Stability of a Fuzzy Logic Based Piecewise Linear Hybrid System

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering

by

Aaron W. Seyfried
B.S.E.E, Wright State University, 2013

2013
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Aaron W. Seyfried ENTITLED Stability of a Fuzzy Logic Based Piecewise Linear Hybrid System BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering.

Matthew Clark, M.S.Egr.
Thesis Co-Director

Kuldip S. Rattan, Ph.D.
Thesis Co-Director

Kefu Xue, Ph.D.
Chair, Department of Electrical Engineering

Committee on Final Examination

Kuldip S. Rattan, Ph.D.

Xiaodong Zhang, Ph.D.

Matthew Clark, M.S.Egr.

R. William Ayres, Ph.D.
Interim Dean, Graduate School
ABSTRACT


Complex cyber-physical systems are difficult to model and control. However, humans are capable of accomplishing these tasks by constantly adapting and redefining the rules to control these complex systems. Fuzzy logic provides a means of encoding human inference into a control methodology. However, the fuzzy logic controllers are nonlinear and their stability is difficult to verify. Therefore, the widespread usefulness of fuzzy logic controllers is limited. It has been proven that fuzzy logic controllers can be implemented as piecewise linear switched controllers. It has also been shown that the peicewise linear system can be implemented as a hybrid system. Piecewise linear hybrid system stability can be verified by extending the Lyapunov proof for one linear system to multiple decreasing Lyapunov functions. The objective of this thesis is to implement fuzzy logic control systems as a piecewise linear hybrid system and examine their stability. A proportional fuzzy logic controller with constant derivative gain is implemented as a piecewise linear hybrid system using Matlab Simulink Stateflow. Stability of the system is examined by obtaining the Lyapunov function of each subsystem and stitching them according to the fuzzy rules. It is shown that the stitching of Lyapunov functions must successively decrease for the system to be stable. Further implications of robustness are examined by varying the fuzzy logic rules and observing the effect on the corresponding stitched Lyapunov functions.
## List of Symbols

**Chapter 1**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_A(x)$</td>
<td>Membership Function Value</td>
</tr>
<tr>
<td>$^\circ C$</td>
<td>Temperature in degrees Celsius</td>
</tr>
<tr>
<td>$\mathbb{R}^n$</td>
<td>Denotes the space, or dimension, of a vector of matrix</td>
</tr>
<tr>
<td>$\in$</td>
<td>&quot;...belonging to...&quot;</td>
</tr>
<tr>
<td>$\forall$</td>
<td>&quot;...for all...&quot;</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Eigenvalues</td>
</tr>
</tbody>
</table>
## Contents

1  Introduction .................................................. 1

2  Background .................................................... 4
   2.1 Hybrid Systems ........................................ 5
   2.2 Fuzzy Logic ............................................. 7

3  Fuzzy Logic .................................................. 9
   3.1 Fuzzy Sets ............................................... 10
   3.2 Linguistic Variables ..................................... 12
   3.3 Fuzzy Partitioning ...................................... 13
   3.4 Fuzzy Rules ............................................. 15
   3.5 Fuzzy Control .......................................... 15
   3.6 Control Surface ........................................ 20
   3.7 FLC as a Piecewise Linear Controller ............... 21

4  Hybrid Systems .............................................. 23
   4.1 Dynamical System Classification ....................... 23
   4.2 The Thermostat: A Hybrid System .................... 24
   4.3 Hybrid Automata ....................................... 26

5  Stability of Hybrid Systems ............................... 28
   5.1 Linear Matrix Inequalities (LMI’s) .................... 28
   5.2 Lyapunov Stability: Lyapunov’s Direct Method ....... 30
   5.3 Lyapunov Equilibria ................................... 31
   5.4 Lyapunov Stability: Control Applications .......... 32
   5.5 Lyapunov Stability: An Example ..................... 34
   5.6 Hybrid System Lyapunov Functions .................. 37

6  Stateflow Hybrid Systems ................................. 39
   6.1 Introduction to Stateflow .............................. 39
   6.2 Stateflow Control Systems ............................ 40
   6.3 Building an Equivalent Stateflow Controller ....... 41
   6.4 FLC to Hybrid System Conversion ................... 44
## List of Figures

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example Decreasing Stitched Lyapunov Function</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>Classical and Fuzzy Sets</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>Union, Intersection and Complement of Two Overlapping Fuzzy Sets</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>Smooth Transitions between Three Fuzzy Sets</td>
<td>14</td>
</tr>
<tr>
<td>3.4</td>
<td>Dual Membership</td>
<td>14</td>
</tr>
<tr>
<td>3.5</td>
<td>Components of a Fuzzy Logic Control System</td>
<td>16</td>
</tr>
<tr>
<td>3.6</td>
<td>Minimum-Maximum Method of Inference to Produce Output Fuzzy Sets [8]</td>
<td>17</td>
</tr>
<tr>
<td>3.7</td>
<td>Input Fuzzy Sets with two non-zero values</td>
<td>18</td>
</tr>
<tr>
<td>3.8</td>
<td>Example Control Surface</td>
<td>20</td>
</tr>
<tr>
<td>3.9</td>
<td>Baseline Controller and Fuzzy Control Surface</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Example Thermostat Chart</td>
<td>25</td>
</tr>
<tr>
<td>4.2</td>
<td>Example Thermostat Plot</td>
<td>26</td>
</tr>
<tr>
<td>5.1</td>
<td>Graphical Representation of Equilibrium Conditions</td>
<td>31</td>
</tr>
<tr>
<td>5.2</td>
<td>Stable State Trajectory</td>
<td>36</td>
</tr>
<tr>
<td>5.3</td>
<td>Example Decreasing Stitched Lyapunov Function</td>
<td>37</td>
</tr>
<tr>
<td>5.4</td>
<td>Decreasing Stitched Lyapunov Function</td>
<td>38</td>
</tr>
<tr>
<td>6.1</td>
<td>Stateflow Chart Example</td>
<td>40</td>
</tr>
<tr>
<td>6.2</td>
<td>Stateflow Flow Controller Example</td>
<td>41</td>
</tr>
<tr>
<td>6.3</td>
<td>Empty Stateflow Chart</td>
<td>42</td>
</tr>
<tr>
<td>6.4</td>
<td>Fuzzy Control Surface</td>
<td>42</td>
</tr>
<tr>
<td>6.5</td>
<td>Stateflow Chart with Fuzzy Dynamics</td>
<td>43</td>
</tr>
<tr>
<td>6.6</td>
<td>Simulink Implementation of Stateflow Fuzzy Controller</td>
<td>44</td>
</tr>
<tr>
<td>6.7</td>
<td>Simulink Implementation of Stateflow and Fuzzy Logic Controllers</td>
<td>45</td>
</tr>
<tr>
<td>6.8</td>
<td>Step Responses of Stateflow and Fuzzy Logic Controller</td>
<td>45</td>
</tr>
<tr>
<td>6.9</td>
<td>Simulink Implementation of PID Controller</td>
<td>46</td>
</tr>
<tr>
<td>6.10</td>
<td>Stateflow Chart of a Hybrid PID Fuzzy Controller</td>
<td>46</td>
</tr>
<tr>
<td>6.11</td>
<td>Step Response of the Hybrid PID Controlled System</td>
<td>47</td>
</tr>
<tr>
<td>7.1</td>
<td>Hybrid System Simulation with Initial Conditions Only</td>
<td>49</td>
</tr>
<tr>
<td>7.2</td>
<td>Control Surface of the Fuzzy Controller</td>
<td>49</td>
</tr>
</tbody>
</table>
7.3 Lyapunov Function of $A$ as a Function of Time. 
7.4 Stitched Lyapunov Functions of the System. 
7.5 Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PS$ is Varied. 
7.6 Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PM$ is Varied. 
7.7 Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PS$ and $PM$ are Varied. 
7.8 Second Order Plant Root Locus. 
7.9 Third Order Plant Root Locus. 
7.10 Simulink Model Used to Generate the State Vector for the Third Order System. 
7.11 Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PS$ is Varied. 
7.12 Error due to Initial Conditions. 
7.13 Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PM$ is Varied. 
7.14 Response of the System to Initial Conditions. 
7.15 Varying Both PM and PS for the Third Order Plant. 
7.16 Error Response due to Initial Conditions.
Acknowledgment

I would like to extend much thanks to Mr. Matthew Clark for providing his insight and guidance throughout the course of this research. I would also like to extend my thanks to Dr. Kuldip Rattan for his help with this project and for his steadfast commitment to quality instruction during my tenure as a student. Without the help from these two individuals, this project would not have been realized.
Dedicated to

My Mother, Nancy
Introduction

Complex cyber-physical systems are difficult to model and control. As the number of components and operating modes increase, so too does the complexity of the model. These complex models yield control techniques that are equally difficult. The tasks that these complex systems accomplish can often be carried out by humans with very little thought. Humans are capable of highly complex control by constantly adapting and redefining rules to control complex systems. Fuzzy logic provides a means of encoding human inference and rules into a control methodology. The rule base capabilities of fuzzy logic provide a method to control complex systems in a similar manner as a human. Fuzzy logic controllers would be more readily accepted for control applications if their stability can be verified. Therefore, the widespread usefulness of fuzzy logic controllers is limited to only those systems who can be verified stable.

It has been proven that fuzzy logic controllers can be implemented as piecewise linear switched controllers [1]. It has also been shown that the piecewise linear systems can be implemented as a hybrid system [2]. The segments of the piecewise linear system form the modes of the hybrid system. These modes are continuous with discrete state transitions between modes called edges. The stability verification tools available for hybrid systems,
therefore, can be applied to the piecewise linear hybrid fuzzy logic system. Piecewise linear hybrid system stability can be verified by extending the Lyapunov proof for one linear system to multiple decreasing Lyapunov functions [3]. The objective of this thesis is to implement a fuzzy logic control system as a piecewise linear hybrid system and examine its stability.

A proportional fuzzy logic controller with constant derivative gain is implemented as a piecewise linear hybrid system using Matlab Simulink Stateflow. With the proportional fuzzy logic controller implemented as a piecewise linear hybrid system, the stability can be examined using Lyapunov functions [3]. The Lyapunov function of each subsystem is obtained and the segments are stitched according to the fuzzy rules. It is shown that the stitched Lyapunov function must successively decrease for the controlled system to be stable about the equilibrium point. Lyapunov functions are a representation of the energy in a system over time. Therefore, the Lyapunov functions show the system’s energy is tending towards the stable equilibrium. The effects of the fuzzy controller on the stability of the system are further examined in this thesis.

Since fuzzy logic controllers are designed to control a wide range of plant models, the dynamics of the controlled system are different depending on the application and desired control metrics. The control surface is a graphical representation of the input/output rules of the fuzzy control. They also illustrate the piecewise linear nature of fuzzy logic controllers. The control surface varies based on the fuzzy logic rules. Implications of robustness are examined in this thesis by varying these rules and studying the associated stitched Lyapunov functions. Since the Lyapunov functions are stitched according to the rules, these variations in the control surface produce unique stitched Lyapunov functions. The requirement that the stitched Lyapunov functions be decreasing still implies stability. Observing the effect of the control surface on the stitched Lyapunov functions provides
insight into the stability of fuzzy logic controllers and potential for further classification of stable control regions.

This thesis presents a small portion of the research available in fuzzy logic and hybrid systems in the background section. More information about fuzzy logic and hybrid systems is presented in the next two chapters. Then, the stability of dynamical systems is considered in the following chapter. The implementation of the hybrid system as a Stateflow controller is shown in Chapter 6. Finally, the simulations conducted in this thesis are shown in Chapter 7 and relevant conclusions are shown in Chapter 8.
Fuzzy control systems are nonlinear and thus there is no closed form solution that may be applied to test stability of these systems. In general, nonlinear systems, like fuzzy control systems, are difficult to verify. To overcome this difficulty, the systems are linearized about a certain operating point, thus, making the verification of the nonlinear system valid as the verification of a piecewise linear system. The expense of linearization increases with the complexity of the system. However, even with very complex piecewise linear systems, Monte Carlo simulations can be used to verify stability.

Monte Carlo simulations are used to verify certain systems when a closed-form solution is not readily available. A system may be verified as stable about a given equilibrium by linearization: the process of treating nonlinear models as linear over a small range. During Monte Carlo simulations, systems are subjected to various inputs and disturbances and the response of the linearized system is obtained. Gain scheduling can be used to specify different linear gains in certain linear operating modes. For example, an actuator on a flight control surface in an airplane may have different qualities dependent on the operating mode of the aircraft. In a "level flight" mode, the nonlinear plant model is linearized about a point where the roll, pitch, and yaw are set to zero. The controller is then designed for
this linearized plant model. However, in the "approach to landing" mode, the plant model is linearized about a new point where the roll and yaw are set to zero but the pitch is set at negative 20 degrees. This produces two different linearized plant models to control. A Monte Carlo simulation for this case would produce random switches between these two modes.

Even with Monte Carlo simulations as a verification tool, the simulation does not account for every possible outcome. The number of simulations and the statistics behind a random sample can show that the test accounts for a percentage of possible outcomes; but not all. To gain more insight into the system stability and cover a larger possible set of outcomes, more simulations must be run. This takes time and money. Thus, there is need for a closed form solution for the verification of such systems.

2.1 Hybrid Systems

Hybrid systems are those that contain both continuous and discrete variables. The modes of a system are the independent, continuous subsystems while the transitions are the discrete changes in operating mode. These transitions follow the predetermined paths between the states called edges. All of these subsystems combine to make up the larger hybrid system.

The stability of hybrid systems has been studied in recent years. The most common stability verification technique for hybrid systems is the use of Lyapunov stability methods. In the Lyapunov stability method, a Lyapunov function is developed to minimize the energy in a given system. For the classical, single system cases, this does not present a
problem, since a Lyapunov function for a classical system will always exist. However, for hybrid systems, with multiple subsystems, the Lyapunov approach becomes more difficult.

In hybrid systems, the switching between the modes plays an important role in stability. Two stable subsystems may be switched in a way that the result is unstable. Likewise, two unstable systems may be switched in a way that results in a stable system. If all modes are stable in a hybrid system, and there exists a common Lyapunov function, then any switching is stable. However, the existence of a common Lyapunov function is rare. Instead, many researchers chose to examine stitched combinations of individual subsystem Lyapunov functions. In this case, the condition for stability is that the stitched Lyapunov function should decrease with time. Also, for every mode transition, the energy in the system must be less than the maximum energy of the previous mode [3]. Figure 2.1 shows this condition graphically.

![Example Decreasing Stitched Lyapunov Function](image)

Figure 2.1: Example Decreasing Stitched Lyapunov Function.
Since much work has been carried out in the area of piecewise linear hybrid systems, this thesis models the dynamics of a fuzzy controller as such. The stability of a fuzzy logic control system is evaluated utilizing the available techniques of Lyapunov stability verification for hybrid systems.

2.2 Fuzzy Logic

Since fuzzy logic was first proposed in 1965 by Lotfi Zadeh, [4], there has been much criticism over its usefulness in engineering and technology. As recent as 2008, its usefulness has been debated. Some researchers take the literal meaning of the word "fuzzy" as a drawback to stability verification. The main consideration made recently by Zadeh is that "fuzzy" is a bit of a misnomer in reference to the decision algorithm and the linguistic variables that fuzzy logic describes. Zadeh claims: "Fuzzy logic is not fuzzy. Basically, fuzzy logic is a precise logic of imprecision and approximate reasoning" [5]. Fuzzy logic’s most practical application is stated in this claim: the ability to make precise inferences on imprecise information and linguistic variables.

On the subject of fuzzy logic stability, there have been studies on very specific cases of fuzzy logic system stability. In his 20120 paper Mohammad Biglarbegian discusses the stability of a Type-2 fuzzy logic system and proposes a novel inference mechanism [6]. Type-2 fuzzy logic systems are those that have parabolic or other nonlinear membership functions. The system stability in this work is determined by operations on the state-space representation of the system using Lyapunov's stability theorem. Tuning algorithms for the controller are proposed as well as analyses of the controlled system stability. These algo-
rithms show promise in the area of stabilizability as unstable plant models are shown to be controllable under certain circumstances; though the paper claims that there is more research to be done on stability analysis for the Type-2 fuzzy logic system [6]. This research is promising in the area of fuzzy logic stability but differs from this thesis as the fuzzy system is of type-2. Unlike the type-2 fuzzy systems, the type-1 systems studied in this thesis have linear membership functions.

In another study, the stability of nonlinear Tagaki-Sugeno (TS) fuzzy models, an application of a fuzzy type-2 model, are assessed using non-quadratic Lyapunov functions. In this research, the authors seek to relax the stability requirements of nonlinear TS models by showing stable ranges of values for the plant model in question. The results show still a decreasing Lyapunov function and the subsequent stability of the systems in question. This research is relevant to this thesis in that similar Lyapunov functions of fuzzy systems are examined as they evolve over time. However, the Lyapunov functions used in this thesis are quadratic. Also, the non-deterministic plant models in this paper do not apply to the research in this thesis. This thesis will focus on deterministic (known) plant models and the stability of type-1 fuzzy models; not those of TS fuzzy models [7].

This chapter gives the background and current research available for both hybrid systems and fuzzy logic. The previous research in this thesis, although similar, bears some differences to that presented in this thesis. Namely, the use of hybrid systems to model the dynamics of a fuzzy logic controller is a new consideration. In the following chapter, the fundamentals of type-1 fuzzy logic systems are described.
Fuzzy Logic

Fuzzy logic is a means of implementing fuzzy, soft phrases into crisp, quantitative values via a mathematical relationship. The coding of human intuition into a framework that can be implemented on a computer is accomplished through fuzzy logic. Thus, fuzzy logic can be used to implement linguistic variables using computers. Humans make thousands of judgment based decisions every day. In the everyday act of pouring a cup of coffee, a human can quickly determine the flow rate of coffee from the pot to the cup. During the initial fill of the cup, the flow rate is high in order to minimize time spent filling the cup. As the cup fills, the flow rate is slowed to avoid spilling or overfilling the cup. These types of judgment decisions are practiced repeatedly by humans. Implementing these judgement based decisions in a control methodology can provide useful results.

The term fuzzy logic was first documented by Lotfi Zadeh, a Professor at the University of California at Berkeley, in 1965 in his paper Fuzzy Sets: Information and Control [4]. Zadeh claimed that people are capable of highly adaptive control and do not require exact, numerical input. Hence, humans utilize fuzzy values to make decisions. However, the term fuzzy does not refer to something that cannot be defined, but rather it is a technical adjective to describe systems that may not have been previously defined [8]. Fuzzy acts as
a language to translate the abstract, imprecise world into the precise world of mathematical logic [9]. Therefore, the linguistic and qualitative aspects of the world may be encoded using fuzzy logic. The imprecise tasks performed by humans can be implemented in an exact manner using fuzzy logic.

3.1 Fuzzy Sets

In classical set theory, a crisp set is used to separate a space from the universe of discourse. In the coffee pouring example, a classical set would only be able to classify a 0% and a 100% flow rate. This method is not useful in control. However a method of defining smooth transitions between the sets is achieved using fuzzy logic. Consider the fuzzy set shown in Figure 3.1. This fuzzy set is defined by the rule method. This methods plots the relationship of maximum flow rate to input flow rate. For example, for any input on the interval [50, 75] a variable flow rate is desired, but with a classical set, the output is the same. In the fuzzy set used, it can be seen that there is a smooth transition from a membership of zero to one and back to zero. This differs from the classical set shown in the same figure. If the fuzzy set were used to control the flow rate during pouring a cup of coffee, it would produce more desirable results than if the classical fuzzy set were to be used.
A fuzzy set in the universe of discourse $U$ is identified by its membership function $\mu_A(x)$ which lies on the interval $[-1, 1]$. Thus, fuzzy sets can take on unions, intersections, and complements. The union is defined as the set that includes the maximum of either set $A$ or set $B$. The intersection is defined as the set that includes the minimum of either set $A$ or $B$. Finally, the complement is defined as $1 - \mu_A(x)$. Figure 3.2 shows the union, intersection and complement of the fuzzy sets $A$ and $B$. 

Figure 3.1: Classical and Fuzzy Sets.

Figure 3.2: Union, Intersection and Complement of Two Overlapping Fuzzy Sets.
3.2 Linguistic Variables

As displayed in Figure 3.1, fuzzy logic allows for identification of real world variables without harsh transitions. In common vernacular, words are often used to describe such variables. If a car is traveling at a high speed, one might say that the cars speed is fast. The linguistic variable is the "car’s speed” while the word ”fast” describes the variable. The variable ”car’s speed” can also take on crisp values, like 50 or 70 miles per hour. Fuzzy logic allows linguistic terms such as slow, moderate, or fast to be associated directly with mathematical (crisp) values with the entire range of speed values from 0 to 100 mph. ”If a variable can take words in natural languages as its values, it is called a linguistic variable, where the words are characterized by fuzzy sets defined in the universe of discourse in which the variable exists” [10].

Defining a set in $U$ that contains only vehicles with 6-cylinders produces a set that either has 6 cylinders or does not. There is no part of that set that can be construed as only partially having 6 cylinders. However, choosing to define a set in $U$ that contains only U.S. made cars presents a problem. Many individuals perceive that a car is a U.S. car if it is a brand manufactured by a U.S. owned auto maker. Otherwise, it would be classified as a non-U.S. car. Other individuals claim that to be a U.S. car, it must be manufactured of only parts made in the United States. Since many car manufacturers utilize other countries as suppliers, the classification of a U.S. car is not a crisp distinction. Under fuzzy set theory, a car can be assigned a linguistic variable based on how much of the car is actually U.S. manufactured. For example:
If $\mu_A(x) = \begin{cases} 
1 & \text{then the car is completely U.S. made} \\
0.8 & \text{then the car is mostly U.S. made} \\
0.6 & \text{then the car is somewhat U.S. made} \\
0.3 & \text{then the car is minimally U.S. made} \\
0 & \text{then the car is not U.S. made} 
\end{cases}$

In this case, the linguistic variables completely, mostly, somewhat, minimally, and not are all used to describe the variable "U.S. made". The linguistic variable is chosen based on the membership function value. [10]

### 3.3 Fuzzy Partitioning

Fuzzy partitioning provides a means of smooth transition between membership subspaces. As opposed to classical set theory in which membership between two sets has an abrupt change in classification. For example, as a car accelerates, there is not a set value at which one would consider the speed to change from slow to fast. Rather, as the speed increases, it begins by belonging solely to the slow membership function. After some time, onlookers might say that the car is traveling at a medium speed. Thus, the membership to the slow classification has decreased while the membership to medium classification has increased. Such is the case from the medium speed classification to the fast speed classification [10]. The illustration in Figure 3.3 shows a possible situation in which membership values are smooth throughout the transition in all states.
For a certain crisp value of input, the membership may belong to two classifications. As the value of the input variable, speed, increases from 1 to 2, it can be seen in Figure 3.3 that the membership function value transitions from belonging to the slow classification to the medium classification without an abrupt change. This an input can belong to two classifications at one time. Figure 3.4 illustrates this capacity of fuzzy sets more clearly.
3.4 Fuzzy Rules

Fuzzy rules, or a fuzzy rule base, define the relationship between two universes of discourse. For example, if during exercise a person’s body temperature is high and heart rate is fast, he should rest for a long time. If the body temperature is average and heart rate is normal, he should not rest. The two linguistic input variables (body temperature and heart rate) are used to define the output variable (duration of rest). The input variables belong to two separate universes of discourse and produce an additional linguistic variable that identify the universe of rest. Fuzzy rule bases are typically used to map the input variables to the output variables by utilizing if-then rules.

3.5 Fuzzy Control

Fuzzy controllers have four essential parts: a fuzzification process, a fuzzy rule base, an inference mechanism, and a defuzzification process. The fuzzification process transforms a crisp value input to the controller into a fuzzy value on the interval \([-1, 1]\). After fuzzification, the inference mechanism examines the fuzzy value against a fuzzy rule base. This rule base contains the embedded rules for operation. After the inference mechanism has applied the rules, the fuzzy value is defuzzified, scaled, and fed back into the system. [10] Figure 3.5 shows graphically, the operation of the fuzzy controller.
Figure 3.5: Components of a Fuzzy Logic Control System.

There are multiple means of fuzzification and inference in the field of fuzzy logic. Almost all forms of fuzzification, inference, and defuzzification function on a variation of a weighted average approach. In this approach, the membership function values are treated as the weights and are utilized to determine the appropriate output during the defuzzification process. This is a mapping of the output response back into crisp output values [10]. For the example used in Figure 3.4, the input value is identified in two fuzzy membership functions. The fuzzy membership value is 0.4 in the slow classification and 0.6 in the medium classification. The crisp output obtained by the weighted average technique is given by:

\[ u_1(t) = \frac{\sum U \mu_A(x)x}{\sum U \mu_A(x)} \]  

(3.2)

\[ = \frac{(0.4)1.6 + (0.6)1.6}{0.4 + 0.6} = 1.6 \]  

(3.3)

This is a trivial example because it assumes that the input maps directly to the out-
put. If a rule base is introduced, however, this relationship may not be one to one. If the following rules are defined:

1. If the speed of the car is slow then reduce acceleration slightly
2. If the speed of the car is medium then reduce acceleration moderately
3. If the speed of the car is fast then reduce acceleration greatly

Figure 3.6 shows how the inference mechanism translates the linguistic rules from the rule base to the output [8].

Figure 3.6: Minimum-Maximum Method of Inference to Produce Output Fuzzy Sets [8].
For the purposes of this thesis, the fuzzification and inference processes is examined using the matrix approach. The matrix approach begins with a traditional fuzzification process to determine the input vector. The input vector is generated by comparing the input to the membership functions. As an example, consider the fuzzy sets for the input shown in Figure 3.7.

Figure 3.7: Input Fuzzy Sets with two non-zero values.

For the input shown on Figure 3.7, it can be seen that there are only two fuzzy sets with non-zero membership function values. These two sets, positive medium (PM) and positive small (PS) are the only two active fuzzy sets; all others are zero. The membership of the input value in fuzzy sets is given by:

\[
I = \mu_{NB}(x) \quad \mu_{NM}(x) \quad \mu_{NS}(x) \quad \mu_{ZR}(x) \quad \mu_{PS}(x) \quad \mu_{PM}(x) \quad \mu_{PB}(x)
\]

(3.4)
Next, the membership of the input vector is multiplied by the rule matrix. The rule matrix is constructed utilizing the chosen peak points for the output membership function triangles. As an example, assume the rule matrix is of the form:

\[
\text{Rule Matrix, } R = \begin{bmatrix}
-1 & -0.8 & -0.25 & 0 & 0.25 & 0.8 & 1
\end{bmatrix}^T
\]  

(3.6)

With the input vector and rule base defined, the two matrices are multiplied to obtain the value of the output.

\[
\begin{align*}
\text{Output} &= IR = \begin{bmatrix}
0 & 0 & 0 & 0.6 & 0.4 & 0
\end{bmatrix}
\begin{bmatrix}
-1 \\
-0.8 \\
-0.25 \\
0 \\
0.25 \\
0.8 \\
1
\end{bmatrix} = 0.47 \\
\end{align*}
\]  

(3.7)

This rule vector implementation is valid only for a proportional FLC. In the case of a proportional plus derivative FLC, there are fuzzy sets for the error and the derivative of error. Given that there are two inputs, the rule matrix must now be defined as a matrix in order to handle comparisons of two fuzzy inputs.

\[
\text{Output} = I_\epsilon R I_\dot{\epsilon}
\]  

(3.8)
where $I_e$ is a row vector containing the membership function values from the error fuzzy set, $R$ is the defined rule matrix, and $I_e$ is a column vector containing the membership function values from the derivative of error fuzzy set. In this case, the rows of the rule matrix represent the derivative of error and the columns represent the error. The rule base and input fuzzy sets are selected in such a way as to produce desired output.

### 3.6 Control Surface

The control surface is a plot of the effective gain in a certain region of operation. Figure 3.8 shows three different lines pieced together to produce the control surface. The slopes of the lines determine the gain to be utilized in that region. For the control surface shown, it can be seen that when the input is between 0 and 0.08, the PS region, the gain is 3.75. Similarly, when the input is between 0.08 and 0.5, the PM region, the gain is 0.95. Finally, when the input is between 0.5 and 1, the PB region, the gain is 0.6.

![Example Control Surface](image)

**Figure 3.8: Example Control Surface.**
3.7 FLC as a Piecewise Linear Controller

As mentioned previously, the gain in a particular region is determined by the slope of the control surface in that region. There is another term that determines the output of the controller. First, consider a “baseline” controller where the input and output fuzzy sets are equally spaced. The resulting control surface is shown in Figure 3.9. For the ”baseline” controller, the fuzzy controller becomes a conventional proportional controller as the gain is constant in every region; thus the slope of the surface is actually the gain in that region. However, when the input and output fuzzy sets are varied so they are not equally spaced, the control surface produces different gains in each region. The effective gains are now not just the slopes of the lines but include the offsets of each region as well.

Figure 3.9: Baseline Controller and Fuzzy Control Surface.

Figure 3.9 shows the baseline controller and the fuzzy control surface given in Figure 3.8. It can be seen from this figure that the baseline controller will perform in a similar manner to a conventional proportional controller. However, the fuzzy controller exhibits different characteristics. The effective gains (slopes) are used in conjunction with the offset value to produce the output of the fuzzy controller. For the section labeled PS (positive
small), there is no offset as this section crosses the origin. The section labeled $PM$ (positive medium), however, has an offset as that line does not cross the origin. The offset has a value of 0.2238. Similarly, for the $PB$ (positive big) state, the offset is 0.4.

In summary fuzzy logic, is a means of translating linguistic variables into mathematical relationships. This chapter describes the basics of fuzzy logic and explores some of its attributes to control. This thesis utilizes the functionality of fuzzy logic to design hybrid systems. Once the hybrid system has been designed, the dynamics are studied in an effort to draw conclusions about overall system stability.
Hybrid Systems

4.1 Dynamical System Classification

A dynamical system is one that describes the evolution of a state over time. Dynamical systems may be influenced by uncontrollable disturbances or by control inputs. Effects of wind shear on an aircraft in flight (uncontrollable) and the pilot’s input to the flight surfaces (control inputs) are two examples of these disturbances. Dynamical systems often have outputs that are utilized to enhance, or alter, the dynamics of a system. These systems and their outputs are used to control the actions of a larger system. They are commonly referred to as control systems [2].

In general, there are three types of dynamical systems: continuous, discrete, and hybrid. The state of a continuous dynamical system changes with smooth non-abrupt movements. Such systems have continuous inputs and outputs that are continuously differentiable over all states. The inputs and outputs of discrete dynamical systems are only known at the sampling instants and thus are not continuously differentiable over all states. Finally,
hybrid systems are those that include continuous and discrete state variables. They can range from systems with mostly continuous dynamics and a few discrete state transitions to systems that contain many discrete state transitions and have little continuous action [2].

Dynamical systems can be classified further as linear or nonlinear and time varying or time-invariant. Linear systems are those systems in which the underlying differential equation (continuous) or difference equation (discrete) is linear. Nonlinear systems are systems in which the underlying differential equation or difference equation is nonlinear. Time varying systems are those whose output will be different depending on when the system is being operated. This is better described as a system whose dynamics are different on different days of simulation. Time invariant systems are systems whose dynamics do not change over periods of time greater than that of simulation. This thesis will discuss systems that are continuous, discrete, and hybrid, linear and nonlinear. All dynamical systems studied are time invariant.

4.2 The Thermostat: A Hybrid System

Consider a room being heated by a furnace. The continuous nature of the system lies inside the states themselves while the discrete nature of the system stems from the two distinct modes: "on" or "off". The furnace is controlled by a thermostat and is given a target temperature and a tolerance. For this example, assume the target temperature is 20°C and the tolerance is ±1°C. The temperature in the room is governed by the differential equation:

\[ \dot{x} = ax; \ a < 0 \]  

(4.1)
This differential equation is stable and thus the temperature will decay to zero if the heater is not engaged. When the heater is engaged the temperature increases exponentially toward 30 °C according to the following differential equation:

\[ \dot{x} = a(x - 30); \quad a < 0 \quad (4.2) \]

The system can be modeled as a hybrid system with continuous dynamics in two discrete states. The discrete states correspond to the heater being ON or OFF while the temperature is governed by the continuous differential equation in each state. Figure 4.1 shows a possible model of the hybrid system. Figure 4.2 shows a possible plot of room temperature versus time when controlled by the hybrid system shown in Figure 4.1.

Since the thermostat is trying to maintain a temperature of 20(±1) °C the temperature will fluctuate between 21 and 19 degrees. The temperature change follows the continuous differential equations shown inside the state blocks. When the state is ON, the temperature will increase exponentially according to the differential equation in the ON block. When the state is OFF, the temperature will decrease exponentially according to the differential equation in the OFF block.
Modeling the thermostat as a hybrid system lets the user examine the system as a whole and not just the individual states. It can be seen that the differential equations shown in equation (4.2) and equation (4.1) are stable when examined individually. However, the discrete state transitions add another element to the system that can cause instability. If the system is modeled as a hybrid system, the tools available for stability verification can be applied to the thermostat.

4.3 Hybrid Automata

A hybrid automaton is used to describe how a hybrid system evolves over time. To be viable, hybrid automata must have certain characteristics of clarity and abstractability. The automata must clearly define the process and dynamics of the hybrid system’s continuous and discrete dynamics. They must be abstractable to allow for the redesign of certain in-
individual components as well as system-wide alterations. The automata must clearly define the system but still allow for changes in the model. Hybrid automata can be described by the following equation:

\[ H = (Q, X, f, Init, D, E, G, R) \] (4.3)

where \( Q = q_1, q_2, \ldots q_n \) is a set of discrete states, \( X = \mathbb{R}^n \) is a set of continuous states, \( f(\cdot, \cdot)Q \times X \rightarrow \mathbb{R} \) is a vector field, \( Init \subseteq Q \times X \) is a set of initial states, \( D(\cdot) : Q \rightarrow P(X) \) is a domain, \( E \subseteq Q \times Q \) is a set of edges (paths), \( G(\cdot) : E \rightarrow P(X) \) is a set of guard condition, and \( R(\cdot, \cdot)E \times X \rightarrow P(X) \) is a reset map.

This thesis will focus on graphical representations of hybrid systems. This graphical method (Figure 4.1) explicitly shows the components of the hybrid automata in an intuitive manner. The discrete and continuous states, edges, and guard conditions are displayed to show the functionality of the system. Using this graphical method, hybrid systems can be studied much easier [2].

This chapter covers the introduction to hybrid systems as well as some functionality of hybrid systems. The classification of dynamical systems has been described with its correspondence to hybrid systems. Also provided in this chapter, is an example of a hybrid system in the form of a thermostat. Finally, this chapter introduced the idea of hybrid automata in their relation to hybrid systems. Fuzzy controllers are implemented in this thesis as piecewise linear hybrid systems.
Stability of Hybrid Systems

Stability is paramount in the study of dynamical systems. There are multiple methods of determining classical system stability such as the Routh-Hurwitz or Jury Stability tests. These two tests for stability are valid for a single-input single-output linear time-invariant systems. However, these methods are not viable in the study of hybrid systems. The non-linearity in switching makes traditional stability tests invalid. Lyapunov stability criterion is selected for this research due to the nonlinear nature of fuzzy logic controllers. The Lyapunov stability criterion deal with equilibrium points and regions of attraction of linear and nonlinear systems. Since the systems are represented, in this thesis as state-space systems, the matrix method of Lyapunov stability will be utilized.

5.1 Linear Matrix Inequalities (LMI’s)

Before studying Lyapunov’s stability analysis, it is beneficial to specify the properties of linear matrix inequalities. Linear matrix inequalities are equations of inequality that contain linear algebra terms. Pertinent terms include matrices that are said to be positive definite,
positive semi-definite, negative semi-definite, and negative definite. Suppose a matrix, $M$, is defined as:

$$
M = \begin{bmatrix}
m_{11} & m_{12} \\
m_{21} & m_{22}
\end{bmatrix}
$$

(5.1)

The matrix is said to be:

- **Positive Definite** iff $Re\{\lambda\} > 0; \ \forall \lambda_i \in \mathbb{R}^n$
- **Positive Semi-Definite** iff $Re\{\lambda\} \geq 0; \ \forall \lambda_i \in \mathbb{R}^n$
- **Negative Semi-Definite** iff $Re\{\lambda\} \leq 0; \ \forall \lambda_i \in \mathbb{R}^n$
- **Negative Definite** iff $Re\{\lambda\} < 0; \ \forall \lambda_i \in \mathbb{R}^n$

where $\lambda_i$ are the eigenvalues of $M$. Eigenvalues are determined by finding the roots of the equation from the determinant of the matrix

$$
\lambda_{i,n} = |\lambda I_n - M|
$$

(5.2)

where $M$ belongs to the space $\mathbb{R}^n$ and $I$ is an identity matrix of similar dimension.

The Lyapunov stability theorem utilizes LMIs to solve for various matrices that are classified as positive definite, positive semi-definite, negative semi-definite, and negative definite [11] [12], [13].
5.2 Lyapunov Stability: Lyapunov’s Direct Method

Consider the system described by:

\[ \dot{x} = f(t, x) \] (general)

\[ \dot{x} = f(x) \] (autonomous)

Lyapunov stability is divided into two methods. The first method deals with the stability of equilibria of differential equations, both linear and non-linear. This method focuses on the differential equation, the solution to which is known. This method is limited in that the solution to the differential equation is not always easily obtained, and may not exist. The second Lyapunov method, also known as Lyapunov’s Direct Method, utilizes a function defined by the user, \( V(x) \). The objective of the second method is to answer questions of stability using \( f(t, x) \) or \( f(x) \) rather than explicit knowledge of the system [14]. This function is chosen by experience and is used instead of evaluating the solution to the differential equation.

In using the Lyapunov’s direct method, there are three conditions that must hold for a system to be stable. If the following conditions hold true, the system is said to be stable in the sense of Lyapunov.

Let \( x = 0 \) be an equilibrium point and \( D \subset \mathbb{R}^n \) is a domain containing \( x = 0 \) and \( V : D \to \mathbb{R} \) be a continuously differentiable function such that

1. \( V(0) = 0 \)

2. \( V(x) > 0; \forall x \in D - 0 \)

3. \( \dot{V}(x) \leq 0; \forall x \in D \)

Then \( x = 0 \) is a stable equilibrium. Moreover, if \( \dot{V}(x) < 0; \forall x \in D - 0 \) then \( x = 0 \) is asymptotically stable; \( x(t) \to 0 \) as \( t \to \infty \).
5.3 Lyapunov Equilibria

A.M. Lyapunov’s work was one of the first to consider the stability of non-linear systems that had been linearized about an equilibrium point [15]. Let us define that the equilibrium state $x_e$ as stable in the sense of Lyapunov if given $t_0$ and $\epsilon > 0$, there exists $\delta(\epsilon, t_0)$ such that:

$$|x_0 - x_e| < \delta \Rightarrow |x(t, t_0, x_0)| < \epsilon; \quad \forall t \geq t_0$$

(5.3)

Figure 5.1 shows the same requirement graphically. At the root of this condition for stability is the requirement that for minor perturbations of the system about the equilibrium point, less than $\delta$, the system will return to the equilibrium state. More importantly is the property that the trajectories may grow larger than the value of $\delta$. It is stated that the trajectories of the system do not exceed another radius, $\epsilon$, about the equilibrium that is larger than the perturbation, $\delta$. Thus the equilibrium is said to be stable...
in the sense of Lyapunov because for all time, \( t > t_0 \), the trajectories of the system are maintained inside the finite value of \( \epsilon > \delta \) [14].

### 5.4 Lyapunov Stability: Control Applications

Lyapunov stability has many useful applications. It can be used to verify the convergence or boundedness of linear and nonlinear differential equations. With Lyapunov stability being a broad topic, this thesis will focus on the applications of the Lyapunov stability theorem to piecewise linear time-invariant systems. Lyapunov stability is determined by operations on the \( A \) matrix of the associated system. The input matrix, \( B \), is neglected because, for internal stability purposes, the input is set to zero. Consider a state-space system with input matrix, \( B \), set to zero:

\[
\dot{x} = Ax
\]  

(5.4)

Lyapunov’s stability theorem states that the system is asymptotically stable, uniformly asymptotically stable and exponentially stable in the large if and only if any one of the following conditions are satisfied:

- \( \Re \lambda(A) < 0 \)
- For any given symmetric, positive-definite matrix \( (Q = Q^T > 0) \), there exists a unique \( P = P^T > 0 \) such that \( A^T P + PA \leq -Q \)
- For any given matrix \( C \) with \((C, A)\) observable, the equation \( A^T P + PA \leq -C^T C \) has a unique solution \( P = P^T > 0 \)

The important consideration for the purposes of this thesis is the condition where
\[ A^T P + PA + Q \leq 0 \]  \hspace{1cm} (5.5)

Or, equivalently

\[ A^T P + PA \leq -Q \]  \hspace{1cm} (5.6)

where \( A \) is the state-space matrix and \( Q \) is chosen as a symmetric positive definite matrix. If there exists a solution to the equation, \( P \), that is symmetric and positive definite, the eigenvalues of the \( A \) matrix must have negative real parts for the inequality to hold. So, what does that mean? This theorem is used to draw conclusions about the energy of a system [16]. More specifically, the internal energy of \( A \) as time progresses. The Lyapunov function of \( A \) must be decreasing with time, (i.e. energy dissipative) to be stable, as well as the derivative of the Lyapunov function must be negative for all time, \( t \). To minimize the energy in the system for control applications, \( V(x) \) is selected as:

\[ V(x) = x^T Px \]  \hspace{1cm} (5.7)

where \( x \) is the state vector of the system and \( P \) is a real valued matrix that solves the Lyapunov equation given by equation (5.6). Also, of concern is the derivative of the Lyapunov function, that must be negative for all time, \( t \). Differentiating \( V(x) \) gives

\[ \dot{V}(x) = \dot{x}^T Px + x P \dot{x} \]  \hspace{1cm} (5.8)
Substituting \( \dot{x} = Ax \) into equation (5.8) gives

\[
\dot{V}(x) = (Ax)^T P x + x^T P (Ax)
\]  
(5.9)

\[= x^T A^T P x + x^T P A x \]  
(5.10)

\[= x^T (A^T P + PA) x \]  
(5.11)

Therefore, from the Lyapunov equation

\[
\dot{V}(x) \leq -x^T(Q)x
\]  
(5.12)

This derivation provides the constraints on the \( P \) matrix.

### 5.5 Lyapunov Stability: An Example

As an example, consider the transfer function given by equation (5.13) and equivalent state-space representation in controllability canonical form given by (5.14):

\[
G(s) = \frac{1}{(s + 3)(s + 15)}
\]  
(5.13)

\[
\begin{bmatrix}
0 & 1 \\
-45 & -18
\end{bmatrix} x + \begin{bmatrix}
0 \\
1
\end{bmatrix} u, \quad y = \begin{bmatrix}
1 & 0
\end{bmatrix} x + 0
\]  
(5.14)

With the Lyapunov function defined as \( V(x) = x^T P x \) there exists a matrix \( P \) such that the condition given by equation (??) is met:
\[ A^T P + PA \leq -Q \]  

(5.15)

where \( Q \) is defined as an identity matrix with the same dimensions as \( A \). Using the `lyap` command in Matlab, \( P \) is given by

\[
P = \begin{bmatrix}
0.2284 & -0.5 \\
-0.5 & 1.2778
\end{bmatrix}
\]  

(5.16)

To verify, the operation \( A^T P + PA \) gives

\[
A^T P + PA = \begin{bmatrix}
-1 & 0 \\
0 & -1
\end{bmatrix} = -Q
\]  

(5.17)

Since \( P \) exists and is positive definite, the system given by equation (5.14) is stable in the sense of Lyapunov. For further proof, the evolution of the system’s states are examined when subjected to initial conditions only (the zero input case). As shown in Figure 5.2, it can be seen that the states move from the initial condition to the stable equilibrium of \((0, 0)\) as time progresses.

It can also be seen that the states begin inside the dotted line of radius \( \delta \) and never exceeds this bound. Thus, the condition mentioned in equation (5.3) is met. Therefore, the system can be considered stable in the sense of Lyapunov. Furthermore, since the states begin inside \( \delta \) and never grows larger than \( \delta \), the system is said to be asymptotically stable. This condition of asymptotic stability is illustrated in another of Lyapunov’s stability theorems.

Let us define a system that is already said to be stable in the sense of Lyapunov (equation (5.3)) and also meets the following condition...
| \text{x}_0 - \text{x}_e | < \delta(t_0) \Rightarrow \lim_{t \to \infty} x(t) = x_e \quad (5.18)

Since the states are always maintained inside $\delta$, the system is said to be asymptotically stable (i.e. $x(t) \to x_e$ as $t \to \infty$). Though the conditions for stability on the Lyapunov matrix, $P$, proves that the system is stable, it can be further classified as asymptotically stable by examining the trajectory of the states variables over time. It should be noted that there are further classifications of stability in Lyapunov theory: exponential stability, global stability, etc. For the purpose of this research, systems will be classified as either stable in the sense of Lyapunov or asymptotically stable.
5.6 Hybrid System Lyapunov Functions

The Lyapunov function can be found and plotted as a representation of the energy in the system versus time. To accomplish this, the Lyapunov matrix, $P$, must be found. Next, the state vector, $x(t)$, is used to find the Lyapunov function, $V(t)$.

$$V(kT) = x_k P x_k^T$$

(5.19)

where $k$ denotes the $k^{th}$ sampling interval and $T$ is the sampling period of the simulation. A decreasing Lyapunov function is shown in Figure 5.3. It can be seen, the Lyapunov function has a maximum value and decays to zero as time progresses: this denotes a stable system.

![v(t) vs. t](image)

Figure 5.3: Example Decreasing Stitched Lyapunov Function.

Figure 5.4 shows a series of stitched Lyapunov functions. It can be seen that the stitched function decays to zero. The discrete steps seen during stitching are acceptable as long as the following two conditions are met.

- The Lyapunov functions in each interval are decreasing
- The maximum value of the Lyapunov function in the $i^{th}$ interval is less than the
maximum value in the \((i - 1)^{th}\)

The stitched Lyapunov functions are shown in Figure 5.4. It can be seen that Lyapunov function decays in each of the sampling intervals and that the maximum of the current interval is less than the maximum of the past interval. Thus this system is stable.

Figure 5.4: Decreasing Stitched Lyapunov Function.
6.1 Introduction to Stateflow

MathWorks Stateflow is an environment for implementing and developing combinatorial and sequential based logic based on state machines and flow charts. State flow diagrams work by utilizing changing dynamics between states [17]. The system defaults to a designated block labeled as the default case. Linking the Stateflow blocks are guard conditions that govern when state transitions should take place. The edges are assigned as the paths between states. As the guard conditions are met, the system transitions to a new state; following the edges as shown in Figure 6.1.

In the Stateflow example shown in Figure 6.1 the output $y$ takes on a value of either 1 or 0 depending on the state. The default state is State 1, denoted by the downward pointing arrow atop the block labeled State 1. The guard condition is solely based on the variable $X$. The system will begin and remain in State 1 as long as $X$ is less than 0.5. Once the value of $X$ becomes larger than 0.5, the state follows the edge from State 1 to State 2 and
the value of $Y$ changes to 0. The system will remain in this state until the edge condition for transfer back into State 1 is met. So, when $X$ is less than 0.5, the system will transition on the edge from State 2 back to State 1. It should be noted that Stateflow blocks can also perform operations on variables that are part of the guard conditions. For example, if the error is an input to the state machine, this variable can be used as an edge condition as well as to determine the controller output, $u$. This will become more apparent when the fuzzy controller is implemented in Stateflow.

6.2 Stateflow Control Systems

One of the goals of this thesis is to implement a fuzzy logic controller using a Stateflow chart. Since the gain of the fuzzy controller changes with the input, the guard conditions can be utilized to change the states of the Stateflow controller in a manner similar to the fuzzy controller. The error signal, defined as the input minus the output, governs the dynamics of the fuzzy proportional controller and thus can be used as the guard condition between states. When the error signal meets a guard condition, the system switches to the appropriate mode. A fuzzy logic controller is implemented in its simplest sense as a
piecewise linear controller as shown in Figure 6.2.

![Figure 6.2: Stateflow Flow Controller Example.](image)

The controller output $U$ is determined by a scalar gain and offset in each state. This gain is multiplied by the error signal just as it is in a classical proportional controller. The default state is State PB. After the absolute value of the error signal, $e$, becomes less than 0.66, the state switches to State PM and the gain changes from a scalar factor of 0.6 to a scalar factor of 0.9. Also, the offset changes from 0.4 to 0.2. Similarly, when the absolute value of the error becomes less than 0.33, the state switches to State PS and the scalar gain changes from 0.9 to 1.5 and the offset changes from 0.2 to 0. As stated earlier, the controller’s dynamics as well as the guard conditions utilize the error signal. This is a viable technique in Stateflow as the guard conditions do not alter the value of the error signal but only transition when the guard condition is met. This is similar to an if-else statement in computer programming.

### 6.3 Building an Equivalent Stateflow Controller

The first step in building an equivalent Stateflow chart of a fuzzy controller is to generate the control surface to be implemented. The slopes of the piecewise control surface are
used to generate the gain and offset in each block and the error vector will be used to generate the guard conditions. The next step is to build the correct architecture to mimic the dynamics of the fuzzy controller. For a seven membership fuzzy controller, six blocks are needed inside the Stateflow chart as shown in Figure 6.3.

![Figure 6.3: Empty Stateflow Chart.](image)

Notice that the default state has been placed on the $PB$ State (i.e. when the error is "positive big"). This is because it is assumed that the system will start in this state when subjected to a unit step input. If the error begins at another value, then the default state can be placed appropriately. After building the Stateflow blocks and assuming the appropriate
default state, the gains and guard conditions can be added. The following figures show the addition of the gains and guard conditions based on the control surface in Figure 6.4.

![Stateflow Chart with Fuzzy Dynamics](image)

**Figure 6.5**: Stateflow Chart with Fuzzy Dynamics.

This chart, Figure 6.5, now shows that upon entry into any block, the output to the plant, $u$, is determined by the scalar gain multiplied by the error plus the offset of the line on the control surface. The guard conditions have also been added, showing that the state machine transitions from the default PB block to the PM block when the error signal is less than 0.4, and so on. The reverse conditions hold as well, if the error grows rather than decays as the simulation progresses the state transitions backwards to the $PB$ block based on the same guard conditions. A constant derivative gain of 1.25 is also added to produce appropriate damping for the system shown in Figure 6.6.

It can be seen that the controller now requires two inputs and produces two outputs. The error signal and its derivative are provided as inputs to the chart. Similarly, the two outputs, $u$ and $udot$ are summed outside the chart to produce the overall control input, $u$. 

43
This signal is then scaled appropriately before sending to the plant as shown in Figure 6.6.

6.4 FLC to Hybrid System Conversion

To verify implementation of a fuzzy logic control system as a piecewise linear hybrid system, the response of the fuzzy controller implemented using Stateflow is compared to the response of the fuzzy controller implemented using if-then rules. It can be seen from Figure 6.8 that both systems give exactly the same response. Therefore it is feasible to implement a proportional fuzzy controller as a hybrid system.
6.5 Hybrid PID Controller

In this section, the implementation of a proportional-plus-integral-plus-derivative (PID) fuzzy controller using a hybrid system is obtained. It has been shown that a PID fuzzy controller can be implemented as a PD controller and a PI controller where the PI controller is only active when the system reaches steady-state [10].
Figure 6.9: Simulink Implementation of PID Controller.

Figure 6.10: Stateflow Chart of a Hybrid PID Fuzzy Controller.

It can be seen from Figure 6.10 that the model of the PID controller is the same as for the PD controller except with the addition of two states for the integral action. The integral gain is zero for all states except one. In the "Test" state, however, the integral gain is made non-zero to drive the steady-state error to zero. This guard condition for this state is based on the second derivative of the error signal. When this value reaches zero, or a value very close to zero, the guard condition is met and the integral action turns on. The integral gain
was selected by increasing the gain stepwise from zero and observing the response. In this case, the integral gain that provided optimal performance was found to be 1.7.

Figure 6.11: Step Response of the Hybrid PID Controlled System.

The response of the system with and without the integral is shown in Figure 6.11. To illustrate the change in steady-state error, the response near steady-state has been expanded. It can be seen from this figure that upon reaching steady-state, the integral action is enacted and the steady-state error is brought to zero. The system reaches its first steady-state value at approximately 250 milliseconds. It is at this point that the hybrid system enters the integral state and the steady-state error is corrected to zero.

In this chapter, the Stateflow program is introduced as a means of implementing hybrid systems. Also, the implementation of a fuzzy controller as a hybrid system is described in this chapter. Chapter 7 examines the stability of dynamical systems using Lyapunov stability methods and phase plots. In the following chapter, the stability of a proportional fuzzy logic controller with constant derivative gain is examined through simulation.
Simulation Results

In this chapter, the stability of a second-order and third-order system that are controlled by fuzzy logic controllers is tested using the tools discussed in previous chapters. The fuzzy controllers are implemented as piecewise linear hybrid systems in Stateflow and the stitched Lyapunov functions are obtained to test the stability of the systems.

7.1 Second-Order System

The state-space representation of the second-order system given by equation (7.1).

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ -45 & -18 \end{bmatrix} x + \begin{bmatrix} 0 \\ 45 \end{bmatrix} u, \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} x + 0
\]  

(7.1)

Figure 7.1 shows the Simulink implementation of the second-order system with the Stateflow controller.
The hybrid system implemented is actually a proportional fuzzy controller with constant derivative gain since the fuzzy sets of the change in error vector are equally spaced. For simplicity of control surface representation, the plot in Figure 7.2 shows only the error versus the rule base and thus is 2-dimensional.

The state-space representation of the plant is given by equation (7.2):
\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
-45 & -18
\end{bmatrix}
\begin{bmatrix}
x \\
u
\end{bmatrix} +
\begin{bmatrix}
0 \\
45
\end{bmatrix} u,
\quad y =
\begin{bmatrix}
1 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
u
\end{bmatrix} + 0 \quad (7.2)
\]

The Simulink model of the plant given by equation (7.2) is shown in Figure 7.1. This model is used to obtain the trajectory of the states, \( x \) due to initial conditions. The state vector, \( x \), is then used to generate plots of the Lyapunov function \( V(x) \). Since there are three separate controllers corresponding to the active state of the hybrid system, three \( P \) matrices and subsequently three plots of the Lyapunov function \( V(x) \) are obtained. To find the three \( P \) matrices, the closed loop state matrices must be determined. Equations (7.3), (7.4), and (7.5) show the closed loop state matrices for each state in the hybrid system.

\[
A_{PB} = A - BK_{PB} =
\begin{bmatrix}
0 & 1 \\
-45 & -18
\end{bmatrix} -
\begin{bmatrix}
0 \\
45
\end{bmatrix}
\begin{bmatrix}
0.6 & 1/37
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
-990 & -60.6
\end{bmatrix} \quad (7.3)
\]

\[
A_{PM} = A - BK_{PM} =
\begin{bmatrix}
0 & 1 \\
-45 & -18
\end{bmatrix} -
\begin{bmatrix}
0 \\
45
\end{bmatrix}
\begin{bmatrix}
0.95 & 1/37
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
-1,545 & -60.6
\end{bmatrix} \quad (7.4)
\]

\[
A_{PS} = A - BK_{PS} =
\begin{bmatrix}
0 & 1 \\
-45 & -18
\end{bmatrix} -
\begin{bmatrix}
0 \\
45
\end{bmatrix}
\begin{bmatrix}
3.75 & 1/37
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
-5,951 & -60.6
\end{bmatrix} \quad (7.5)
\]
After the closed loop state matrices are determined, the `lyap` command in Matlab is used to determine the $P$ matrices for use in the Lyapunov function. Equation (7.6) gives the values of the $P$ matrices corresponding to the $PB$, $PM$, and $PS$ states of the hybrid system.

\[
P_{PB} = \begin{bmatrix}
8.2115 & 0.0005 \\
0.0005 & 0.0083
\end{bmatrix}, \quad P_{PM} = \begin{bmatrix}
12.78 & 0.0003 \\
0.0003 & 0.0083
\end{bmatrix}, \quad P_{PS} = \begin{bmatrix}
49.14 & 0.0001 \\
0.0001 & 0.0083
\end{bmatrix}
\] (7.6)

Since the state vector $x$ has been generated from the model shown in Figure 7.1, it is used to plot the Lyapunov function obtained using equation (7.7) as

\[
V_i(x) = x^T P_i x
\] (7.7)

where $i$ denotes one of the active states of the controller. Since the generation of the state vector $x$ is found versus time, this produces $V(x)$ as a function of time. Figure 7.3 shows the plots of the three Lyapunov functions, $V_i(x)$ as a function of time.
Since the gain in the $PS$ state is larger than the gain in the $PM$ state, the resulting Lyapunov function for the $PS$ is larger than the Lyapunov function for the $PM$ and $PB$ state. However, this function is only active when the system is in the $PS$ state. Likewise, since the gain in the $PM$ state is larger than the gain in the $PB$ state, the resulting $PM$ Lyapunov function is larger than the $PB$ Lyapunov function. To find a true representation of the Lyapunov function of the system as a whole, the time duration of the active regions is found and the functions are stitched together accordingly. Since the system defaults in the $PB$ state, it is assumed that the hybrid system will remain in this state until the error...
signal meets the guard condition to transition to the $PM$ state. Using data of the state transitions obtained from the model given in Table 7.1, the active regions can be determined. Figure 7.4 shows the stitched Lyapunov function of the system versus time.

![Stitched Lyapunov Function](image)

**Figure 7.4:** Stitched Lyapunov Functions of the System.

<table>
<thead>
<tr>
<th></th>
<th>$0 - 37ms$</th>
<th>$37ms - 64ms$</th>
<th>$&gt; 64ms$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Table 7.1:** Time Duration of the Active States.
It can be seen by examining Figure 7.4 that the Lyapunov function is decreasing as time progresses. The discontinuity (jump) between the PB and PM states is undesirable, but acceptable in this case, considering the Lyapunov function is always less than the initial. Thus, the system is said to have a decreasing Lyapunov function since it is decreasing even with the state transitions creating discontinuities. The Lyapunov function is never larger than the initial as time progresses.

During the multiplication of $x^T P x$, the two diagonal terms are significantly larger than the off-diagonal terms. If the off-diagonal terms in $P$ are ignored, and the Lyapunov function $V(x)$ is shown algebraically, it yields:

$$V(x) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = p_{11} x_1^2 + p_{22} x_2^2$$ \quad (7.8)

This shows that the maximum value of the Lyapunov function occurs when $x_1$ and $x_2$ are at their maximum values. In this thesis, the values of the state vector, $x$, is normalized to the interval $[-1, 1]$. Thus, the maximum value that the Lyapunov function can attain is $p_{11} + p_{22}$. This can be seen by examining Figure 7.4. The dotted line is plotted for the maximum value of $p_{11} + p_{22}$. 
7.2 Effects of Control Surface - 2nd Order Plant

In this section, the effects of changes in the rules vector in the control surface are examined using Lyapunov functions and state trajectories. First, the effect of only changing the $PS$ portion of the rule vector is examined. Then, the effect of only changing the $PM$ portion of the control surface is examined. Finally, the effect of changing both the $PS$ and $PM$ portions of the rule vector are examined. The plant being used for simulation is displayed in equation (7.9). There are multiple cases viewed in each graph, therefore, a particular control surface corresponds to a its own Lyapunov Function and state trajectory. This section is best viewed when the figures are printed in color.

\[ G_p(s) = \frac{45}{s^2 + 18s + 45} \quad (7.9) \]

To examine the effects of the $PS$ portion of the rule vector on the Lyapunov function and the state trajectories of the the system, the $PS$ section of the control surface is varied. The resulting Lyapunov function and trajectories are shown in Figure 7.5.
Figure 7.5: Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PS$ is Varied.
Figure 7.5 shows 6 different locations of the $PS$ portion of the rule vector. They range from 0.08 to 0.6. The associated Lyapunov functions and state trajectories are displayed. It can be seen that as the $PS$ portion of the control surface is increased, the maximum value of the Lyapunov function increases but each case reaches zero at a similar time. The same test is carried out by holding $PS$ constant and varying the $PM$ portion of the rule vector. Figure 7.6 shows the resulting control surface, Lyapunov functions, and state trajectories.
Figure 7.6: Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PM$ is Varied.
The values of the $PM$ portion of the rule vector is varied from 0.5 to 0.9 while the $PS$ state is held constant at 0.08. The associated Lyapunov functions and state trajectories are also shown in Figure 7.6. Finally, the effects of the control surface on stability are examined when both the $PS$ and $PM$ portions of the rule vector are varied. Figure 7.7 shows the control surface and the associated Lyapunov functions and state trajectories.
Figure 7.7: Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PS$ and $PM$ are Varied.
It can be seen from Figure 7.7 that the control surface changes the Lyapunov function and the state trajectories of the system but the stitched Lyapunov function is still decreasing. Therefore, the system is stable regardless of changes in the control surface of the fuzzy controller. This is due to the nature of the plant model being simulated. The root locus for the plant being simulated is shown in Figure 7.8. It can be seen from this figure that the system is stable for all positive gains of the controller.

Figure 7.8: Second Order Plant Root Locus.

To verify that the test for stability using the Lyapunov function and phase plot is accurate, the test must show instability for truly unstable cases. In the following section, a third order plant is used for simulation since a change in gain can make a third-order system unstable.
7.3 Effects of Control Surface - 3rd Order Plant

In this section, a third-order plant given by equation (7.10) is considered. It can be seen from Figure 7.9 that the system becomes unstable when the gain of the proportional controller is increased.

\[
G_p(s) = \frac{25}{s(s + 1)(s + 10)} = \frac{25}{s^3 + 11s^2 + 10s} \tag{7.10}
\]

The effect of different control surfaces on the Lyapunov functions and state trajectories are examined in a similar manner as with the second order system. First, the effect of only changing the PS portion of the rule vector is examined. Second, the effect of only changing the PM portion of the rule vector is examined. Finally, the effect of changing both the PS and PM portions of the rule vector are examined. The simulink model of the plant and hybrid controller used to generate the state vector is shown in Figure 7.10.
In the first test, only the location of the PS value of the control surface is varied. The results can be seen in Figure 7.11.
Figure 7.11: Control Surface and Corresponding Lyapunov Functions and State Trajectories when $PS$ is Varied.
It can be seen that as the $PS$ portion of the control surface reaches its largest value, ($PS = 0.4$), the Lyapunov function is no longer decreasing and the system is unstable. To further illustrate the effects of this change in the control surface, the error due to initial conditions is obtained as shown in Figure 7.12. It can be seen from Figure 7.12 that the system becomes unstable if the $PS$ portion of the rule vector increases to 0.4.

![Figure 7.12: Error due to Initial Conditions.](image)

In the second test, the value of the $PM$ portion of the rule vector is varied. The results can be seen in Figure 7.13.
Again, it can be seen from Figure 7.13, any deviation from the original value of 0.5 results in an increasing Lyapunov function as well as unstable response of the system. Therefore, for the simulated \( PM \) values greater than the baseline, the system becomes...
unstable. This can also be seen by examining the response of the system due to initial conditions shown in Figure 7.14.

![Figure 7.14: Response of the System to Initial Conditions.](image)

In the third case, the value of both $PS$ and $PM$ portions of the rule vector are changed. The results can be seen in Figure 7.15.
Figure 7.15: Varying Both PM and PS for the Third Order Plant.

It can be seen from this figure that any deviation from the baseline controller produces an unstable response. As shown in Figure 7.15, the baseline controller is the only controller with a decreasing Lyapunov function as well as output trajectories that tend to \((0, 0)\). All other cases give increasing Lyapunov functions as well as divergent output trajectories.
This is further apparent from the output response due to initial conditions shown in Figure 7.16.

Figure 7.16: Error Response due to Initial Conditions.
Conclusion

Nonlinear systems are difficult to model and control. Further, the stability of these systems is difficult to verify. Though fuzzy logic has many applications in the area of control systems, the verification of such a system is difficult. Many designers spend countless hours and millions of dollars verifying the performance of systems before or during implementation. Most solutions utilize a less-than-elegant methodology in the form of countless simulations or other extrinsic techniques; which wastes manpower and energy.

A fuzzy logic controller can be implemented as a piecewise linear controller. This piecewise linear controller can be implemented as a hybrid system and used to control plant models. The assumption that the fuzzy controller is accepted as a piecewise linear controller is verified in this thesis. The dynamics of a piecewise linear hybrid control system match exactly that of the fuzzy controller when implemented with if-then rules. Therefore, the feasibility of modeling a fuzzy controller as a piecewise linear hybrid system is justified. The link between such systems and the stability verification tools for hybrid systems provides further justification for modeling fuzzy controllers as hybrid systems.
An approach to verifying the stability of fuzzy logic controlled systems is presented in this paper. The applications of this technique can be extend to any system that can be modeled as a hybrid system. This thesis presented a method of generating stability considerations using Matlab Simulink Stateflow. Two plants were used in the simulation; the always stable type-zero second-order system and a type-one third order system that can be unstable. Both plants have real, negative roots. It was observed through simulation that the second order plant is stable for any control surface of the fuzzy logic controller. Examining the stitched Lyapunov functions yields this conclusion as they were decreasing. However, it is observed that the third order system becomes unstable for certain control surfaces. The stitched Lyapunov functions for the third order system were increasing and hence can be used to test the stability of hybrid systems.

Unique to this thesis is the use of Lyapunov functions to examine the stability of a fuzzy logic controlled system. Lyapunov functions yield information about the energy in a system. In this thesis, the Lyapunov function showed that some control surfaces yield unstable systems. For the type-zero second-order plant simulation, it was found that, when stitched, the Lyapunov functions were always decreasing. This way true for any variation in the rules of the fuzzy logic controller. In the type-one third order plant simulation, it was found that the stitched Lyapunov functions could be increasing or decreasing. This shows the existence of a boundary between stable and unstable control surfaces. Further research is needed to precisely identify this boundary and identify control surfaces that produce unstable results.

In this thesis, the proportional fuzzy controller with constant derivative gain is only considered. Future work could incorporate higher-dimensional control surfaces such as a truly fuzzy PD or PID controller. Further extensions of the research presented in this thesis could be used to examine the robustness of these systems. As mentioned previously,
the plant model is seldom exactly known, thus, the system model could be subjected to changes or failures.
Bibliography


Available: http://web.mae.cornell.edu/hadaskg/courses/mae6740.html


time assurance for complex cyber physical systems,” 2012.

697–699, 1968.


[36] N. Sahab, “A type-2 nonsingleton type-2 fuzzy logic system to handle linguistic and
numerical uncertainties in real world environments,” 2011 IEEE Symposium on

Mathematics and Computer Science, University of Salerno.

[38] B. L. Stefan Pettersson, “Stability and robustness for hybrid systems,” Control
Engineering Lab, Chalmers University of Technology, Gothenburg, Sweden, Print.


Appendix A: Matlab Code

%Aaron Seyfried
% This m-Function returns the value of the triangular membership function % corresponding to the value for x, center point and width.
%%%%%%%%%%%%%%%%%%%%%%%

function [ y ] = triang(x,cp,width)

%UNTITLED Summary of this function

lower = cp - (width/2); %defines lower limit zero point of triangle
upper = cp + (width/2); %defines upper limit zero point of triangle

if(x<=lower) %handles below lower limit case
    y = 0;
elseif (x>=upper) %handles above upper limit case
    y = 0;
elseif (x>lower && x<cp)
    y = ((x-lower)/(cp-lower)); %between lower and center point
else
    y = ((x-upper)/(cp-upper)); %between center point and upper
end

%%%%%%%%%%%%%%%%%%%%%%%
function [y]=fuzzy(x,centers)
d = length(centers);
%'ZEROS' Zeros all the arrays
y=zeros(1,d);
%test for extremes of Universe of discourse and for values within universe
if x <= centers(1)
    y(1) = 1;
elseif x >= centers(d)
    y(d)=1;
elself
    for i=1:d
        if (x>=centers(i)) & (x <= centers(i+1));
            y(i)=(centers(i+1)-x)/(centers(i+1)-centers(i));
            y(i+1)=(x-centers(i))/(centers(i+1)-centers(i));
        end;
    end;
end;

%Aaron Seyfried
%This function is used by the FLC to compute the output of the proportion controller
%UNTITLED3 Summary of this function goes here
% Detailed explanation goes here

function [ y ] = PFLC_(x, centers, rulevector)
%UNTITLED3 Summary of this function goes here
% Detailed explanation goes here

    out = [];
    out = fuzzy(x, centers);
    y = out*rulevector';
end
function [ y ] = PDFLC(x0,x1,centers1,centers2,rule_matrix)

I1 = []; %Defines inputs as vectors
I2 = [];

I1 = fuzzy(x0, centers1); % defines I1 given the centers and x value
I2 = fuzzy(x1, centers2); % defines I2 given the centers and x value

y = I1*rule_matrix*(I2');  % y
end

%Aaron Seyfried
%This program requires inputs for the control surface to generate the Lyapunov functions
%and state trajectories for the second order plant.

Kd_p = (1/37); %1/Max delta e from scope 3 in pdflc
centers_e = [-1 -.5 -.08 0 0.08 .5 1]; %x axis of gain triangles
%centers_e = [-1 -2/3 -1/3 0 1/3 2/3 1]; %x axis of gain triangles
a = 1;
centers_ce = [-1 -2/3 -1/3 0 1/3 2/3 1]; %Don’t change this

NB = -1; %y axis of gain triangles
NM = -.5; %-.4;
NS = -.08; %-.066;
ZR = 0;
PS = -1*NS;%.066;
PM = -1*NM;%0.4;
PB = 1;
rule_vector = [NB NM NS ZR PS PM PB];

rule_matrix = [NB NB NB NB NM NS ZR;
               NB NB NB NM NS ZR PS;
               NB NB NM NS ZR PS PM;
               NB NM NS ZR PS PM PB;
               NM NS ZR PS PM PB PB;
               NS ZR PS PM PB PB PB;
               ZR PS PM PB PB PB PB];

plot(centers_e,[NB NM NS ZR PS PM PB])
grid
axis([0 1 0 1])
m = -1*[(NB-NM)/(centers_e(7)-centers_e(6));
       (NM-NS)/(centers_e(6)-centers_e(5));
       (NS-ZR)/(centers_e(5)-centers_e(4))];

% Effective value of Kp (for error)
for j=1:6
    num=rule_vector(j+1)-rule_vector(j);
    den=centers_e(j+1)-centers_e(j);
    kpe(j)=num/den;
    numc=centers_e(j+1)*rule_vector(j)-centers_e(j)*rule_vector(j+1);
    cons(j)=numc/den;
end
%Simulate the Model
sim PSFmV1_1

%Initialize B and K for CL feedback
B = [0 45]';
Kpb = 35*[kpe(3) Kd_p];
Kpm = 35*[kpe(2) Kd_p];
Kps = 35*[kpe(1) Kd_p];

%Closed Loop A matrices
A1 = [0 1; -45 -18]-B*Kpb;
A2 = [0 1; -45 -18]-B*Kpm;
A3 = [0 1; -45 -18]-B*Kps;

%Determine Lyapunov Functions for CL A matrices
P1 = lyap(A1',eye(rank(A1)));
P2 = lyap(A2',eye(rank(A2)));
P3 = lyap(A3',eye(rank(A3)));

emat = xout.signals.values;
for i = 1:size(xout.time)
    Vx1(i) = emat(i,:)*P1*emat(i,:)';
end

for i = 1:size(xout.time)
    Vx2(i) = emat(i,:)*P2*emat(i,:)';
end
for i = 1:size(xout.time)
    Vx3(i) = emat(i,:)*P3*emat(i,:)’;
end

% Generate Variable to find active regions for Vx_i based on
% the state transitions
state1 = state.signals.values;
state1 = state1’;
ind = [1 2 3 4 5 6 7 8 9];
state1(ind) = [];
choice = [ones(1,9) state1];
%%
% Stitching together the segments of Vx_i into one vector.
for i = 1:length(choice)
    if choice(i)== 3
        Vcomm(i) = Vx1(i);
    elseif choice(i) == 2
        Vcomm(i) = Vx2(i);
    elseif choice(i) ==1
        Vcomm(i) = Vx3(i);
    end
end

%% Plots
figure
subplot(121)
hold on
plot(xout.time,Vx1, 'k')
plot(xout.time,Vx2, 'g')
plot(xout.time,Vx3, 'r')
legend('PS','PM','PB')
axis([0 0.14 0 (max(Vx1)+3)])
title('V(x)')
xlabel('Time')
ylabel('V_i(x)')
grid
subplot(122)
hold on
plot(xout.time,Vcomm, 'k')
axis([0 0.14 0 (max(Vcomm)+0.5)])
title('Stitched V_1, V_2, V_3')
xlabel('Time')
ylabel('V_i(x)')
grid

hold_rv = [hold_rv [NB NM NS ZR PS PM PB]]; 
hold_xout = [hold_xout xout.signals.values]; 
hold_time = [hold_time xout.time]; 
hold_Vcomm = [hold_Vcomm Vcomm']; 

% hold_rv = []; 
% hold_xout = []; 
% hold_time = []; 
% hold_Vcomm = [];