot.

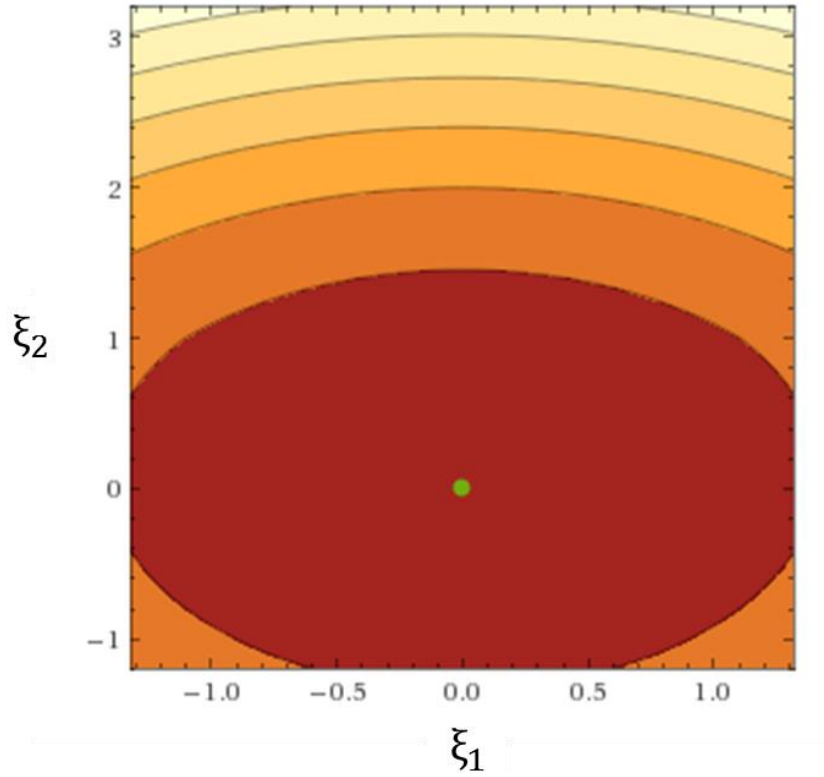


Figure 79. Level curves of $V_2(\boldsymbol{\xi})$. The global minimum at (0,0) is marked with a green dot.

Is $\frac{\partial \dot{V}_1}{\partial u} \neq 0$ and/or $\frac{\partial \dot{V}_2}{\partial u} \neq 0$ at every point (excluding the origin)? The expansion of $\frac{\partial \dot{V}_1}{\partial u} = \frac{\partial \dot{V}_2}{\partial u} = 0$ is:

$$\xi_r L_g\left(L_f^{r-1}(h)\right) = [\xi_r(0.9 + 0.1|\xi_r - 1|) + 0.1V_1 * sign(\xi_r - 1)] * L_g\left(L_f^{r-1}(h)\right) = 0 \tag{7.15}$$

A solution to Equation 7.15 is:

$$L_g\left(L_f^{r-1}(h)\right) = 0 \tag{7.16}$$

The other possible solution for $\xi_r L_g\left(L_f^{r-1}(h)\right) = 0$ (the left-hand side) is $\xi_r = 0$; however, assuming $r \neq 1$, the middle equality is not satisfied since, for nonzero V_1,

$0.1V_1 * sign((0) - 1) \neq 0$. Hence either $\frac{\partial \dot{V}_1}{\partial u}$ or $\frac{\partial \dot{V}_2}{\partial u}$ is reducible unless $V_1 = 0$ (the system is at the setpoint) or $L_g\left(L_f^{r-1}(h)\right) = 0$. The latter makes sense at an intuitive level since the control effort has no effect if $L_g\left(L_f^{r-1}(h)\right) = 0$ (refer to Equation 7.7).

Drawbacks and advantages of the controller in normal form

Comparing $V_2(\boldsymbol{\xi})$ (Equation 7.11) for systems in normal form to $V_2(\mathbf{x})$ (Equation 3.35) for systems in state-space form, it is clear that each form has its own merits. $V_2(\boldsymbol{\xi})$ has simpler partial derivatives and it does not vary in a pointwise fashion, so the implementation of a controller based on $V_2(\boldsymbol{\xi})$ is simpler. However, the surface described by $V_2(\boldsymbol{\xi})$ is just approximately radially symmetric, whereas $V_2(\mathbf{x})$ has obvious radial symmetry. This makes the proof of ($\frac{\partial \dot{V}_1}{\partial u} \neq 0$ and/or $\frac{\partial \dot{V}_2}{\partial u} \neq 0$) simpler and more elegant for $V_2(\mathbf{x})$. The other, most significant difference is that $V_2(\mathbf{x})$ is applicable to a much broader range of systems, including multiple-input systems.

Simulation

This example comes from Khalil [1996, page 542]. The system is:

$$\dot{x_1} = -x_1 + \frac{2+x_3^2}{1+x_3^2}u \tag{7.17a}$$

$$\dot{x_2} = x_3 \tag{7.17b}$$

$$\dot{x_3} = x_1 x_3 + u \tag{7.17c}$$

$$y = x_2 \tag{7.17d}$$

It is a 3rd-order, minimum phase system with relative degree of two. Note how *u* appears in two places; this system could not be controlled with PID, in general. The accessibility distribution *C* is calculated as follows [Hedrick, 2005]:

$$f = \begin{bmatrix} -x_1 \\ x_3 \\ x_1 x_3 \end{bmatrix} \tag{7.18a}$$

$$g = \begin{bmatrix} \frac{2+x_3^2}{1+x_3^2} \\ 0 \\ 1 \end{bmatrix} \tag{7.18b}$$

$$[f,g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g = \begin{bmatrix} 0 & 0 & \frac{-2x_3}{(x_3^2+1)^2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -x_1 \\ x_3 \\ x_1 x_3 \end{bmatrix} - \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ x_3 & 0 & x_1 \end{bmatrix} \begin{bmatrix} \frac{2+x_3^2}{1+x_3^2} \\ 0 \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{-2x_1 x_3^2}{(x_3^2+1)^2} + \frac{2+x_3^2}{1+x_3^2} \\ -1 \\ -x_1 - \frac{2x_3+x_3^3}{1+x_3^2} \end{bmatrix} \tag{7.18c}$$

$$f,[f,g] = \frac{\partial [f,g]}{\partial x} f - \frac{\partial f}{\partial x} [f,g]$$

$$= \begin{bmatrix} \frac{2+x_3^2}{1+x_3^2} + \frac{2x_1 x_3^3 - 2x_1 x_3^2}{(x_3^2+1)^2} + \frac{2{x_1}^2 x_3^3 (x_3^2-3)}{(x_3^2+1)^3} \\ \frac{2+x_3^2}{1+x_3^2} + 1 \\ x_1 + x_1 \left[\frac{2+x_3^2}{1+x_3^2} + 1 \right] - x_3 \left[\frac{2+x_3^2}{1+x_3^2} - \frac{2x_1 x_3^2}{(x_3^2+1)^2} \right] + \frac{(x_1 x_3 (-x_3^4 + x_3^2 + 2))}{(x_3^2+1)^2} \end{bmatrix}$$

(7.18d)

$$C = [g \quad [f,g] \quad f,[f,g]] = \begin{bmatrix} \frac{2+x_3^2}{1+x_3^2} & \frac{-2x_1x_3^2}{(x_3^2+1)^2} + \frac{2+x_3^2}{1+x_3^2} & \\ 0 & -1 & f,[f,g] \\ 1 & -x_1 - \frac{2x_3+x_3^3}{1+x_3^2} & \end{bmatrix} \tag{7.18e}$$

C has full rank over the vast majority of state-space, so the system is largely accessible.

The simulation settings are given in Figure 80.

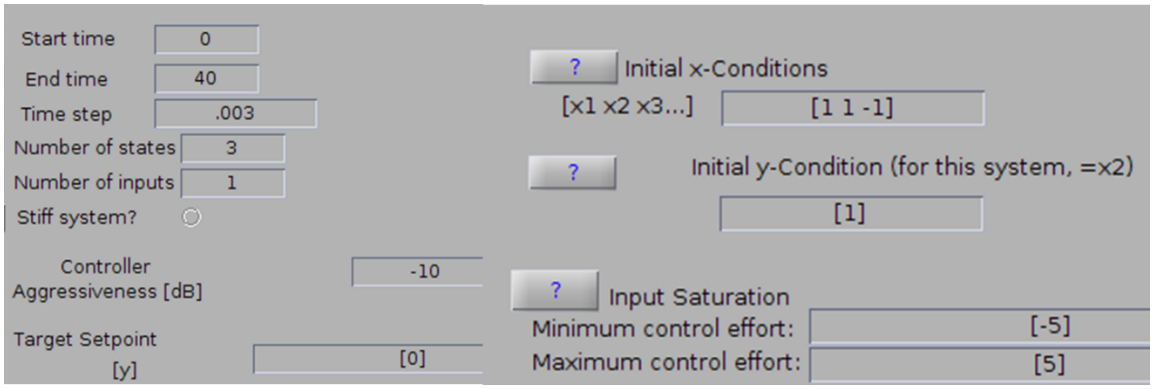


Figure 80. Settings for the simulation of a 3rd-order system in normal form.

The open-loop system is clearly unstable (Figure 81). Under closed-loop control, state x_2 (the state of interest) is driven towards the setpoint of zero (Figure 82).

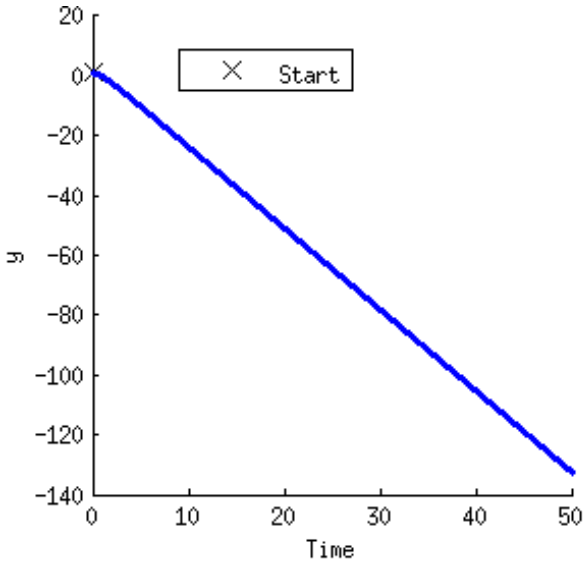


Figure 81. Open loop instability.

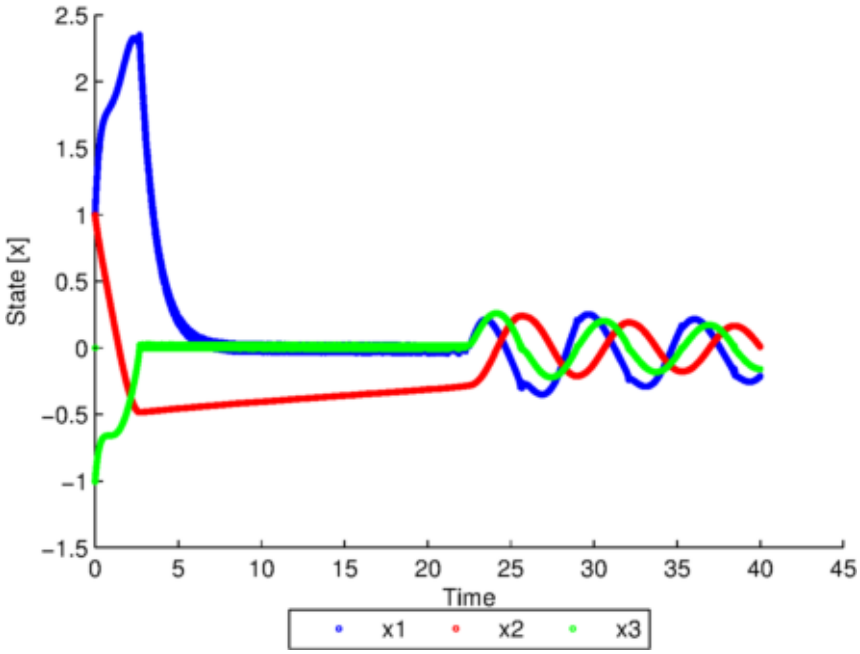


Figure 82. Closed loop stability.

Figure 83 shows the error drop in approximately exponential fashion, as it was designed to do. Figure 84 logs the switching between the CLF's. V_1 happens to dominate, initially, but the CLF switches rapidly throughout the rest of the simulation. Figure 85 shows the control effort over the course of the simulation. Initially, there is significant chatter but it subsides as the system approaches its setpoint.

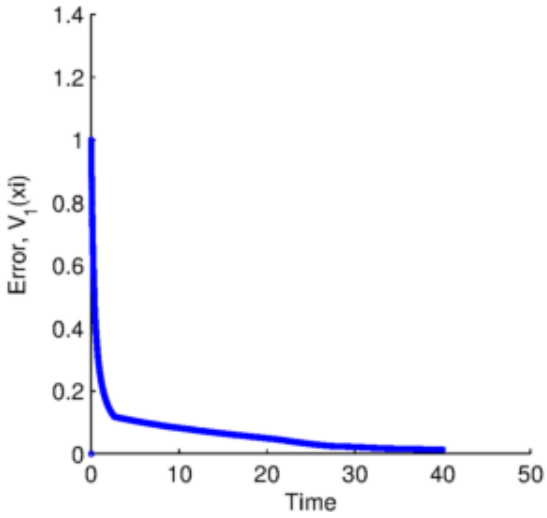


Figure 83. The Lyapunov value drops approximately exponentially.

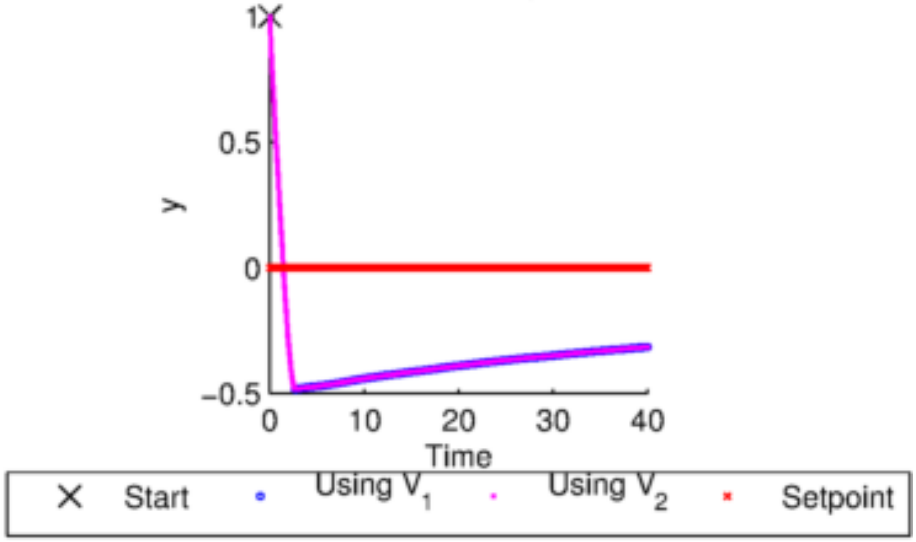


Figure 84. A log of the CLF switches. V_2 is active almost exclusively over the first 5% of the simulation, then the switches occur frequently.

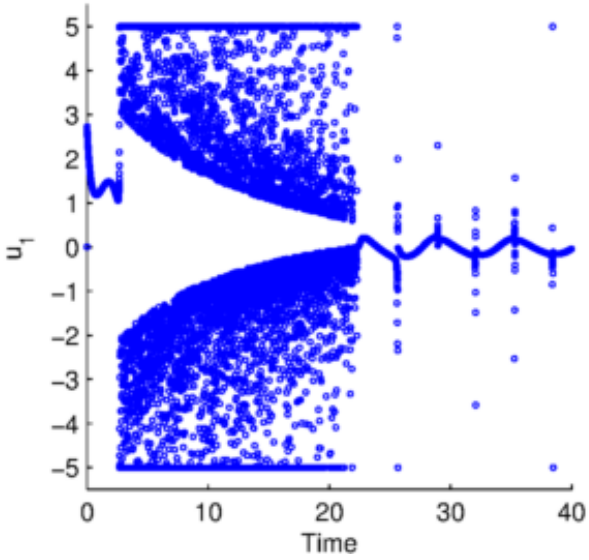


Figure 85. Control effort during simulation of the coupled system in normal form.

Finally, Figure 86 and Figure 87 plot the denominators for the control efforts as calculated with V_1 and V_2 over the course of the simulation. These figures verify that the algorithm is often useful for avoiding singularities.

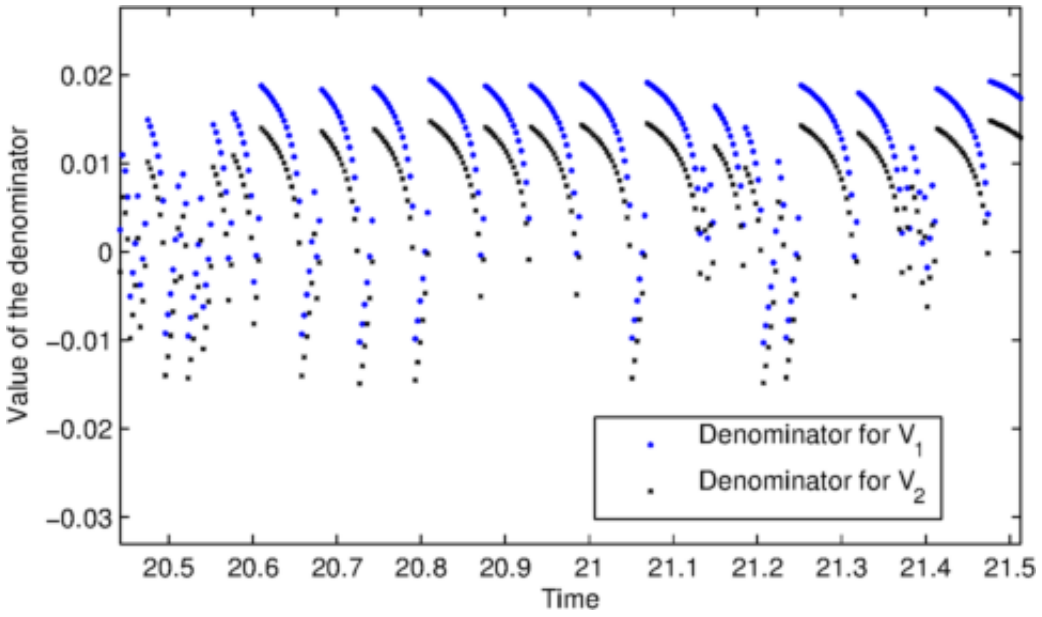


Figure 86. The difference in the magnitude of the denominators was small but it is significant when the error is small. This confirms that the algorithm is useful for avoiding singularities.

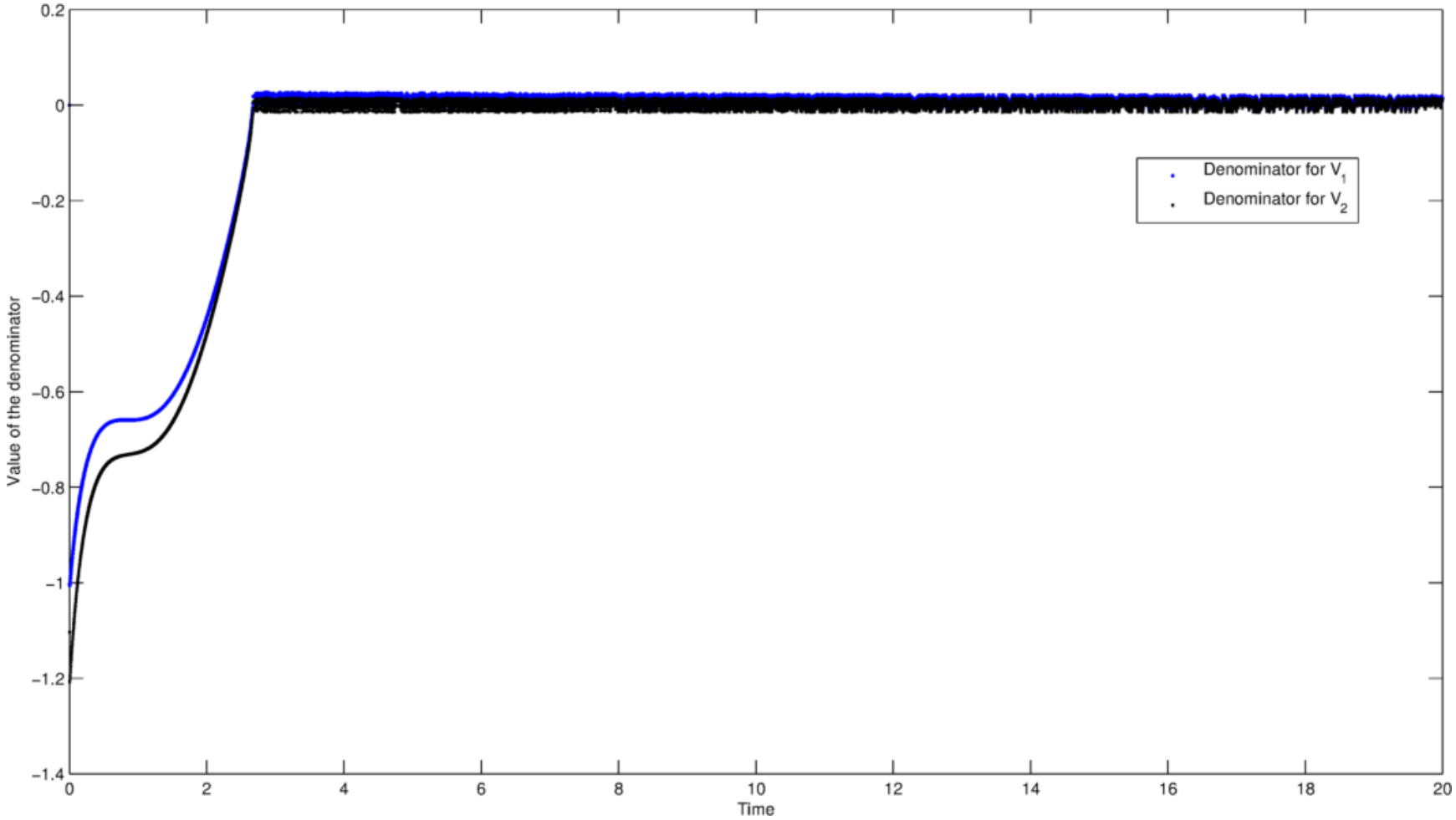


Figure 87. Values of the denominators for V_1 and V_2 over the entire simulation.

7.7 CONSIDERATIONS OF ROBUSTNESS

Matching Condition

Khalil [1996, page 578] describes a technique that can be used to ensure the stability of a Lyapunov controller for uncertainties that satisfy the "matching condition." The uncertainties are corrected by adding "additional control" to the nominally calculated stabilizing control effort. The uncertainties can be arbitrarily large as long as a bound on their size is known. Such uncertainties might arise from simplified models, parameter uncertainty, or computational error, and they are named as such because they enter the plant at the same point as the control effort:

$$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}) + G(t, \boldsymbol{x})[\boldsymbol{u} + \partial(t, \boldsymbol{x}, \boldsymbol{u})] \tag{7.19}$$

Where ∂ is the lumped expression of uncertainties. Khalil notes that, following some algebra, ∂ can account for uncertainties in f and G. If the control effort is decomposed into a nominally stabilizing control effort $\boldsymbol{\varphi}$ and an additional control $\boldsymbol{\gamma}$:

$$\boldsymbol{u} = \boldsymbol{\varphi}(t, \boldsymbol{x}) + \boldsymbol{\gamma}(t, \boldsymbol{x}) \tag{7.20}$$

Khalil lists two possible forms of $\boldsymbol{\gamma}$ that ensure stability despite arbitrarily large uncertainties.

Nonminimum Phase Systems

Minimum phase systems are defined as those having zeros in the left half-plane only [Khalil, 1996, page 539]. A nonlinear system is minimum phase if it has zero dynamics and they are asymptotically stable [Khalil, 1996, page 541]. In layman's terms, this means that the internal states of a system are stable. Although the internal states (i.e. the "zero dynamics") have no bearing on the output of a system, it is often important that

they be stable. For example, a redundant kinematic manipulator is a nonminimum phase system; operators of these robots know to be particularly careful when the robot is close to a singular configuration, i.e. when the controller commands large, ill-conditioned joint velocity solutions.

Nonminimum phase systems pose an insurmountable challenge to all controllers, and the present controller is no exception. It is impossible to achieve perfect or asymptotic tracking for such systems [Slotine, 1991, pg. 195] (e.g. linear systems with zeros in the right-half plane) because cancelling the plant dynamics at the frequencies of the zeros requires infinite control effort. "For nonminimum phase systems, perfect tracking of arbitrary trajectories requires infinite control effort."

However, at frequencies much lower than the frequencies of the zeros, tracking performance can still be quite good. Readers are reminded that perfect *tracking* performance (as stated in Chapter 3) would require proof of asymptotic tracking by applying Barbalat's Lemma for each individual system.

Linear Systems: Unstable Poles

For controllers like the present which are based on Lyapunov's direct method, the open-loop poles of the system have no bearing on the stability of the system. The controller will compensate for unstable poles as long as a valid CLF is known. For systems in normal form, the existence of a valid CLF depends only on a nonzero $L_g\left(L_f^{r-1}(h)\right)$.

Time Delays

In the 1990's, study of the stability of time delay systems became an active topic of research [Wu, 2010]. This topic is very practical, in part, because time delays are unavoidable in networked digital systems and (by extension) to many control systems [Tipsuwan, 2003]. Most of that research has been focused on linear systems; stability analysis of nonlinear time-delay systems seems to be a nascent field which is focused on Lur'e systems and yields conservative, sufficient results [Wu, 2010]. A Lur'e system is a linear system with the addition of a single, scalar nonlinear term.

For linear systems, i.e. systems of the form:

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + A_d\boldsymbol{x}(t-h) \tag{7.21a}$$

$$\boldsymbol{x}(t) = \boldsymbol{\varphi}(t),\ t\epsilon[-h, 0]\ \ (Initial\ conditions) \tag{7.21b}$$

A well-known method to ensure stability is based on the "Lyapunov-Krasovskii functional." A basic, quadratic Lyapunov function is assumed then a second term is added to account for the time delay:

$$V_1(\boldsymbol{x_t}) = \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}(t) + \int_{t-h}^{t}\boldsymbol{x}^T(s)\boldsymbol{Q}\boldsymbol{x}(s)ds \tag{7.22}$$

Where $\boldsymbol{x_t}$ denotes a "translation operator." The integral term accounts for the drift of the system during the delay. If the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ can be solved for such that $\dot{V}_1 < 0$, then stability is ensured despite arbitrarily large delays! The main reason (according to Wu [2010]) why the study of this stability topic became more popular in the 1990's was the appearance of computational tools (e.g. MATLAB) to solve for $\boldsymbol{P}$ and $\boldsymbol{Q}$. One issue with Equation 7.22 is that, since it assures stability for time delays of any magnitude, it is very conservative. A less conservative stability proof can be formulated based on the maximum

size of the time delay h, but it requires the addition of a double integral term to Equation 7.22 and the techniques for solving that expanded equation are complex.

It is well-known that the assumption of a basic Lyapunov function often fails, so that is a major drawback to the Lyapunov-Krasovskii approach. However, it seems that the incorporation of a second, complementary Lyapunov-Krasovskii [LK] functional into the standard LK method would broaden that technique's applicability, presenting an alternative to Equation 7.22. If the second LK functional were collectively exhaustive with the first, as the presented controllers are for non-time-delay systems, that would be a significant contribution to the field.

The switched CLF's we have presented do not include an extra integral term to account for time delay, so our controllers as presented in Chapter Three and Section 7.6 do not ensure stability if the system includes time delays.

Excessive Control Effort

When designing linear controllers, a designer must be cautious because excessively large gains can shift the location of the poles into the right-half plane and destabilize a system. For our "normal form" controller based on Lyapunov's direct method (Section 7.6), the analogous concern is to ensure $\dot{V}_1 < 0$. Fortunately, the relationship between u and $\frac{\partial \dot{V}_1}{\partial u}$ is linear (see Equation 7.10). For analog implementations where $\frac{\partial \dot{V}_1}{\partial u}$ is updated constantly and there are no time delays, $\dot{V}_1$ will be negative if u has the correct sign. Thus, there is no risk from excessively large $\dot{V}_{target}$ or control efforts.

For digital implementations, there are two practical considerations that could destabilize the system:

1) If $sign\left(\frac{\partial \dot{V}_1}{\partial u}\right)$ changes within a time step, e.g. if the system's behavior changes abruptly or the time step is excessively long, then u suddenly has the wrong sign and $\dot{V}_1$ will increase; hence, the system could become unstable.
2) If the time step is excessively long or a very large u is applied, the system could overshoot the setpoint and u would have the wrong sign. Again, the system could become unstable.

Both considerations are resolved with small time steps. Consideration (2) can occur only when the system is already close to the setpoint, so it is of minor concern.

Shock upon switching CLF's

A potential cause for concern with any switched control algorithm (e.g. gain scheduling, sliding mode control) is shock to the system when the switch occurs. A simple but non-ideal solution to this problem is a low-pass filter on u, a feature which is already present in the developed MATLAB software package. It was described in Chapter 4 "Simulation 1: First-order RC circuit". We also note that, according to the simulation results in Figure 87, the difference between u as calculated with $\dot{V}_{1,target}$ versus u as calculated with $\dot{V}_{2,target}$ is very small unless the system is very close to its setpoint.

Slotine and Li [1990, pg. 291] approached the smoothing problem from a slightly different perspective. For a sliding mode controller, they began smoothing the switching

discontinuities when the system is within a narrow "boundary layer" of the target trajectory. This "boundary layer method" has advantages over the global application of a low-pass filter:

- The filter is not active unless it is needed.
- The width of the boundary layer can be adjusted as needed because there is a "trade-off between tracking precision and robustness to unmodeled dynamics."
- The width of the boundary layer can be adjusted as needed based on the controller's available bandwidth. For example, the boundary layer can be widened, thereby increasing the degree of smoothing, to avoid exciting any unmodeled high-frequency dynamics of the system.

We propose a different method for smoothing the switched Lyapunov controller which is based on inspection of Equation 7.9 (reproduced below for convenience).

$$u = \frac{\dot{V}_1(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r} \xi_i L_f^i(h)}{\xi_r L_g\left(L_f^{r-1}(h)\right)} = \frac{\dot{V}_1(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r} \xi_i L_f^i(h)}{\frac{\partial \dot{V}_1}{\partial u}} = \frac{\dot{V}_1(\boldsymbol{\xi})_{target} - \sum_{i=1}^{r-1} \xi_i \xi_{i+1} - \xi_r L_f^r(h)}{\frac{\partial \dot{V}_1}{\partial u}} \quad (7.9)$$

The smoothing method applies to the switched Lyapunov controller for systems in normal form, which implies that the system is singe-input. If a very negative $\dot{V}_{target}$ is specified, then the sign of the numerator is constant and $sign(\frac{\partial \dot{V}_1}{\partial u})$ determines the required sign on u. Hence the crucial requirement for stability is that u maintains the required sign, as determined by $sign(\frac{\partial \dot{V}_1}{\partial u})$. Mathematically, this simplified requirement is:

$$sign(u) = -sign(\frac{\partial \dot{V}_1}{\partial u}) \quad (7.23)$$

When the sign of the unfiltered u-signal has not changed recently, filtering does not affect the sign of u. Thus the low-pass filter is applied unless the calculated, unfiltered u-signal changes sign from the previous iteration. If the sign does change, the filter is not applied and the u command is allowed to change instantaneously. In summary, a low-pass filter is applied on u unless:

$$sign(\frac{\partial \dot{V}_1}{\partial u})_{iteration\ k} \neq sign(\frac{\partial \dot{V}_1}{\partial u})_{iteration\ k-1} \tag{7.24}$$

(Since $sign\left(\frac{\partial \dot{V}_1}{\partial u}\right) = sign\left(\frac{\partial \dot{V}_2}{\partial u}\right)$, Equation 7.22 could be based on V_2 equivalently.)

In the case of a sign change, u is unfiltered for the next iteration because a delay in the response would be destabilizing. We call this technique "selective filtering." For perfectly-modeled systems, this ensures perfect tracking while allowing for an arbitrary degree of filtering on the control input. The simulation of Section 7.6 was redone with this filtering method and the results are given in Table 20-Table 22. The low-pass filter was a second-order Butterworth filter with a cutoff frequency of 50 $\frac{cycles}{time\ unit}$. The selectively filtered simulation had a much lower degree of chatter while its performance was similar in other aspects. In fact, the total error across all states was considerably less at the end of the simulation when selective filtering was employed.

Table 20. The effect of selective filtering on the trajectories of the states. The performance was similar except there is slightly more chatter when the error is small and selective filtering is employed.

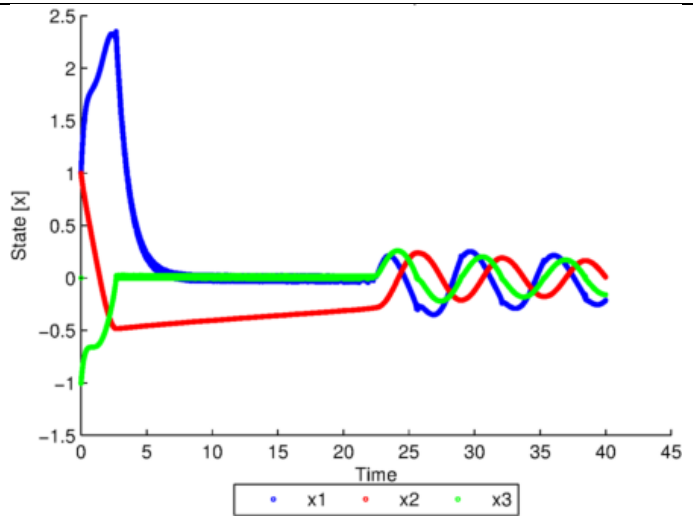
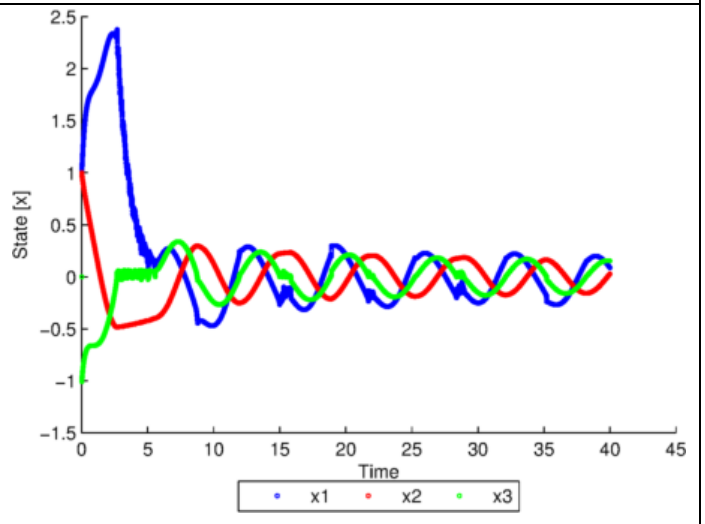
No Low-Pass Filter	Selective Low-Pass Filtering
	

Table 21. The effect of selective filtering on error. Note that the error at the end of the simulation is slightly less when selective filtering is applied.

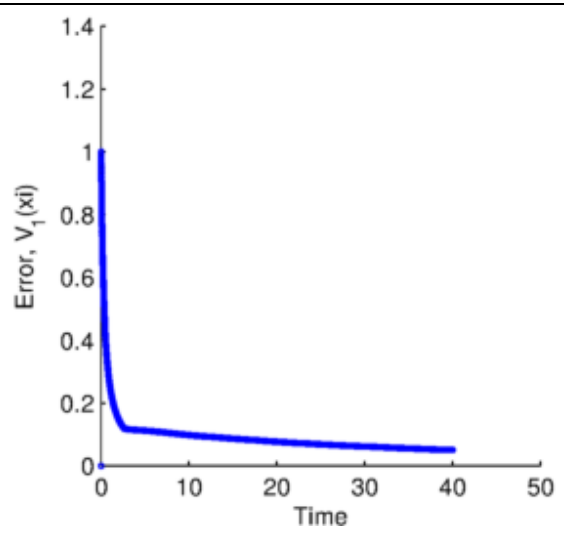
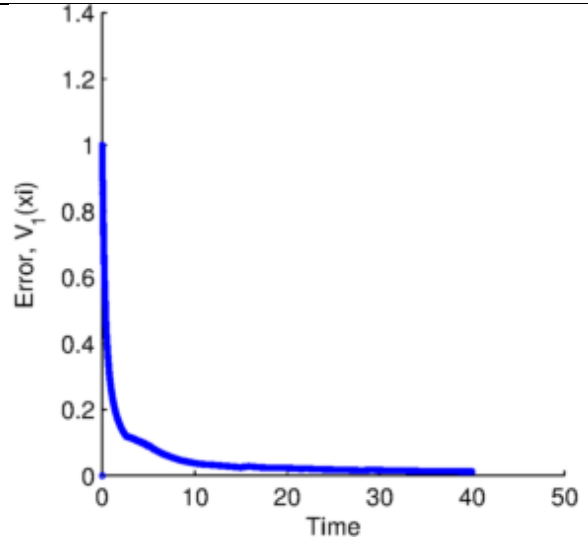
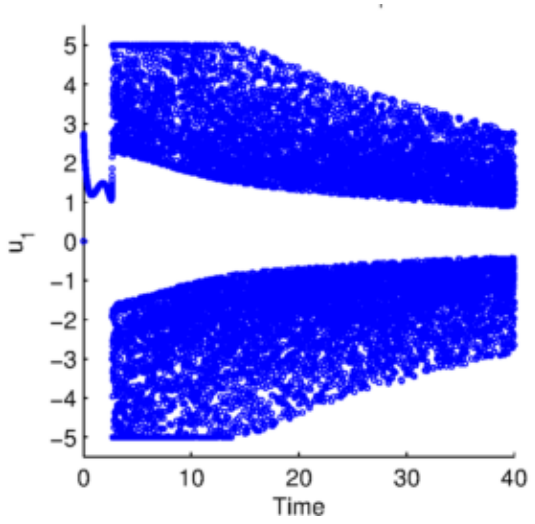
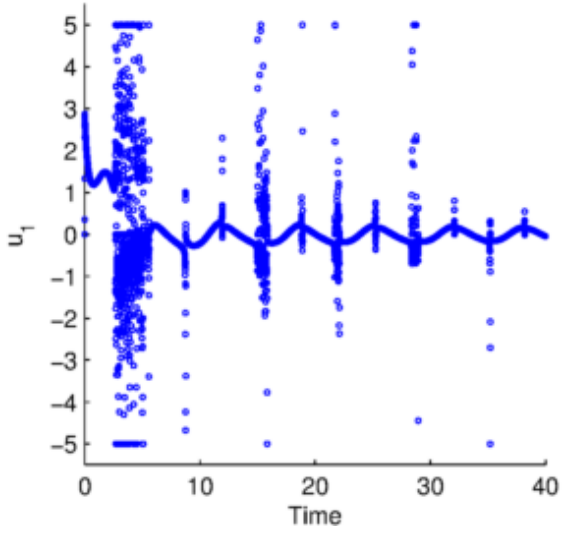
No Low-Pass Filter	Selective Low-Pass Filtering
	

Table 22. The effect of selective filtering on control effort. The selective filter greatly decreased chatter. The unfiltered spikes which remain are necessary to ensure stability.

No Low-Pass Filter	Selective Low-Pass Filtering
	

Unmodeled High-Frequency Dynamics (e.g. Noise)

Even the best models of physical systems cannot capture every aspect of a system's behavior. For example, quantization and sampling effects are two aspects of digitally implemented systems that usually go unmodeled. Slotine and Li [1990, pg. 416] note that such unmodeled dynamics are "of high frequency" if the model is "adequate." Furthermore, many sources of noise are zero-mean white noise which has no correlation with the state of the system and will not affect the stability of a passive system.

The classical approach to deal with unknown high-frequency dynamics is to limit the control effort to a bandwidth which is less than the frequency range of the unmodeled dynamics. Slotine and Li call that upper limit the "classical bandwidth limitation." In the special case where the classical bandwidth limit is due to a passive element, the controller for the passive element may be tuned separately and the bandwidth limitation on the rest of the system can be relaxed. This can lead to compartmentalized controllers which act at multiple time scales.

A practical approach to limit the effect of umodeled dynamics is to apply a low-pass filter on the control signal. As we discussed in Chapter 4 "Simulation 1: First-order RC circuit," a low-pass filter has already been implemented in the developed MATLAB software package. It has the additional benefits of reducing chatter and shock to the system.

Adding Robustness with Sliding Mode Control

The switched Lyapunov controller for systems in normal form (Section 7.6) can be supplemented with a sliding mode controller in parallel to add robustness at the expense of chatter. The new control effort is a sum of the contributions from each controller:

$$u = u_{lyap} + u_s \tag{7.25}$$

This superposition is stabilizing for the linear system because $V_1(\xi)$ is a common Lyapunov function for the sliding mode controller and the Lyapunov controller. A proof of that statement follows in the next subsection.

u_s acts in the phase plane of $(\xi_1, \dot{\xi}_1)$, pushing the system towards a "sliding surface." When the system reaches the sliding surface, the nominal, open-loop dynamics of the plant carry the system towards the origin [Slotine and Li, 1991]. The sliding surface is defined as:

$$\sigma \equiv \alpha_0 \xi_1 + \alpha_1 \dot{\xi}_1 = \alpha_0 \xi_1 + \alpha_1 \xi_2 \tag{7.26}$$

In this case, α_0 and α_1 adjust the slope of a linear sliding surface. The control effort which pushes the system onto this surface is calculated by:

$$u_s = -\eta * sign(\sigma) \tag{7.27}$$

η is a tunable parameter that can be scaled to account for disturbances of arbitrary size. However, it is desirable to minimize the size of η as it increases the magnitude of chatter from the sliding mode controller – and that is why it makes sense to combine the sliding mode controller with another type of controller which stabilizes the nominal dynamics of the system. The simulation in this section used these parameters:

$$\eta = 1.1 \text{ (sliding mode "gain")} \tag{7.28a}$$

$$\mathrm{d} = 1 \text{ (disturbance)} \tag{7.28b}$$

$$\alpha_0 = \alpha_1 = 1 \text{ (sliding surface slope)} \tag{7.28c}$$

$$\dot{V}_{target} = -10 \text{ (Lyapunov "gain")} \tag{7.28d}$$

A constant disturbance d was added to the control signal, u, so that $u = u_{lyap} + u_s + d$. Note how η was chosen to be just slightly larger than the disturbance. The control effort from the switched Lyapunov controller (u_{lyap}) was unfiltered and saturation limits were not applied.

Without the sliding mode controller, the disturbance destabilized the system. Likewise, the sliding mode controller's gain was insufficient for stability without the switched Lyapunov controller. However, when both controllers were combined in a hybrid scheme, the performance was very good.

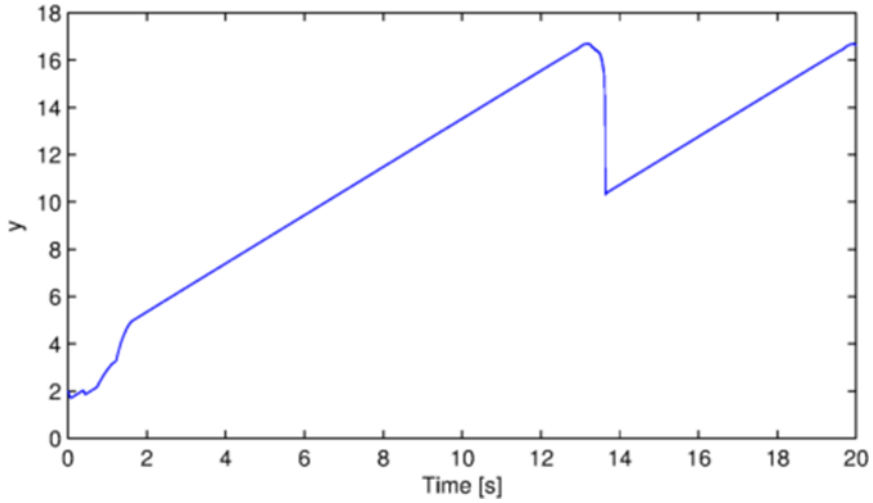


Figure 88. Without sliding mode control, the disturbance destabilized the switched Lyapunov controller.

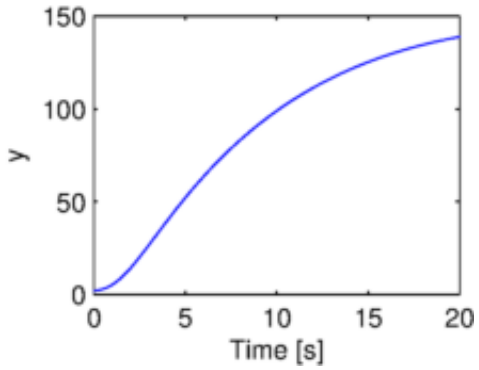


Figure 89. Without the switched Lyapunov controller, the gain of the sliding mode controller is not sufficient for stability.

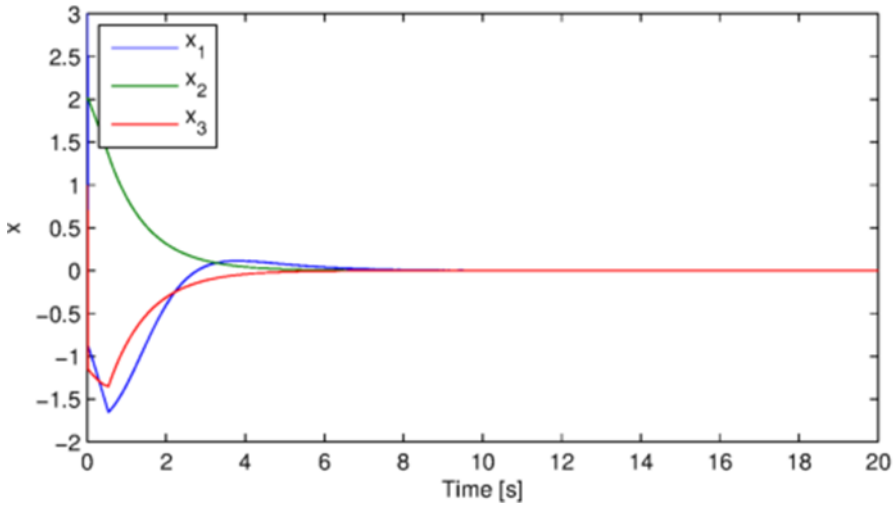


Figure 90. The hybrid Lyapunov-sliding controller performs very well.

Proof of Stability for Two Individually-Stabilizing Controllers in Parallel for Systems in Normal Form

In general, there is no guarantee that the combination of two individually-stabilizing controllers in parallel is stable, but the previous section about a hybrid Lyapunov/sliding mode controller calls for such a guarantee. Such a parallel arrangement is shown in Figure 91.

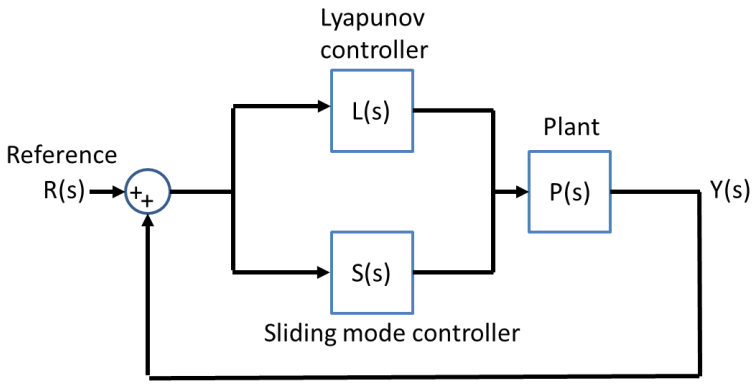


Figure 91. Parallel arrangement of controllers.

Passivity is one common way to analyze such combinations of controllers, but we provide a detailed graphical proof for SISO systems in [Zelenak, 2016]. Here we sketch the main concepts of the proof using $V_1 = \frac{1}{2}(\xi_1^{\,2} + \xi_2^{\,2})$ (a common Lyapunov function for the Lyapunov controller and the sliding mode controller) and for second-order systems (although the concept is extensible to higher orders and other Lyapunov functions).

Recall that this applies to systems in normal form and the dynamics are described by:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})u, \;\; \boldsymbol{x}\epsilon\boldsymbol{R}^n \tag{7.3}$$

$$y = h(\boldsymbol{x}) \tag{7.4}$$

The level curves of $V_1(\xi)$ are circles in the (ξ_1, ξ_2) plane. A level curve is shown in Figure 92, along with an arbitrary vector $\boldsymbol{f}(\boldsymbol{x})$, which is the open-loop "drift" dynamics of the system. This vector $\boldsymbol{f}$ is potentially destabilizing (i.e. it will cause the value of the Lyapunov function to increase), so a controller needs to add another vector $(\boldsymbol{g}(\boldsymbol{x})u)$ to return the system to stability.

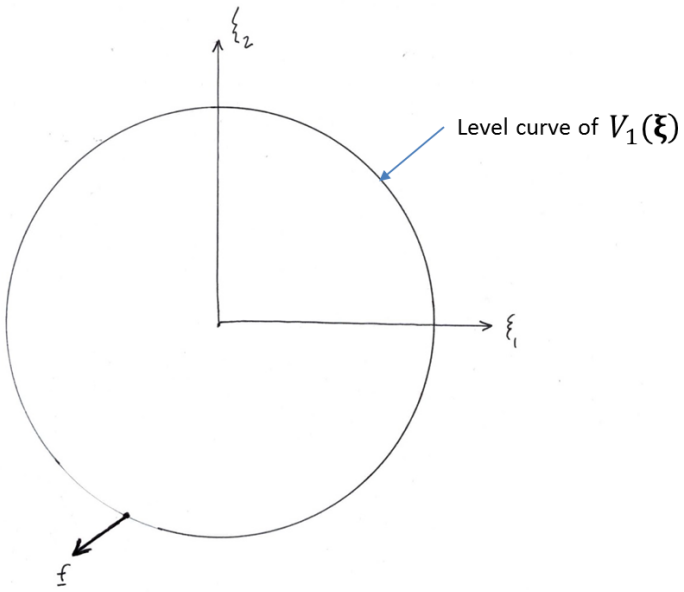


Figure 92. Level curve of V_1 and a potentially destabilizing "drift vector" for a nonlinear system in normal form.

The range of marginally stabilizing control vectors from the first controller is shown in Figure 93 in blue. The second controller can individually stabilize the system with the same range of vectors. Because the system is affine with respect to *u*, the control vectors sum when the controllers are placed in parallel. The possible vectors from the second controller are shown in green in Figure 93. An “edge case” is illustrated.

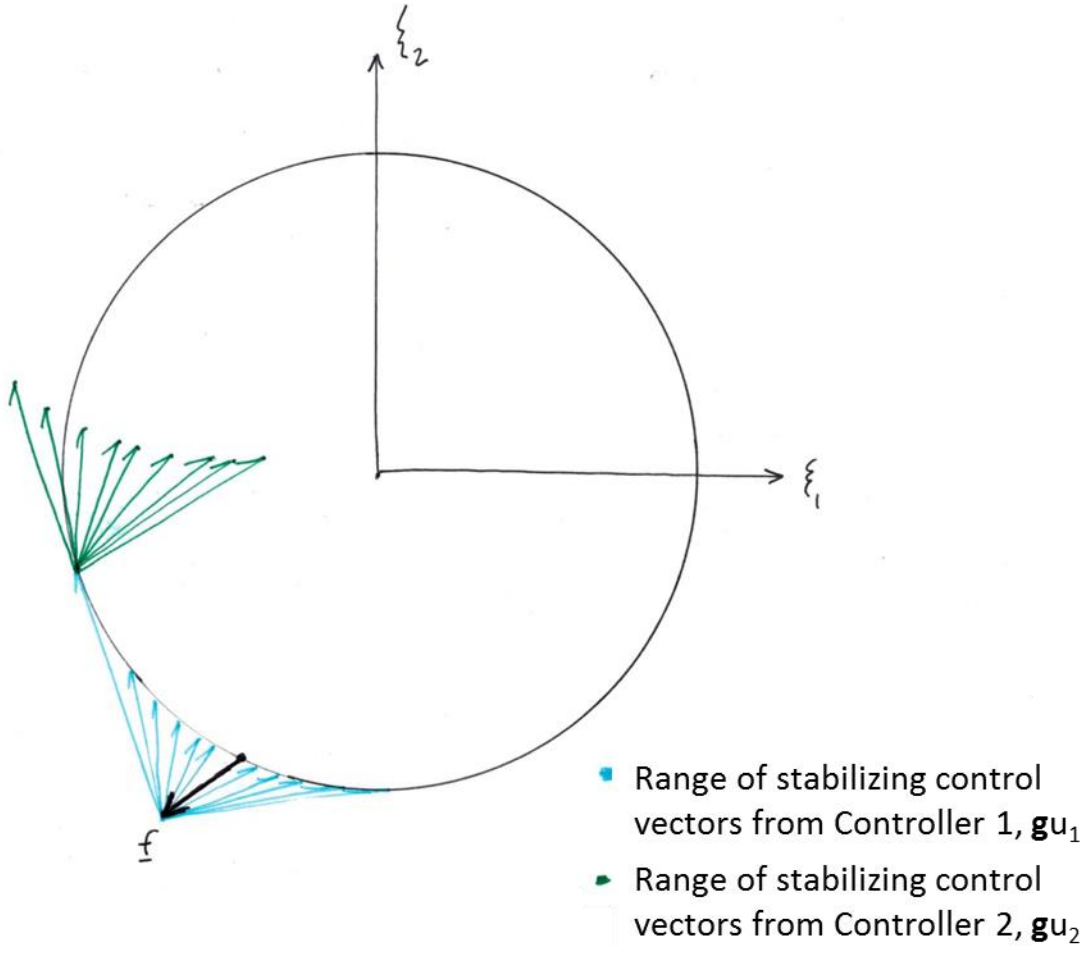


Figure 93. Range of possible sums for the stabilizing control vectors from Controller 1 and Controller 2

Finally, the range of the sum of all three vectors ($\boldsymbol{f} + \boldsymbol{g}u_1 + \boldsymbol{g}u_2$) is shown in red in Figure 94. Note how the net motion of the system is towards the interior of the level curve in any case, i.e. the value of the Lyapunov function will decrease and the system is stable. The same also applies for Lyapunov functions of arbitrary, non-circular shapes (Figure 95).

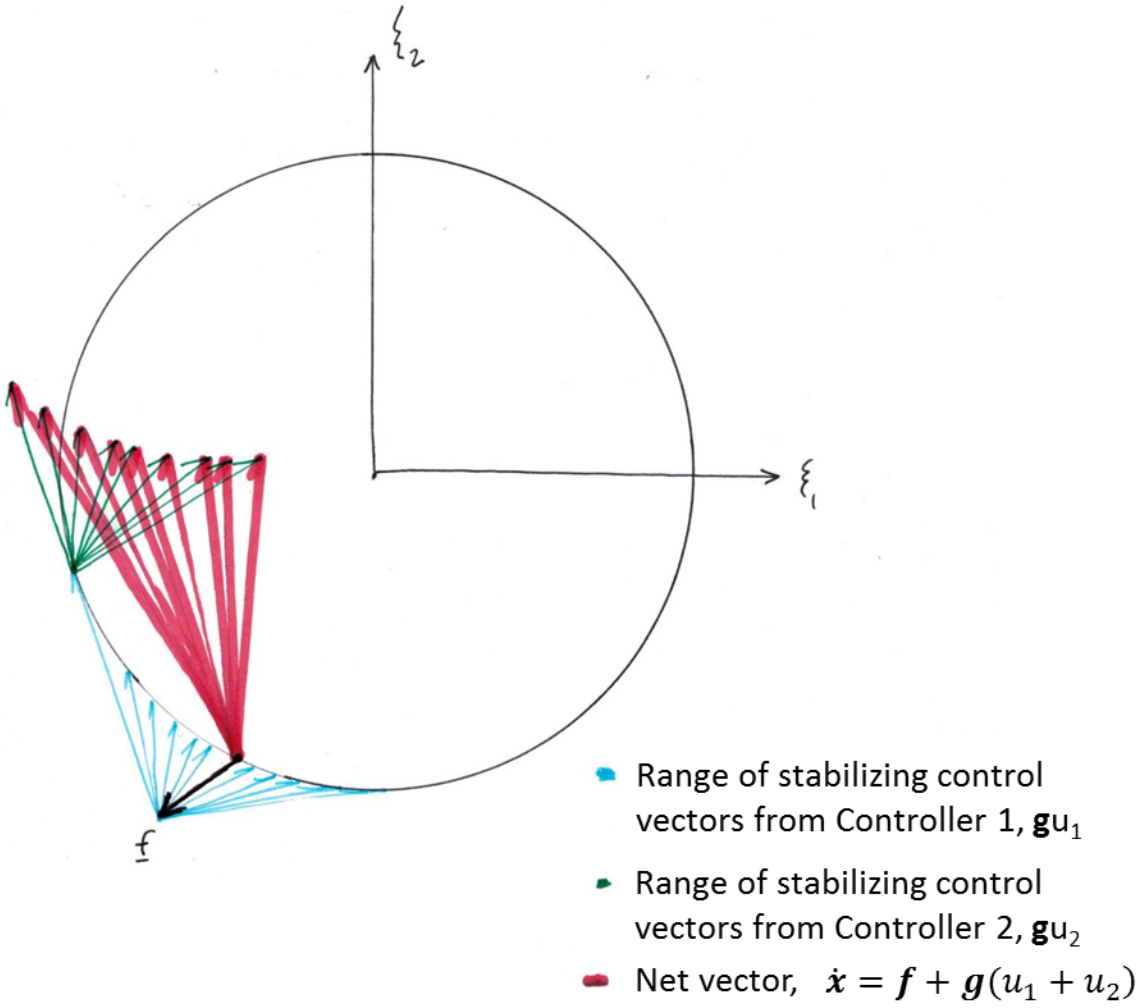


Figure 94. Range of net vectors from the controllers in parallel.

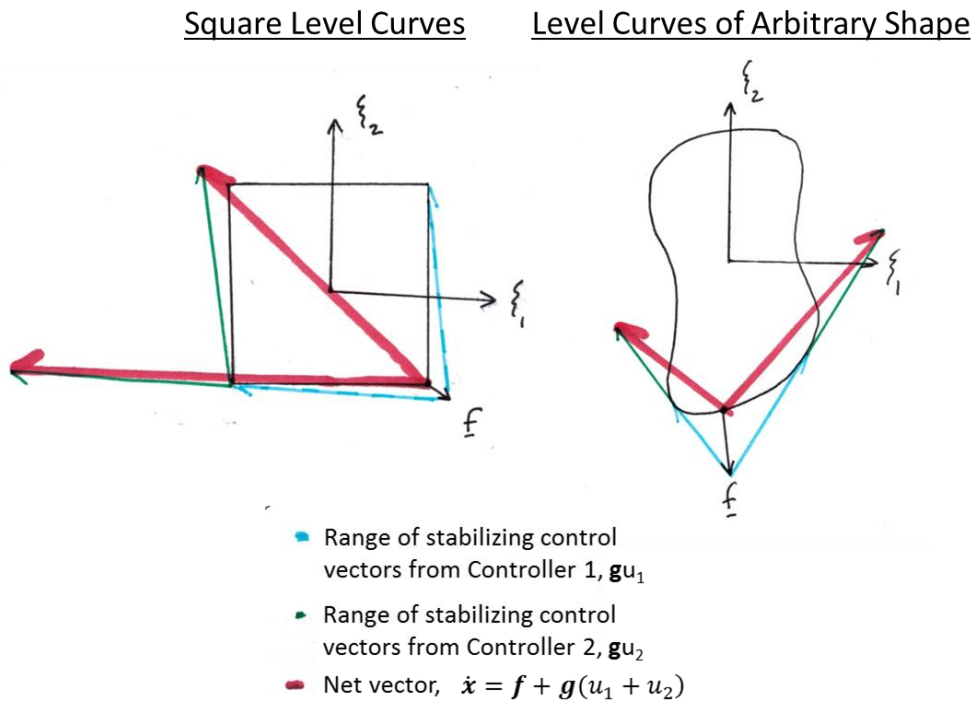


Figure 95. The Proof for Lyapunov Functions of Other Shapes

7.8 GUARANTEED STABILITY DESPITE ACTUATOR LIMITATIONS

Actuator Saturation

To be more realistic, our simulations have included saturation limits on the actuators as saturation limits are always present in physical systems. A technique which allows one to adjust $\dot{V}_{target}$ according to saturation limits follows. Although it is not necessary for stability, it creates more predictable performance by ensuring that the desired $\dot{V}_{target}$ is achievable.

The relationship between $\dot{V}_{target}$ and u was given by Equation 3.37:

$$u = \frac{\dot{V}_{target} - \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i}{\sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \frac{\partial f_i}{\partial u}} \tag{3.37}$$

When saturation limits are present, the modification is:

$$\dot{\mathrm{V}}_{\mathrm{target}} = \left\{ \begin{array}{c} \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i} + \left[\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}} \right] u_{min} \text{ if } \frac{\dot{\mathrm{V}}_{\mathrm{target}} - \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i}}{\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}}} < u_{min} \\ \text{Nominal user} - \text{specified value if } u_{min} < \frac{\dot{\mathrm{V}}_{\mathrm{target}} - \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i}}{\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}}} < u_{max} \\ \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i} + \left[\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}} \right] u_{max} \text{ if } \frac{\dot{\mathrm{V}}_{\mathrm{target}} - \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i}}{\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}}} > u_{max} \end{array} \right\} \tag{7.29}$$

Assuming the band of allowable controls $[u_{min}, u_{max}]$ is large enough to maintain a negative $\dot{\mathrm{V}}_{\mathrm{target}}$, stability remains guaranteed.

Slew Rate Limits

A slew rate limit is a limit on $\dot{u}$, the rate of change of the actuator effort. Of course, all physical systems have slew rate limitations. A formula to calculate the minimum slew rate required to track the user-specified $\dot{V}_{target}$ is:

$$\dot{\mathrm{u}} = \frac{\left(\ddot{V}_{target} - \sum_{i=1}^{n} \left[\frac{\partial}{\partial t} \left(\frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \right) \mathrm{f_i} - \dot{\mathrm{f}}_1 \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \right] \right) \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}} - \sum_{i=1}^{n} \left[\frac{\partial}{\partial t} \left(\frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \right) \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}} - \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} * \frac{\partial}{\partial t} \left(\frac{\partial \mathrm{f_i}}{\partial \mathrm{u}} \right) \right] * \left(\dot{\mathrm{V}}_{\mathrm{target}} - \sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \mathrm{f_i} \right)}{\left(\sum_{i=1}^{n} \frac{\partial \mathrm{V}}{\partial \mathrm{x_i}} \frac{\partial \mathrm{f_i}}{\partial \mathrm{u}} \right)^2} \tag{7.30}$$

Equation 7.30 was derived by differentiating Equation 3.37.

Chapter 8: Summary

Industry has largely depended on the PID controller for autonomous control since its conception in the early 20th century, but the PID algorithm suffers from many drawbacks. Often it is not suitable for nonlinear systems and our preliminary observations of students using it have shown that inexperienced engineers frequently misapply the algorithm in potentially dangerous ways. There are many other, more sophisticated nonlinear control algorithms that perform better but they generally suffer from a lack of scope (e.g. feedback linearization) or heavy computational cost (e.g. optimal control). A new class of nonlinear controller that performs similarly well to PID controllers on simple control problems while being extensible to large, highly nonlinear dynamic systems has been presented. This chapter summarizes the work and suggests avenues for research that would broaden its scope and increase its impact.

8.1 SUMMARY

Chapter One presents a perspective on modern controls education and describes the gaps in control theory that the switched Lyapunov controller fills. It also covers the theoretical background that is necessary to understand Lyapunov controller synthesis.

Chapter Two surveys the nonlinear control literature, which is a necessary step for the validation of any new algorithm. It describes how the switched Lyapunov controller is unique as a computationally fast algorithm which is applicable to large, highly nonlinear systems. Chapter Two also covers controllability analysis, an important tool for the design of control systems.

Chapter Three describes the switched Lyapunov control algorithm, starting with the proof for a low-order, single-input system and generalizing to systems of any size and any number of inputs.

Chapter Four covers the simulation results for a broad array of dynamic systems ranging from simple first-order, single-input systems up to seventh-order, seven-input. These simulations were conducted with a MATLAB graphical user interface which was designed to be easy to use.

Chapter Five presents the ROS control package and applies it to several practical problems. This ROS package was intended for easy integration with hardware and is capable of conducting much larger simulations than the MATLAB equivalent. The applications were an inverted pendulum simulation, a compliant robot demonstration on physical hardware, and simulation of a 14^{th}-order, highly nonlinear system.

Chapter Six reviews the results after four students compared a switched Lyapunov controller directly against PID controllers on a second-order system. The Lyapunov controller was (understandably) confusing for some students who had no previous experience with it; it was useful to learn that tuning $\dot{V}_{target}$ was their biggest challenge and switching to dB units has helped alleviate that concern. Several other improvements were proposed and tested. Although the PID controllers were faster to tune, the students relied on intuition, which led to some potentially destabilizing parameter sets.

Chapter Seven addresses the practical limitations of the switched Lyapunov control algorithm. It was motivated by feedback from new users of the algorithm as well as expert input. The issues addressed in Chapter Seven include

- The appropriateness of decibel units
- Isotropic error measurements
- The effect of numerical integration accuracy
- The effect of different linearization techniques
- Integral action
- A specialized controller for nonlinear systems in normal form
- The effect of actuator limitations
- A discussion of robustness, including several types of disturbances

Based on this effort, the controller is more user-friendly and we often have a definitive answer for whether the controller will be able to stabilize a given type of system with a given variety of disturbance.

8.2 RECOMMENDATIONS FOR FUTURE WORK

This research has revealed two avenues for future research that seem particularly promising:

- The preliminary feedback involving PID controller synthesis revealed some surprising deficiencies in PID controller education. Unfortunately our very small sample size is not enough to draw statistically significant conclusions, but additional research into the topic of common PID pitfalls would be insightful and useful to society, especially considering the prevalence of PID controllers. It seems likely that a few minor adjustments in the classroom could greatly improve students' comprehension of the algorithm. A literature search on topics related to education and control revealed little

research on the topic, so bringing this discussion to the attention of a larger audience would be a valuable contribution.

- Lazar, Teel et al. [2009] have stated that in the most general case of a discrete-time, discontinuous system, a "uniformly strict" Lyapunov function is required to ensure stability. The Lyapunov functions V1 and V2 which we suggest are not uniformly strict. It would be a useful extension of our work to apply the same concept but redefine V1 and V2 as norms, so it could easily be proven that they are uniformly strict. This would ensure that discrete implementations would be stable as well as continuous implementations.
- The most significant drawback of the switched Lyapunov controller may be chatter. The root cause of this chatter is the tapering of $\dot{V}_{target}$, which requires a very large $\dot{V}_{target}(0)$ to ensure asymptotic stability. We gave two examples where a low-pass filter was applied to reduce this chatter, but that approach is non-ideal because it requires additional tuning. To eliminate chatter completely would greatly improve the controller's practicality.

Since the switched Lyapunov controller concept is new, it is not well-understood yet. Thus, other topics of research to consider include:

- What control rate is required to stabilize a particular system?
- Can the regions where a second CLF is necessary be predicted?
- What are the classes of systems for which the alternative CLF will never be necessary? In the simulations presented here, V_2 was rarely required, which simplifies the algorithm greatly.
- McCourt [2015] makes the point that the availability of multiple CLF's allows the possibility of selecting the control mode based on a cost function.

This might allow switching in order to increase robustness or convergence rate, for example, in addition to ensuring stability.

8.3 CONCLUDING REMARKS

The original objective of this dissertation was to develop a control algorithm which makes nonlinear control more accessible, and that objective was achieved with the following contributions:

- Theoretical foundations of a Lyapunov-based nonlinear controller which:
  - Is computationally fast.
  - Is extensible to high order systems.
  - Performs similarly to or better than PID control representative canonical systems from the literature.
  - Includes a parameter set which can be intuitively tuned is invariant with respect to system complexity.
  - Eliminates the need to derive a Lyapunov function.
- Extensive simulations and a hardware demonstration to verify accuracy of the controller's derivation and validate it as a useful tool.
- Two open-source software packages for nonlinear control:
  - An easy-to-use MATLAB GUI
  - A high-performance ROS node for hardware integration and large/complex systems

The switched Lyapunov controller has the potential to make nonlinear control more widely available and its release as the first ROS nonlinear control package is a significant step in that direction. As of December 2015, the ROS package receives approximately 20 pageviews daily so it appears the package is already finding use in the real world.

Unexpectedly, this study also lead to the observation that PID controllers are often being tuned incorrectly and dangerously. Those initial results infer a new avenue of study which may eventually prove even more significant than the original scope of work.

Appendix A: Preliminary Controller Comparison - Instructions

These are the instructions for the exercise that was given to three students to compare the switched Lyapunov controller to PID control.

Instructions

Hello, thanks for helping me! The objective of this survey is to compare a new nonlinear control technique against a classic technique you learned as an undergrad: the PID controller. The survey requires a computer running Ubuntu with ROS Indigo or Jade installed. (http://wiki.ros.org/jade/Installation)
I don't know how long the survey will take. It might be 3 hours if you're familiar with Linux/Ubuntu already, or it might take a week or more. Feel free to ask a grad student if you have a question about the Ubuntu environment, such as:

- How do I open a terminal window? (CTRL+ALT+T)
- How do I stop a process? (CTRL+C)

If you have a question about the controllers or the code, feel free to send me an email (andyz@utexas.edu)

Setting Up

- Open a new terminal (CTRL+ALT+T)
- Create a folder and prepare it for ROS integration.
  - *mkdir –p ~/Desktop/catkin_ws/src*
  - *cd ~/Desktop/catkin_ws/src*
  - *catkin_init_workspace*
- Clone the repository (similar to downloading the code)
  - *git clone https://github.com/AndyZelenak/pid_lyap.git*
  - *cd ..*
- Compile
  - *catkin_make*
- Specify use of the new setup file
  - *source devel/setup.bash*
    - You will have to do this every time you open a new terminal window.
  - OR, add this line to the end of Home/bashrc to handle it automatically:
    - *source ~/Desktop/catkin_ws/devel/setup.bash*
- Check that it runs
  - *roscore*
  - In a new terminal window: *rosrun pid_lyap lyap_controller*
  - You should see about 5 lines of text output.

- Close all terminals. (CTRL+C)

Section 1- Tuning the Lyapunov controller

You will be using the new 'Lyapunov controller' to control the second-order system:

$$\dot{x}_0 = x_0 + u_0$$
$$\dot{x}_1 = x_1 - 100 * u_1$$

x_i is a 'state.' For example, it could be the angular velocity of a motor.
u_i is an input to the system. For example, it could be the current to the motor.
Your goal is to track the setpoints:

$$x_{0,setpoint}(t) = \sin(4t^2)$$
$$x_{1,setpoint}(t) = t$$

The Lyapunov controller is based on the 1892 work of a Russian mathematician. The idea is to move along a bowl-like surface where the setpoint is at the bottom of the bowl (Figure 96). You will define the parameters that determine how quickly the system moves towards the bottom of the bowl.

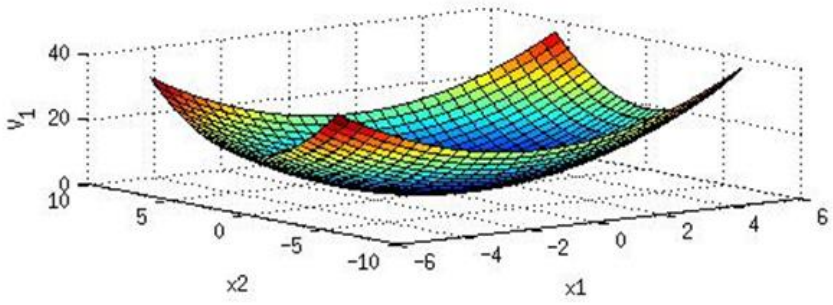


Figure 96. Bowl-like surface with the setpoint at the bottom. The setpoint is (0,0) in this case.

Open catkin_ws/src/pid_lyap/src/lyap_controller.cpp in a text editor. Notice lines 14-16. These are the only 3 lines that you're allowed to change for this exercise.

- $\dot{V}_{target,initial}$ determines how quickly the system moves towards the bottom of the bowl. It must be a negative value. A very large negative value moves faster down the bowl, but it is too negative, the system might overshoot or become unstable. You need to specify a good value. (The default -1.0 is not negative enough.)
- The *high_saturation_limit* defines the largest possible u_i's. In the motor example, this is like specifying a maximum current to a motor. You need to specify good values for u_0 and u_1. Again, the default values will not work very well.
- The *low_saturation_limit* defines the smallest possible u_i's. In the motor example, this is like specifying the most negative current to a motor. You need to specify good values for u_0 and u_1. Again, the default values will not work very well.

To run the system:

- Open a terminal (CTRL+ALT+T) and compile any changes you might have made:
  - *cd ~/Desktop/catkin_ws*
  - *catkin_make*
- Open another terminal and start roscore
  - *roscore*
- Open another terminal and start the Lyapunov controller
  - *rosrun pid_lyap lyap_controller*
- Open another terminal and simulate the system.
  - *rosrun pid_lyap plant*
- Open another terminal and start plotting the data
  - *rqt_plot /state/x /state/setpoint*
  - Play around with the plotting interface. Autoscroll is a nice feature. You can also edit axis limits by clicking the check.
  - You should see something similar to Figure 97. x_0 is certainly not stable.

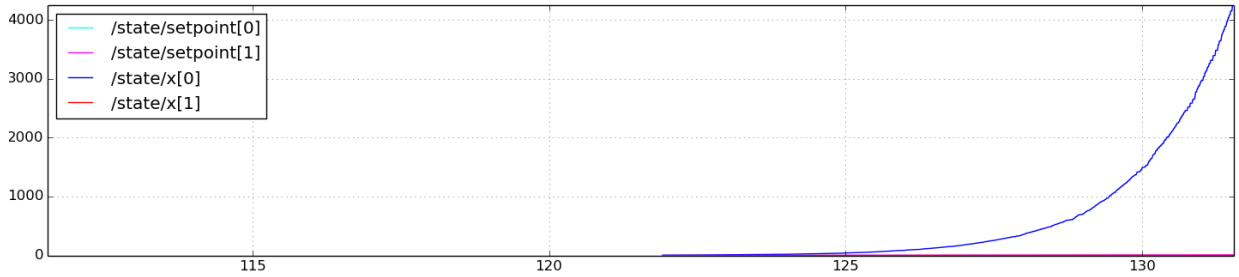


Figure 97. Unstable with the default parameters.

- Tune the parameters and recompile until you are satisfied with the performance. My results are shown in Figure 98.

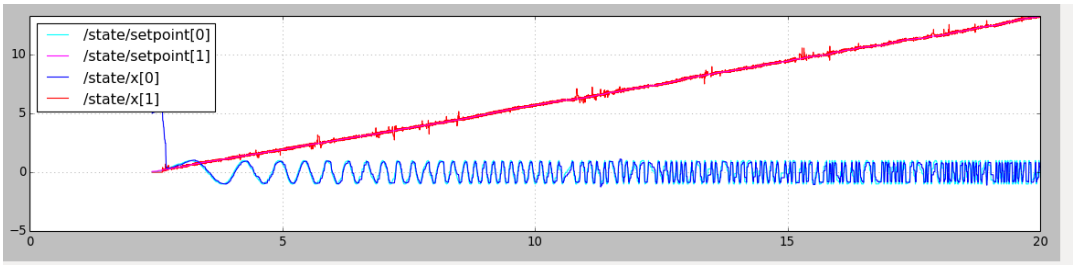


Figure 98. Well-tuned Lyapunov controller

- When your controller is well-tuned, save an image of the plots (like Figure 98). Adjust the axes to capture at least 10s of data at the beginning of the simulation. Then fill out Section 1 on the Questionnaire. Close all of the terminals.

Section 2- Tuning the first PID controller

The Lyapunov controller was able to stabilize both states simultaneously, but a PID controller can only stabilize one. In this section, we stabilize x_0. To run the PID controller:

- Run roscore in a new terminal.
  - *roscore*
- Start the PID controller in a new terminal.
  - *rosrun pid_lyap pid_controller1 1 0.02 0.03 10000*
  - *The first argument (1) is the proportional gain.*
  - *The second argument (0.02) is the integral gain.*
  - *The third argument (0.03) is the derivative gain.*
  - *The fourth argument (10000) is the simulation refresh rate. Do not adjust this parameter.*
- Open a new terminal for plotting.
  - *rqt_plot /state/x[0] /state/setpoint[0]*
- Open a new terminal and start the simulation.
  - *rosrun pid_lyap plant*
- The default parameters will not be good, but you should see 2 lines on the plot, as in Figure 98.

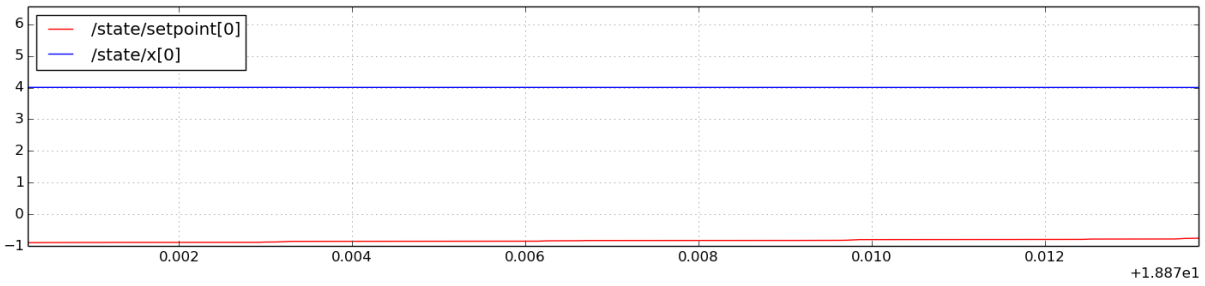


Figure 99. Initial PID1 simulation

- Adjust the P, I, and D gains and run the system again until you are satisfied with the performance. Then:
  - Save an image of the plots. Make sure the plot captures at least 10s of data from the start of the simulation. The results from a well-tuned controller are shown in Figure 100.
  - Complete Section 2 of the Questionnaire.
  - Close all terminals.

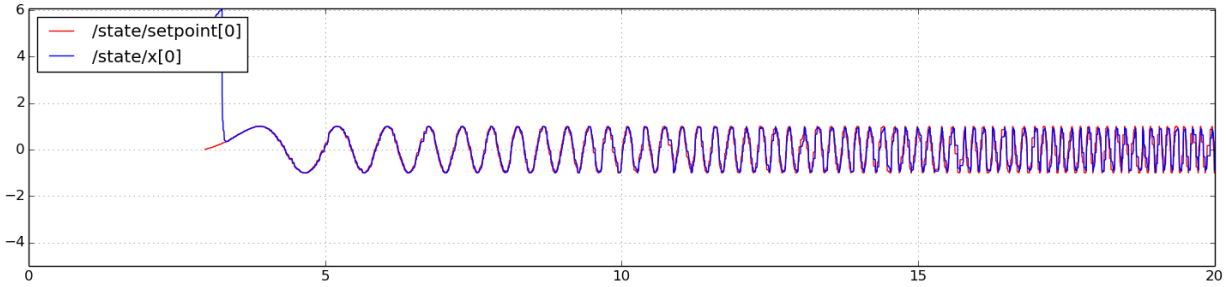


Figure 100. Well-tuned PID1

Section 3- Tuning the second PID controller

Now we will essentially repeat Section 2, using a different PID controller to stabilize the second state.

- Run roscore in a new terminal.
  - *roscore*
- Start the PID controller in a new terminal.
  - *rosrun pid_lyap pid_controller2 1 0.02 0.03 10000*
  - *The first argument (1) is the proportional gain.*
  - *The second argument (0.02) is the integral gain.*
  - *The third argument (0.03) is the derivative gain.*
  - *The fourth argument (10000) is the simulation refresh rate. Do not adjust this parameter.*
- Open a new terminal for plotting. Note that we are plotting index 1 now.
  - *rqt_plot /state/x[1] /state/setpoint[1]*
- Open a new terminal and start the simulation.
  - *rosrun pid_lyap plant*
- Adjust the P, I, and D gains and run the system again until you are satisfied with the performance. Then:
  - Save an image of the plots. Make sure the plot captures at least 10s of data from the start of the simulation. The results from a well-tuned controller are shown in Figure 101.
  - Complete Section 3 of the Questionnaire.
  - Close all terminals.

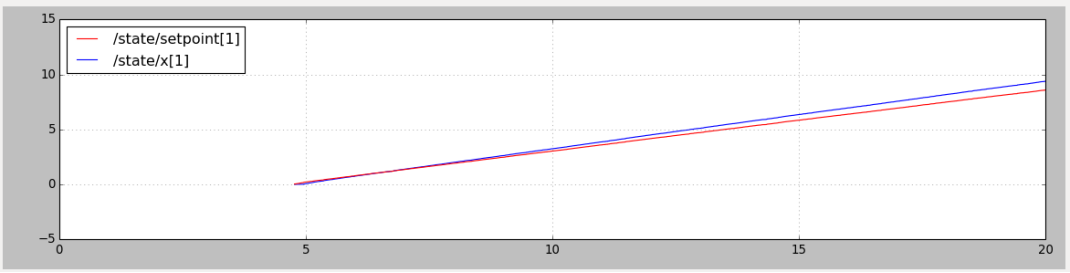


Figure 101. Well-tuned PID2

Now please fill out the rest of the Questionnaire. Thank you! Send the Questionnaire and all 3 plots to andyz@utexas.edu.

Appendix B: Preliminary Controller Comparison - Questionnaire

This is the questionnaire to be filled in as the students completed the controls comparison.

Questionnaire

Section 1 – Tuning the Lyapunov controller
How long did you spend on Section 1 before you were satisfied with the performance of the controller?
_____(hours)

What were the final parameters of your controller?
V_dot_target_initial: _____
high_saturation_limit: _____
low_saturation_limit: _____

Section 2 – Tuning the first PID controller
How long did you spend on Section 2 before you were satisfied with the performance of the controller?
_____(hours)

What were the final parameters of your controller?
Proportional gain: _____
Integral gain: _____
Derivative gain: _____

Section 3 – Tuning the second PID controller
How long did you spend on Section 3 before you were satisfied with the performance of the controller?
_____(hours)

What were the final parameters of your controller?
Proportional gain: _____
Integral gain: _____
Derivative gain: _____

To be answered when all 3 sections are complete
Did you notice any performance differences between the Lyapunov controller and the PID controllers?

Was it easier to stabilize both states at once with the Lyapunov controller, or was it easier to stabilize each state individually with two PID controllers?

If you had a system with 5 states, would you rather tune 5 separate PID controllers? Or would you rather stabilize all 5 simultaneously with a Lyapunov controller?

Was it easy to recall how to tune a PID controller?

Was it easy to learn how to tune the Lyapunov controller from the description in the instructions?

Appendix C: Preliminary Controller Comparison - Responses

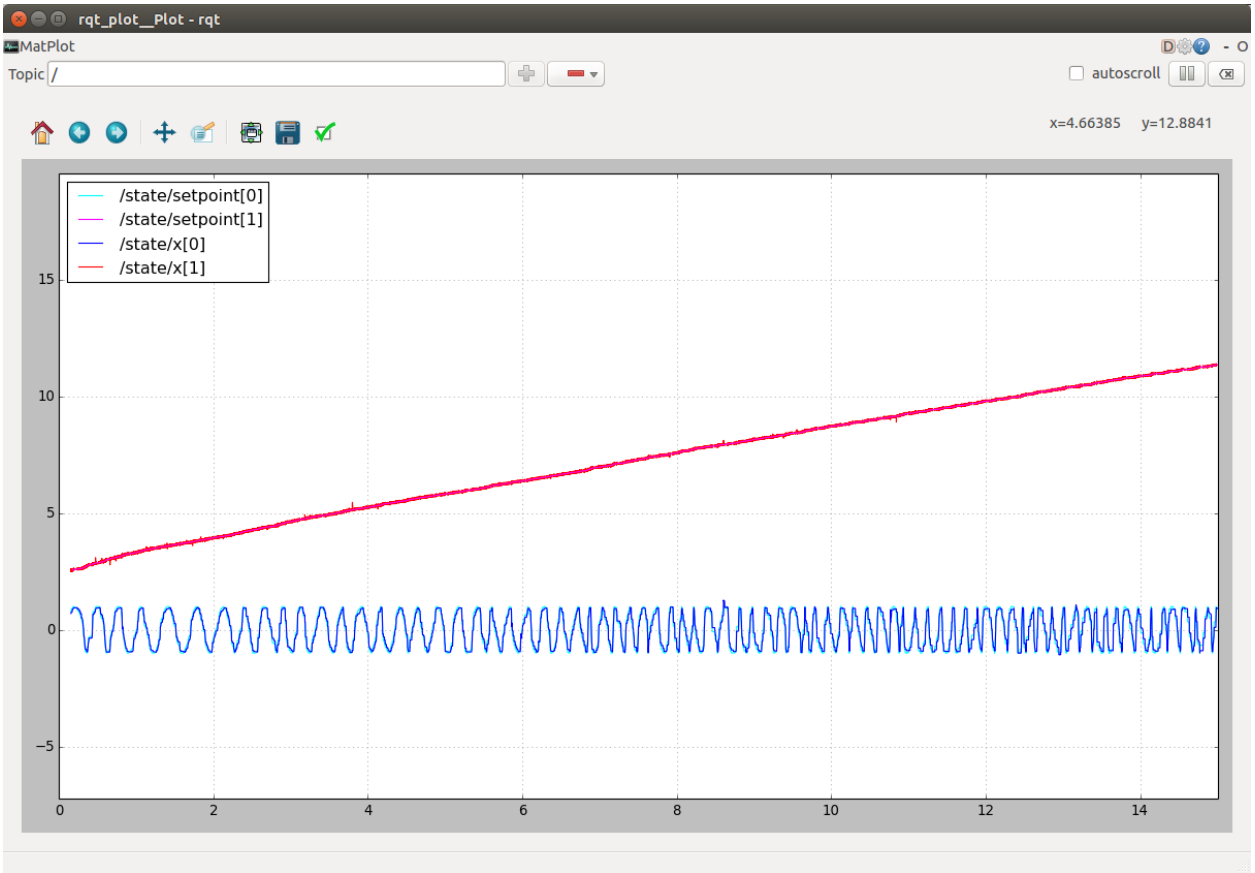
These are the responses to the preliminary controller comparison.

Questionnaire 1

Section 1 – Tuning the Lyapunov controller

How long did you spend on Section 1 before you were satisfied with the performance of the controller?
2 (hours)

What were the final parameters of your controller?
V_dot_target_initial: -200000000.0
high_saturation_limit: {900., 1.}
low_saturation_limit: {-900., -1.}
Plot:

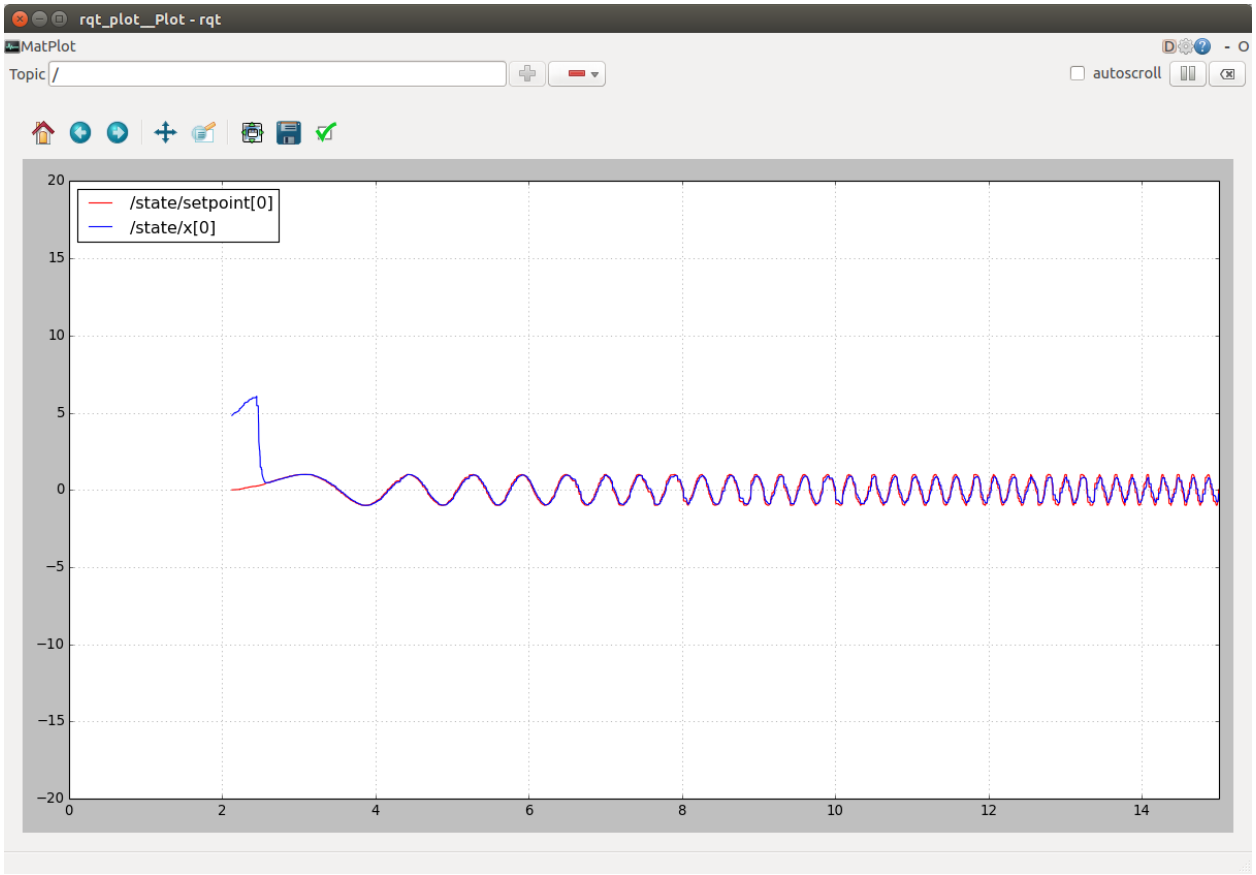


Section 2 – Tuning the first PID controller

How long did you spend on Section 2 before you were satisfied with the performance of the controller?

0.3 (hours)

What were the final parameters of your controller?
Proportional gain: 110
Integral gain: 1.7
Derivative gain: 0.9
Plot:

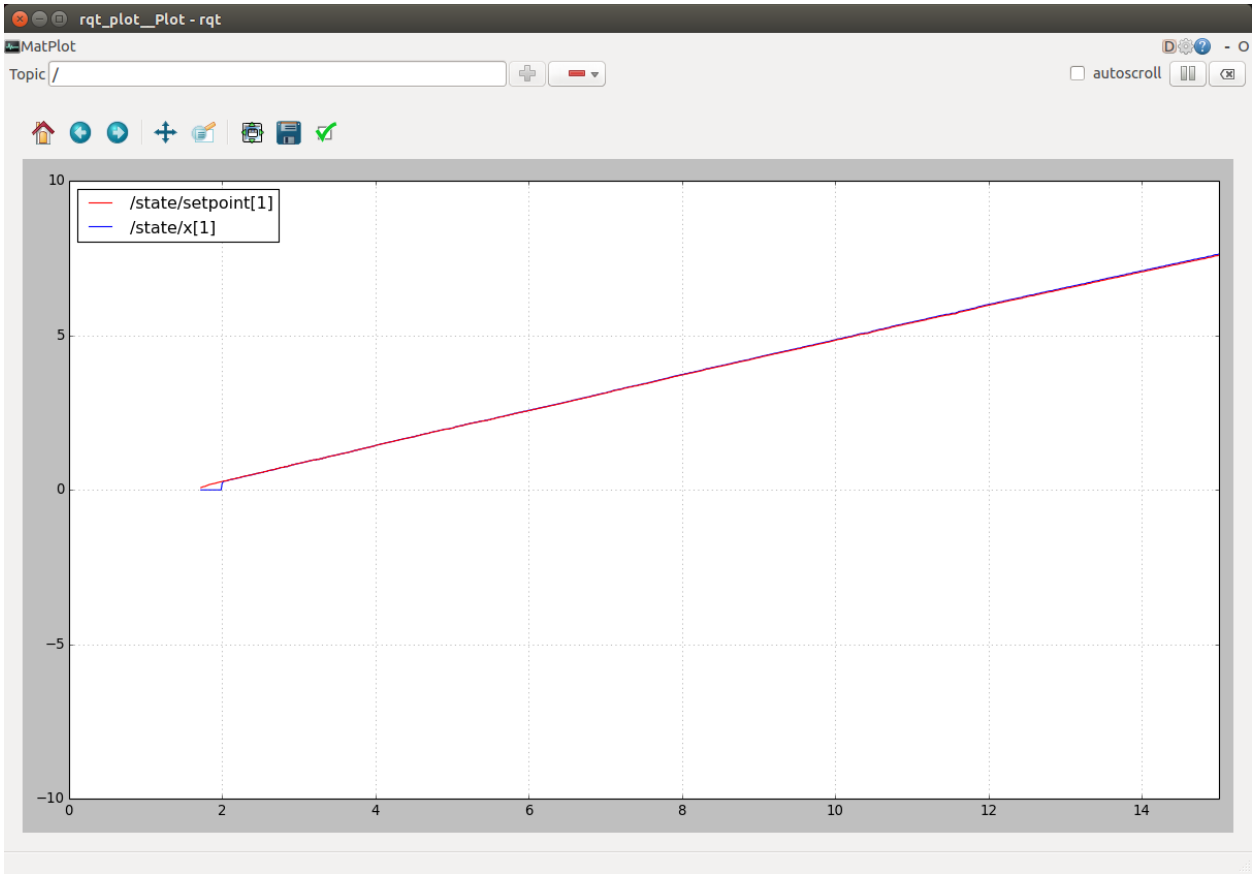


Section 3 – Tuning the second PID controller
How long did you spend on Section 3 before you were satisfied with the performance of the controller?

0.5 (hours)

What were the final parameters of your controller?
Proportional gain: -2.1
Integral gain: 0.1
Derivative gain: 0.0

Plot:



To be answered when all 3 sections are complete

Did you notice any performance differences between the Lyapunov controller and the PID controllers?

The PID controller seemed to track more smoothly once it was tuned correctly.

Was it easier to stabilize both states at once with the Lyapunov controller, or was it easier to stabilize each state individually with two PID controllers?

For me, it was easier to stabilize the two states individually. That might have had something to do with the fact that I was plotting both states together on the Lyapunov controller and if one blows up you have no idea what is happening with either.

If you had a system with 5 states, would you rather tune 5 separate PID controllers? Or would you rather stabilize all 5 simultaneously with a Lyapunov controller?

I think that with more and more states, the Lyapunov would become more appealing. And it felt like with practice the Lyapunov became a little bit more intuitive. So yeah, with 5 states I would probably rather tune a Lyapunov.

Was it easy to recall how to tune a PID controller?

There were certain things that were easy to recall like integral gain getting rid of steady state error. Otherwise, it was kind of guess and check. I was seriously thrown off by the negative proportional gain in the last section because I thought that gains were always positive.

Was it easy to learn how to tune the Lyapunov controller from the description in the instructions?

It was easy once I got the system stable. It took a long time to figure out the play between the limits and the speed variables and how to make sure there was stability. Once I got that figured out, the tuning became a lot easier.

Questionnaire 2

Section 1 – Tuning the Lyapunov controller

How long did you spend on Section 1 before you were satisfied with the performance of the controller?

_____1_____(hours)

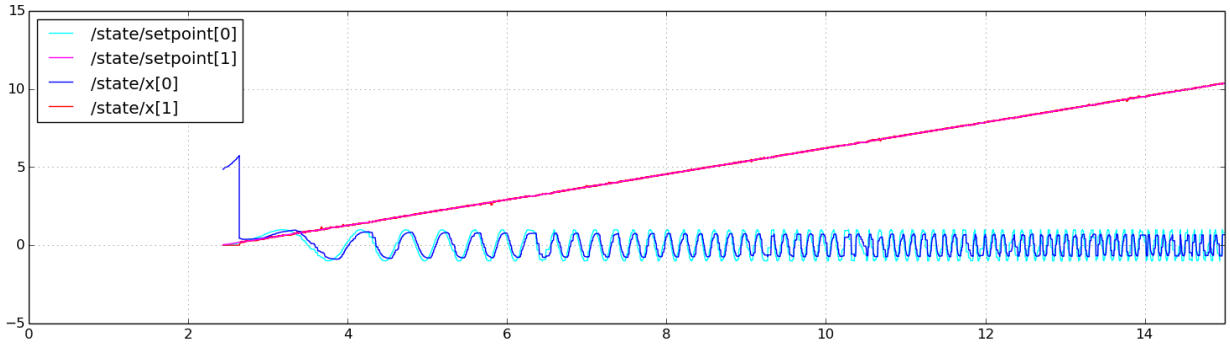
What were the final parameters of your controller?

V_dot_target_initial: _____-1000000_____

high_saturation_limit: _____50000, 0.3_____

low_saturation_limit: _____-50000, -0.3_____

Plot:



Section 2 – Tuning the first PID controller

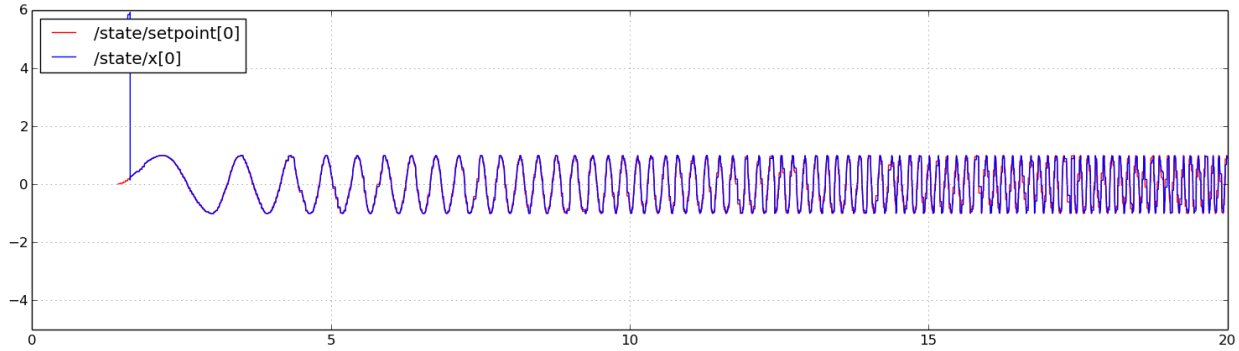
How long did you spend on Section 2 before you were satisfied with the performance of the controller?

_____0.25_____(hours)

What were the final parameters of your controller?

Proportional gain: _____300_____

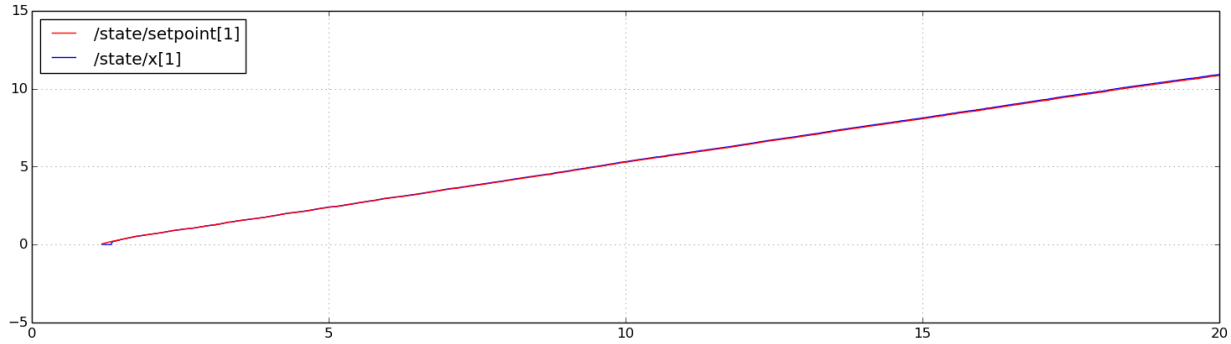
Integral gain: _____0.02_____
Derivative gain: _____0.05_____
Plot:



Section 3 – Tuning the second PID controller
How long did you spend on Section 3 before you were satisfied with the performance of the controller?

_____0.25_____(hours)

What were the final parameters of your controller?
Proportional gain: _____-1.0_____
Integral gain: _____-0.1_____
Derivative gain: _____0.0_____
Plot:



To be answered when all 3 sections are complete
Did you notice any performance differences between the Lyapunov controller and the PID controllers?

- Specifying saturation limits means that the Lyapunov controller, in simulation, eventually grows unstable for both scenarios, where the PID does not.

Was it easier to stabilize both states at once with the Lyapunov controller, or was it easier to stabilize each state individually with two PID controllers?

- It was much easier to stabilize each state with the two PID controllers, although I did have to look at the state equations to realize that I would need negative gains for the second PID controller

If you had a system with 5 states, would you rather tune 5 separate PID controllers? Or would you rather stabilize all 5 simultaneously with a Lyapunov controller?

- It would really depend on the situation, but based on this experience I would rather tune 5 separate PID controllers

Was it easy to recall how to tune a PID controller?

- Yes, because the names all make intuitive sense

Was it easy to learn how to tune the Lyapunov controller from the description in the instructions?

- No, because I didn't immediately understand the direct implications of changing the different parameters on the system, and I ended up having to use really large numbers to get the oscillating setpoint system to work, which made me think at first that I was doing something wrong.

Questionnaire 3

Section 1 – Tuning the Lyapunov controller

How long did you spend on Section 1 before you were satisfied with the performance of the controller?

_____3_____(hours)

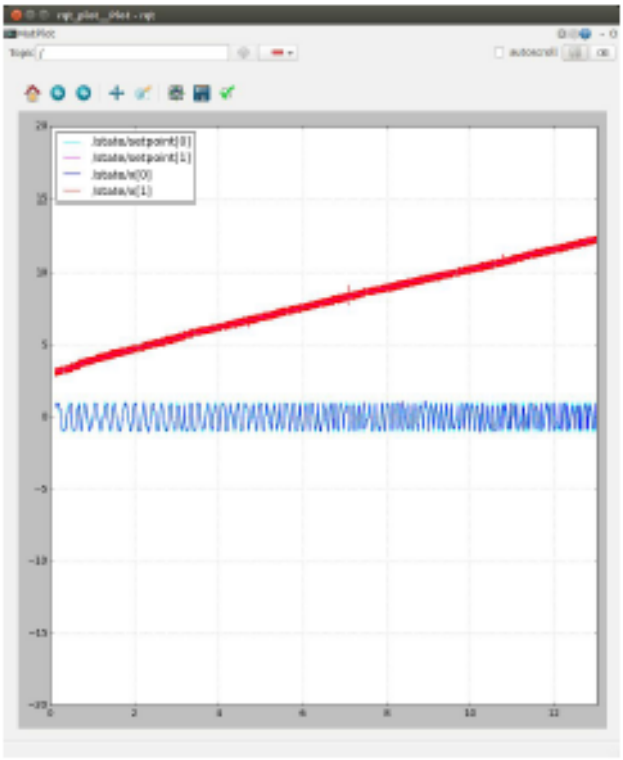
What were the final parameters of your controller?

V_dot_target_initial: -100000000

high_saturation_limit: {1000, 5}

low_saturation_limit: {-1000, -5};

Plot:



Additional comments: Because I have never worked with this controller, I lacked the intuition needed to understand how altering parameters would affect the graph. I had to ask Alex (after being stuck for

1.5 hours) why State/x[0] was not following the setpoint, and he basically told me V had to be ridiculously high.

Section 2 – Tuning the first PID controller

How long did you spend on Section 2 before you were satisfied with the performance of the controller?

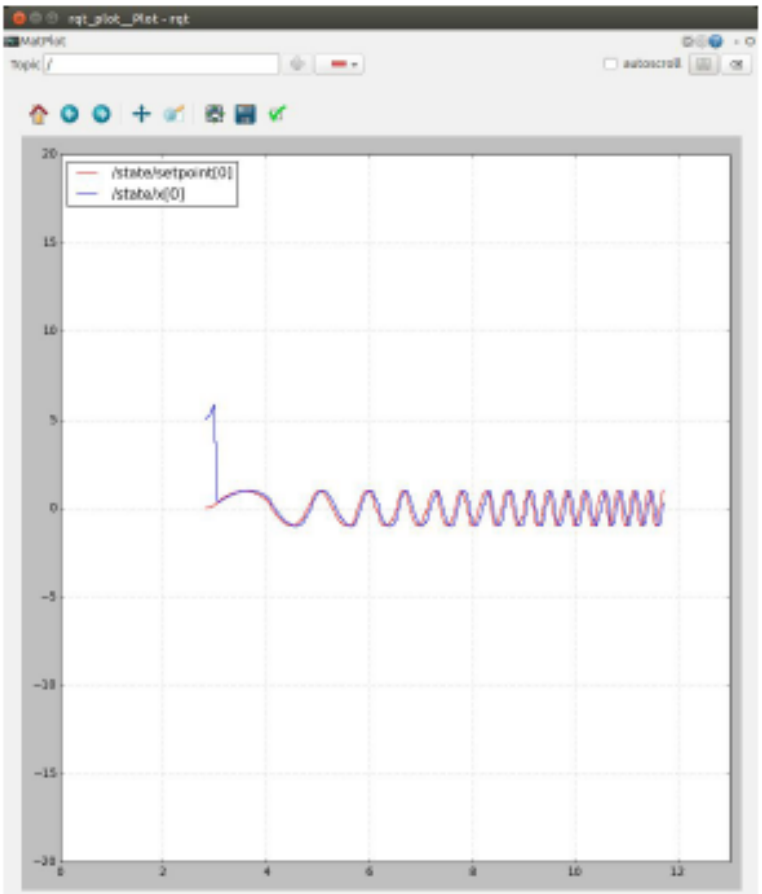
_____0.25_____(hours)

What were the final parameters of your controller?

Proportional gain: _____30_____

Integral gain: _____0.05_____

Derivative gain: _____-.45_____



Section 3 – Tuning the second PID controller

How long did you spend on Section 3 before you were satisfied with the performance of the controller?

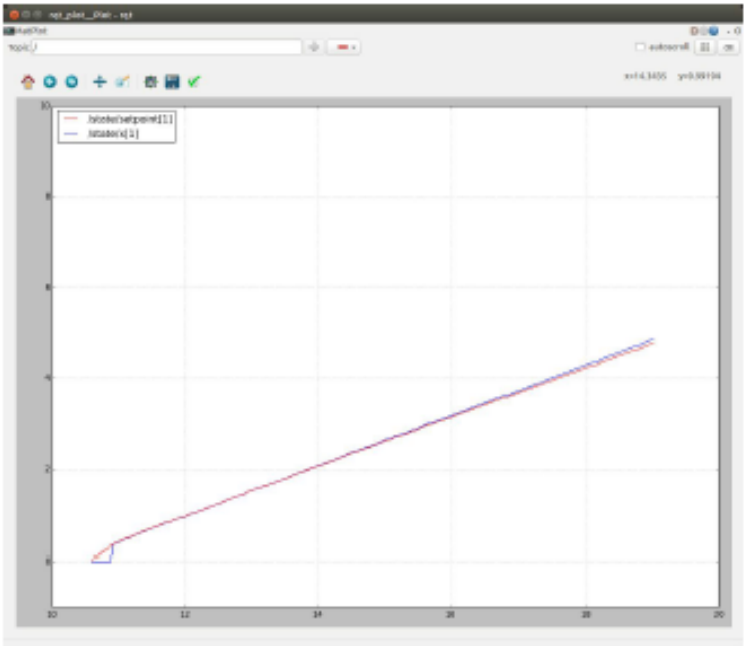
_____0.3333_____(hours)

What were the final parameters of your controller?

Proportional gain: _____-1_____

Integral gain: _____0.5_____

Derivative gain: _____0.0005_____



To be answered when all 3 sections are complete

Did you notice any performance differences between the Lyapunov controller and the PID controllers?

I did not.

Was it easier to stabilize both states at once with the Lyapunov controller, or was it easier to stabilize each state individually with two PID controllers?

It was easier for me to stabilize each state individually than with the Lyap controller.

If you had a system with 5 states, would you rather tune 5 separate PID controllers? Or would you rather stabilize all 5 simultaneously with a Lyapunov controller?

Maybe if I were better acquainted at Lyap, I would find that to be easier, but right now, it's so much faster to just tune pid controllers. (It'd be even faster to tune PID controllers if I had access to Nyquist and other MATLAB functions too.)

Was it easy to recall how to tune a PID controller?

Relatively so. I did have to double check myself and refer back to Dr. Pryor's notes on PID. But for the most part, I had an intuition on how to tune it.

Was it easy to learn how to tune the Lyapunov controller from the description in the instructions?

No. I didn't really understand what each parameter did based on your descriptions for them. It was only when I gave up and asked Alex to explain to me did I actually understand what needed to be done.

Questionnaire 4

Section 1 – Tuning the Lyapunov controller

How long did you spend on Section 1 before you were satisfied with the performance of the controller?

_____1.1_____(hours)

What were the final parameters of your controller?

V_dot_target_initial: ____-1000000000.0_____

high_saturation_limit: _____{100, 5};_____

low_saturation_limit: _____{-100, -5};_____

Section 2 – Tuning the first PID controller

How long did you spend on Section 2 before you were satisfied with the performance of the controller?

_____0.75___(hours)

What were the final parameters of your controller?

Proportional gain: _____1000_____

Integral gain: _____-0.15_____

Derivative gain: _____0.02_____

Section 3 – Tuning the second PID controller

How long did you spend on Section 3 before you were satisfied with the performance of the controller?

_____1.0_____(hours)

What were the final parameters of your controller?

Proportional gain: _____-20_____

Integral gain: _____-8_____

Derivative gain: _____-0.005_____

To be answered when all 3 sections are complete

Did you notice any performance differences between the Lyapunov controller and the PID controllers?

The Lyapunov seemed simpler to tune as well as achieved a better result. It was easier to see how the changes affected the system as a whole, rather than one by one.

Was it easier to stabilize both states at once with the Lyapunov controller, or was it easier to stabilize each state individually with two PID controllers?

It was easier to tune both states at once versus each state independently. It was difficult to know the correct range numbers needed for tuning the parameters due to lack of experience with this controller.

If you had a system with 5 states, would you rather tune 5 separate PID controllers? Or would you rather stabilize all 5 simultaneously with a Lyapunov controller?

I'm not sure on this one. I don't know if the linear time needed for PID compares to an unknown time scale for a Lyapunov controller needs. However, if the approach is the same as in section 1, then using the Lyapunov would be preferable.

Was it easy to recall how to tune a PID controller?

Yes. The amount of times we had to do them in controls class made it easy to recall, and knowing the range at which the parameters would work well was easy to recall as well.

Was it easy to learn how to tune the Lyapunov controller from the description in the instructions?

Yes and no. I felt that the broad overview of the Lyapunov controller is okay, it was harder to understand how the variables affected the final stability of the system.

References

Ananthan, A., 2011, “Generating C code from your MATLAB algorithms,” from http://blogs.mathworks.com/loren/2011/11/14/generating-c-code-from-your-matlab-algorithms/

Annaswamy, A.M., *et al.,* 2008, "Adaptive gain-scheduled controller in the presence of actuator anomalies," *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*. Honolulu, pp. 18-21.

Artstein, Z., 1983, “Stabilization with relaxed controls,” Nonlinear Analysis, Theory, Methods, and Applications, **7**(11), pp. 1163-1173.

Aström, K.J. and Murray, R.M., 2011, *Feedback Systems*, Princeton University Press, Princeton, New Jersey.

Aström, K.J. and Furuta, K., 2000, “Swinging up a pendulum by energy control,” Automatica, **36**(2), pp. 287-295.

Atherton, D.P. and Majhi, S., 1999, “Limitations of PID controllers,” *American Control Conference*, IEEE, **6**, San Diego.

Bennett, S., 1993, “Development of the PID controller,” *Control Systems*, IEEE, **13**(6), pp. 58-62.

Bertsekas, D.P., 1996, *Dynamic programming and optimal control*, Athena Scientific, Belmont, Massachusetts.

Bizon, N., 2010, "On tracking robustness in adaptive extremum seeking control of the fuel cell power plants," *Applied Energy*, 87(10), pp. 3115-3130.

Bogacki, P. and Shampine, L.F., 1989, “A 3(2) pair of Runge-Kutta formulas,” Applied Math Letters, **2**, pp. 321-325.

Branicky, M.S., 1998, “Multiple Lyapunov functions and other analysis tools for switched and hybrid systems,” IEEE Transactions on Automatic Control, **43**(4), pp. 475-482.

Coleman, C.P. and Godbole, D., 1994, “A comparison of robustness: fuzzy logic, PID, & sliding mode control,” Proceedings of the Third IEEE World Congress on Computational Intelligence, pp. 1654-1659.

Cominos, P. and Munro, N., 2002, “PID controllers: recent tuning methods and design to specification,” IEEE Proceedings: Control Theory and Applications, **149**(1), pp. 46-53.

Cook, J.A. and Samad, T., 2009, “Controls curriculum survey,” IEEE Control Systems Society, New York.

Control Tutorials for MATLAB and Simulink, “DC motor position: system modeling,” from

http://ctms.engin.umich.edu/CTMS/index.php?example=MotorPosition&section=SystemModeling

Curtis, J.W. and Beard, R.W., 2004, "Satisficing: a new approach to constructive nonlinear control," IEEE Trans. on Automatic Control, **49**(7), pp. 1090-1102.

Daafouz, J. *et* al., 2002, "Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach," IEEE Transactions on Automatic Control, **47**(11), pp. 1883-1887.

Desborough, L. *et al*., 2002, "Increasing customer value of industrial control performance monitoring-Honeywell's experience," *AlChe Symposium series*, American Institute of Chemical Engineers, New York, pp. 169-189.

Devane, E. and Lestas, I., 2012, "Lyapunov stability theory and applications to wireless networks," Cambridge Centre for Analysis, University of Cambridge, England.

Di Palma, F. and Magni, L., 2007, "On optimality of nonlinear model predictive control," Systems and Control Letters, **56**, pp. 58-61.

Diehl, M. *et al*., 2009, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," Nonlinear Model Predictive Control, pp. 391-417.

Dorf, R.C. and Bishop, R.H., 2008, *Modern control systems*, Pearson Prentice Hall, Upper Saddle River, NJ.

Du, D. *et al.*, 2007, "H∞ of discrete-time switched systems with state delays via switched Lyapunov function approach," IEEE Transactions on Automatic Control, **52**(8), 1520-1525.

Edwards, S., 2012, "abb," from http://wiki.ros.org/abb

Felder, R.M., *et al.,* 2003, "Designing and teaching courses to satisfy the ABET engineering criteria," Journal of Engineering Education-Washington, **92**(1), pp. 7-26.

Feron, E. *et al.*, 1996, "Analysis and synthesis of robust control systems via parameter-dependent Lyapunov functions," IEEE Transactions on Automatic Control, **41**(7), pp. 1041-1046.

Foote, T., 2014, "ROS community metrics report," from http://download.ros.org/downloads/metrics/metrics-report-2014-07.pdf

Freeman, R.A. and Kokotović, P.V., 1996, "Approaches to robust nonlinear control," Robust Control via Variable Structure and Lyapunov Techniques, pp. 1-14.

Freeman, R.A. and Kokotović, P.V., 1996, *Robust Nonlinear Control Design*, Birkhäuser, Boston.

Garcia, D. *et al*., 2005, "PID controller design for multivariable systems using Gershgorin bands," *IFAC World Congress*.

Guillard, H. and Bourlés, H., 2000, "Robust feedback linearization," *Proc. 14th International Symposium on Mathematical Theory of Networks and Systems,* Perpignan, France, pp. 1-6.

Gustafsson, T., 1995, "Modeling and control of a rotary crane," *European Control Conference*, Rome, Italy, pp. 3805-3810.

Hedrick, J.K. and Girard, A., 2005, *Control of Nonlinear Dynamic Systems: Theory and Applications*.

Hencey, B. and Alleyne, A., 2010, "Robust gain-scheduled control," *American Control Conference*, IEEE, pp. 3075-3081, Baltimore, MD.

Hespanha, J.P. and Morse, A. S., 2002, "Switching between stabilizing controllers," *Automatica,* **38**, pp. 1905-1917.

Hogan, N., 1984, "Impedance control: an approach to manipulation," *American Control Conference*, IEEE, pp. 304-313.

Hoorn, G.A. vd., 2012, "fanuc," from http://wiki.ros.org/fanuc

How, J.P. and Frazzoli, E., 2010, "Lecture 18: feedback control systems," from http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-30-feedback-control-systems-fall-2010/lecture-notes/MIT16_30F10_lec18.pdf

Hsu, S.B. and Huang, T.W., 1995, "Global stability for a class of predator-prey systems," SIAM J. on Applied Mathematics, **55**(3), pp. 763-783.

Iglesias, P.A. and Ingalls, B.P., 2010, *Control Theory and Systems Biology*, MIT Press, Boston, MA.

Jiang, Z.P. and Hill, D.J., 2002, "A robust adaptive backstepping scheme for nonlinear systems with unmodeled dynamics," IEEE Transactions on Automatic Control, **44**(9), pp. 1705-1711.

Jiang, Z.P. and Nijmeijer, H., 1997, "Tracking control of mobile robots: a case study in backstepping," Automatica, **33**(7), pp. 1393-1399.

Jiang, Z.P. and Praly, L., 1998, "Design of robust adaptive controllers for nonlinear systems with dynamic uncertainties," Automatica, **34**(7), pp. 825-840.

Khalil, H., 1996, *Nonlinear Systems*, Prentice Hall, Upper Saddle River, NJ.

Kokotović, P.V., 1992, "Bode lecture: the joy of feedback," IEEE Control Systems Magazine, **3**, pp. 7-17.

Krstić, M. *et al*., 1992, "Adaptive nonlinear control without overparameterization," Systems and Control Letters, **19**, pp. 177-185.

Krstić, M. and Wang, H.H., 2000, "Stability of extremum seeking feedback for general nonlinear dynamic systems," Automatica, **36**(4), pp. 595-601.

Lecoq, J., 2012, "MATLAB is no longer slow at for loops," from http://www.matlabtips.com/matlab-is-no-longer-slow-at-for-loops/

Liberzon, D. and Morse, S., *[67]* 1999, "Basic problems in stability and design of switched systems," *IEEE Control Systems Magazine,* **19**(5), pp. 59-70.

Lin, Y. *et al*., 1996, "A smooth converse Lyapunov theorem for robust stability," SIAM Journal on Control and Optimization, **34**(1), pp. 124-160.

Lin, Y. and Sontag, E.D., 1991, "A universal formula for stabilization with bounded controls," Systems and Control Letters, **16(6)**, pp. 393-397.

Liu, S. and Krstić, M., 2012, *Stochastic averaging and stochastic extremum seeking*, Springer.

Lyapunov, A.M., 1892, "The general problem of stability in motion*,"* Ph.D. thesis, Univ. Kharkov.

Madani, T. and Benallegue, A., 2006, "Backstepping control for a quadcopter helicopter," *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, pp. 3255-3261.

Margaliot, M. and Langholz, G., 1998, "Adaptive fuzzy controller design via fuzzy Lyapunov synthesis," *Fuzzy System Proceedings*, IEEE World Congress on Computational Intelligence, **1**, pp. 354-359.

Margaliot, M. and Langholz, G., 1999, "Fuzzy Lyapunov-based approach to the design of fuzzy controllers," Fuzzy sets and systems, **106**(1), pp. 49-59.

Mason, M.T., 1979, "Compliance and force control for computer controlled manipulators," Ph.D. thesis, Massachusetts Institute of Technology.

Mastascusa, E.J., n.d., "An introduction to system dynamics – first order systems," from http://www.facstaff.bucknell.edu/mastascu/econtrolhtml/SysDyn/SysDyn1.html

MathWorks, 2012, "AbsTol, RelTol," from https://www.mathworks.com/matlabcentral/answers/34237-abstol-reltol

MathWorks, 2015, "Vectorization," from http://www.mathworks.com/help/matlab/matlab_prog/vectorization.html

MathWorks Support Team, 2013, "Is there a fixed-step ordinary differential equation solver in MATLAB 8.0?" MATLAB Answers, from http://www.mathworks.com/matlabcentral/answers/98293-is-there-a-fixed-step-ordinary-differential-equation-ode-solver-in-matlab-8-0-r2012b

MathWorks Support Team, 2013, "What is the maximum matrix size for each platform?" MATLAB Answers, from http://www.mathworks.com/matlabcentral/answers/91711-what-is-the-maximum-matrix-size-for-each-platform

McCourt, M.J. *et al.*, 2015, "Multiple CLFs for stabilization of nonlinear systems with input constraints," *IEEE Conference on Systems, Man, and Cybernetics*, Kowloon, China.

Minorsky, N., 1922, "Directional stability of automatically steered bodies," J. American Soc. Naval Eng., **34**(2), pp. 280-309.

Mirrahimi, M. *et al.*, 2013, "Lyapunov control of bilinear Schrödinger equations," Automatica, **41**(11), pp. 1987-1994.

MIT Office of the Registrar, n.d., "MIT subject listing & schedule, Fall 2015 search results," from http://student.mit.edu/catalog/search.cgi?search=2.003&style=verbatim

Mulone, G. and Roniero, S., 1989, "On the nonlinear stability of the rotating Bénard problem via the Lyapunov direct method," J. of Mathematical Analysis and Applications, **144**(1), pp. 109-127.

National Instruments, 2011, "PID theory explained," from http://www.ni.com/white-paper/3782/en/

Necoara, I. and Clipici, D., 2013, "Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed MPC," Journal of Process Control, **23**(3), pp. 243-253.

Open Source Robotics Foundation, 2015, "Info for 'lyap_control'", from http://wiki.ros.org/action/info/lyap_control?action=info&hitcounts=1

Open Source Robotics Foundation, 2015, "Info for 'pid'", from http://wiki.ros.org/action/info/pid?action=info&hitcounts=1

Open Source Robotics Foundation, 2014, "URDF in Gazebo," from http://gazebosim.org/tutorials/?tut=ros_urdf

Peleties, P. and DeCarlo, R., 1991, "Asymptotic stability of m-switched systems using Lyapunov-like functions," *American Control Conference*, **28**, pp. 1679-1684.

Perkins, T. and Barto, A., 2002, "Lyapunov design for safe reinforcement learning control," *Safe Learning Agents: Papers from the 2002 AAAI Symposium*, The Association for the Advancement of Artificial Intelligence, Palo Alto, CA, pp. 23-30.

Pickhardt, R., 1998, "Application of adaptive controllers to a solar power plant using a multi-model description," *American Control Conference*, IEEE, **5**, Cleveland, OH.

Pradhan, N. *et al., 2012,* "Robotics as a learning medium for engineering practice and team-based design in capstone projects," *Proceedings of the Capstone Design Conference*, Champaign-Urbana, IL.

Qin, S.J. and Badgwell, T.A., 2000, "An overview of nonlinear model predictive control applications," Nonlinear Model Predictive Control, pp. 369-392.

Rivera, D.E. *et* al., 1987, "Internal model controller 4. PID controller design," Industrial and Engineering Chemistry Process Design and Development, **25**, pp. 252-265.

ROS-Industrial, n.d., from rosindustrial.org

Ryabov, V.B., 2011, "Predicting chaos with second method of Lyapunov," *Chaos Theory: Modeling, Simulation and Applications-Selected Papers from the 3rd Chaotic Modeling and Simulation International Conference (chaos2010),* World Scientific.

Sassano, M. and Astolfi, A., 2013, "Dynamic Lyapunov functions," Automatica, **49**, pp. 1058-1067.

Schroeder, K., 2011, "On the use of force data for kinematically controlled manipulators," Ph.D. thesis, The University of Texas at Austin.

Slotine, J.J. and Li, W., 1991, *Applied Nonlinear Control*, Prentice Hall, Englewood Cliffs, New Jersey.

Stocco, L. *et al*., 1998, "Matrix normalization for optimal robot design," *Proceedings of the 1998 IEEE Conference on Robotics & Automation*, **2**, Leuven, Belgium.

Sucan, I., *et al.*, 2014, "JointStateGroup class reference," http://docs.ros.org/groovy/api/moveit_core/html/classrobot__state_1_1JointStateGroup.html

Sung, S.W. and Lee, I.B., 1996, "Limitations and countermeasures of PID controllers," Industrial and Engineering Chemistry Research, **35**(8), pp. 2596-2610.

Sutton, R.S. *et al*., 1992, "Reinforcement learning is direct adaptive optimal control," IEEE Control Systems, **12**(2), pp. 19-22.

Tan, W. and Packard, A., 2004, "Searching for control Lyapunov functions using sum of squares programming," 42nd Annual Allerton Conference on Communications, Control and Computing, Monticello, Illinois, pp. 210-219.

Tanaka, K. *et al*., 2007, "A descriptor system approach to fuzzy control system design via fuzzy Lyapunov functions," IEEE Transactions on Fuzzy Systems, **15**(3), pp. 333-341.

The Computer Language Benchmarks Game, n.d., "Which programs are fastest?" from http://benchmarksgame.alioth.debian.org/u32/which-programs-are-fastest.html

The New York Times, 1930, "Elmer Sperry Dies; Famous Inventor."

Tipsuwan, Y. and Chow, M.Y., 2003, "Control methodologies in networked systems," Control Engineering practice, **11**, pp. 1099-1111.

Tomei, P. and Perelli, C.M., 2010, "Learning control for induction motor servo drives with uncertain rotor resistance," International Journal of Control, **83**(7), pp. 1515-1528.

Topcu, U. *et al.*, 2008, "Local stability analysis using simulations and sum-of-squares programming," Automatica, **44**(10), pp. 2669-2675.

Utkin, V.I., 1992, *Sliding Modes in Control and Optimization*, Springer-Verlag, Berlin.

van der Pol, B., 1920, "A theory of the amplitude of free and forced triode vibrations," Radio Review, **1**, pp. 754-762.

van der Schaft, A.J., "Lecture 2: controllability of nonlinear systems," University of Groningen, from http://www.math.rug.nl/arjan/DownloadTeaching/DISCnonlinear2.pdf

Visioli, A., 1999, "Fuzzy logic based set-point weight tuning of PID controllers," IEEE Transactions on Systems, Man, and Cybernetics, **29**(6), pp. 587-592.

Vrabie, D. *et al*., 2007, "Continuous-time ADP for linear systems with partially unknown dynamics," *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, Honolulu, Hawaii, pp. 247-253.

Widnall, S., 2009, "Lecture L20 – Energy Methods: Lagrange's Equations," from http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec20.pdf

Willems, J., 2003, "Direct method for transient stability studies in power system analysis," IEEE Transactions on Automatic Control, **16**(4), pp. 332-341.

Wu, M. *et al.*, 2010, *Stability Analysis and Robust Control of Time-Delay Systems*, Science Press Beijing.

Yaskawa America, Inc., 2013, "Yaskawa Motoman announces new LABview and ROS-Industrial support for Motoman robots," from http://www.motoman.com/media/pr/201301-LabView-ROS.php

Yerramalla, S. *et al.*, 2003, "Lyapunov analysis of stability in an adaptive flight control system," Self Stabilizing Systems.

Yi, S.Q. *et al*., 2004, "Robust and adaptive backstepping for control of nonlinear systems using RBF neural networks," IEEE. Transactions on Neural Networks, **15**(3), pp. 693-701.

Yu, Y. and Zhong, Y.S., 2011, "Semiglobal robust backstepping output tracking for strict-feedback form systems with nonlinear uncertainty," International Journal of Automation and Systems, **9**(2), pp. 366-375.

Zelenak, A., 2016, "A lyapunov proof of stability for parallel controllers of nonlinear siso systems," *American Control Conference*, Seattle, Washington, submitted.

Zelenak, A., 2014, "Lyapunov Nonlinear Control GUI," MATLAB File Exchange, from http://www.mathworks.com/matlabcentral/fileexchange/48429-lyapunov-nonlinear-control-gui

Zelenak, A., 2015, "pid package summary," from http://www.wiki.ros.org/pid

Zelenak, A., 2015, “lyap_control package summary,” from http://www.wiki.ros.org/lyap_control

Zelenak, A., 2015, “Lyap_pend_demo,” from https://bitbucket.org/AndyZe/lyap_pend_demo

Zelenak, A. and Pryor, M., 2015, “Stabilization of nonlinear systems by switched Lyapunov function,” *ASME 2015 Dynamic Systems and Controls Conference*, Columbus, Ohio.

Zelenak, A. *et al*., 2015, “The advantages of velocity control for reactive robot motion,” *ASME 2015 Dynamic Systems and Controls Conference*, Columbus, Ohio.

Zhang, C. and Ordóñez, R., 2009, "Robust and adaptive design of numerical optimization-based extremum seeking control," Automatica **45**(3), pp. 634-646.

Ziegler, J.G. and Nichols, N.B., 1942, “Optimum settings for automatic controllers,” *trans. ASME*, **64**(11).