Stability-Based Hybrid Automata for Safety Verification Using Continuation Methods

Peter Uth

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics & Astronautics

University of Washington

2017

Committee:

Anshu Narang-Siddarth, Chair

Christopher Lum

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

Stability-Based Hybrid Automata for Safety
Verification Using Continuation Methods

Peter Uth

Chair of the Supervisory Committee:
Assistant Professor Anshu Narang-Siddarth
Aeronautics & Astronautics

The rapid development of increasingly autonomous systems has advanced the challenge of safety assurance beyond the capabilities of existing methods. Therefore, new means of verification and validation are required to ensure the safe operation of emerging systems. Numerical continuation characterizes system behavior as parameters are varied and can be used to facilitate a bifurcation analysis, where equilibria and their stability properties are identified. This paper introduces hybrid stability automata – system models constructed using numerical continuation that capture stability properties within a series of dynamic modes. These automata readily support safety analyses by explicitly defining stable, i.e. safe, regions of the operational envelope. The processes to create hybrid stability automata for one-dimensional and multi-dimensional systems are discussed and prototypical examples are presented. Example safety analyses using hybrid stability automata are demonstrated on the space shuttle reentry dynamics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This research is supported by the National Science Foundation under grant no. CPS-1658659, the Air Force Research Laboratory, and the Washington Research Foundation Capital. In addition to these organizations, the author would like to thank his adviser, Professor Anshu Narang-Siddarth, his lab mates in the Advanced Dynamics, Validation & Control Research Laboratory, and his friends/family.

Chapter 1

INTRODUCTION

Assuring the safe performance of a system is a critical component of the design process. An analysis of a system's underlying dynamics can result in the detection of unsafe properties early in the process, thus avoiding more expensive design changes in later stages of development. However, a complete understanding of nonlinear system dynamics is typically difficult to obtain due to the wide range of possible behavior. This presents a fundamental challenge to ensuring safe operation, particularly for highly autonomous systems with large sets of possible operating conditions. As advances in autonomy yield more intelligent and complex systems, the common practice of verifying safety properties by performing a time simulation for every possible set of operating conditions becomes inadequate. Verification and validation (V&V) architectures that employ model checking [1, 2, 3, 4] or theorem proving [5, 6] techniques can provide an exhaustive means of verifying safety properties. However while these techniques are useful, they are restrictive in the class of systems they can handle and are often computationally intensive. Therefore, new V&V techniques are required to keep pace with the development of increasingly intelligent and diverse systems.

Numerical continuation methods can be used to characterize how the behavior of a system changes as parameters are varied via a bifurcation analysis [7]. The term bifurcation defines a point in state-parameter space for a dynamical system where changes occur in the number and/or stability of equilibrium solutions [8]. The results of a numerical continuation analysis, typically a bifurcation diagram depicting the equilibrium solution branches and their stability, provide insight on safe and unsafe operating conditions for the system being analyzed. Since this analysis is typically performed with a single computational process, a continuation-based approach to safety verification can provide advantages in computational

efficiency over methods that rely on repeated simulations.

While bifurcation diagrams are useful representations of system behavior, they do not explicitly define stable and unstable regions of the operational envelope. This paper presents the concept of hybrid stability automata, a method of modeling a system that uses the stability information obtained through numerical continuation analysis to capture the basins of attraction (i.e. the domain of initial conditions for which trajectories will converge to a stable equilibrium) to identify stable and unstable domains of the state-space. A hybrid automaton is a model that represents a system's dynamics as a series of modes with continuous-time dynamics. Discrete switches activate transitions between these modes depending on the satisfiability of certain conditions. Hybrid systems, including hybrid automaton representation, and verification approaches are detailed in [9]. A hybrid stability automaton expands on the structure of a typical hybrid automaton by adding valuable stability information to a series of modes and switches. Since stability properties directly correlate to safety properties, hybrid stability automata are well-suited for automated safety analyses and provide useful visualizations of system behavior. Additionally, this type of representation has the potential to interface with verification schemes such as model checking and theorem proving to enhance their capabilities. Software is developed that generates hybrid stability automata for one-dimensional (i.e. single state) and multi-dimensional (i.e. two or more states) systems. The output is a graphical visualization of the automaton and an executable function that accesses the dynamic modes based on the current satisfiability of the mode conditions.

The organization of this thesis is as follows. Chapter 2 provide relevant background information. Topics include numerical continuation, bifurcation analysis, hybrid automata, and existing V&V tools. Hybrid stability automata are introduced in Chapter 3, along with the construction process and examples for one-dimensional systems. Chapter 4 follows with hybrid stability automata for multi-dimensional systems. In Chapter 5, the space shuttle reentry dynamics are used to demonstrate analyses using both one-dimensional and multi-dimensional hybrid stability automata. Conclusions and recommendations for future work are presented in Chapter 6.

Chapter 2
BACKGROUND

This chapter provides background information fundamental to the work presented in this thesis. The following sections focus on numerical continuation, bifurcation analysis, hybrid automata, and existing verification tools.

2.1 Numerical continuation

Numerical continuation methods seek to approximate sets of solutions to equations in the form of

$$F(z) = 0, \tag{2.1}$$

where z is a vector of unknowns [7]. The system must be underdetermined such that F is a smooth map where

$$F : \mathbb{R}^{n+1} \to \mathbb{R}^n, \tag{2.2}$$

i.e. the number of unknowns must be one greater than the number of equations. Sets of z that satisfy equation (2.1) are referred to as *solution branches*. Continuation numerically approximates these solution branches via predictor and corrector iterations. This process is as follows. First, an initial point on the solution curve is identified. Then, a predictor step is made. An Euler predictor of the form

$$z_{i+1} = z_i + h\left(F_z\left(z_i\right)\right) \tag{2.3}$$

is often used, where h is a step in some element of z. F_z is the Jacobian matrix where the subscript z denotes differentiation with respect to z (i.e. $F_z = \partial F/\partial z$). Next, a corrector is applied to solve the minimization problem

$$\min_{w}\{\|v-w\|\big|F(w)=0\}, \tag{2.4}$$

thus increasing the accuracy of the updated predicted point v to the solution point w until a defined tolerance is reached. A Newton-type corrector of the general form

$$z_{i+1} = z_i - F_z\left(z_i\right)^{-1} F\left(z_i\right) \tag{2.5}$$

is commonly used and iterated until the Euclidean norm $\|v-w\|$ is smaller than the desired tolerance. The newly obtained corrected point is then used as the next initial point and the process is repeated. These predictor-corrector steps are iterated to trace a solution branch. This process is illustrated in Figure 2.1, where u_i is the initial point, v_{i+1} is the predicted point, u_{i+1} is the corrected point, and w_{i+1} is the target point on the original nonlinear solution branch [7].

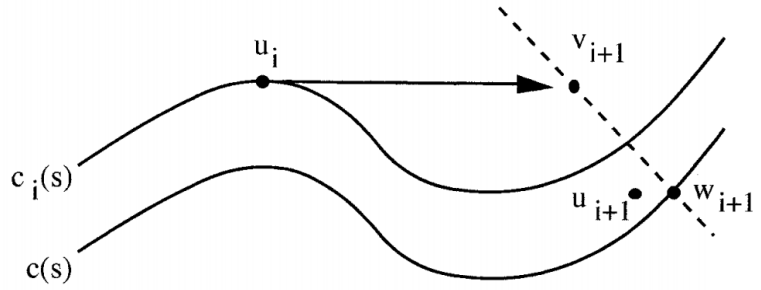

Figure 2.1: Continuation process illustrated.

2.1.1 Numerical Continuation Software

The Continuation-based System Analysis (COSY) tool [10, 11] is a numerical continuation algorithm written in MATLAB that employs a pseudo-arclength predictor-corrector continuation method. The predictor step is defined by

$$z_{i+1} = z_i + t_i \Delta_i, \tag{2.6}$$

where t_i is a step in the direction tangent to the solution at the initial point (found using the Jacobian F_z) and Δ_i is the step size at i. The pseudo-arclength predictor in equation (2.6) differs from the Euler predictor in equation (2.3) in that the predictor step moves tangentially rather than with a fixed step h. This difference gives pseudo-arclength continuation the ability to handle turning points of a solution curve that cause the Euler predictor to fail. The COSY tool adapts the step size Δ_i at every step depending on the number of corrector iterations that were required to reach the desired tolerance; Δ_i is decreased if many iterations were required and increased if few iterations were required. This feature allows for flexibility in capturing system behavior by balancing computational speed and solution accuracy. COSY is used for all numerical continuation processes described in this thesis.

2.2 Bifurcation analysis

A *bifurcation analysis* typically investigates changes in the number and stability of equilibria for dynamical systems of the form

$$\dot{x} = f(x, p), \tag{2.7}$$

where x is the vector of system states and p is a parameter [8]. *Equilibrium points* occur at sets of x and p that solve $\dot{x} = f(x, p) = 0$. Therefore, equilibrium properties are dependent on system behavior at values of p. Numerical continuation determines these properties by finding solution sets as parameter p is varied using the process outlined in Section 2.1 with $z = \begin{bmatrix} x & p \end{bmatrix}^T$, therefore

$$F(z) = F(x, p), \quad F_z = \begin{bmatrix} F_x & F_p \end{bmatrix}. \tag{2.8}$$

The result is information on the number and location of equilibria for different values of parameter p. The *stability* of each equilibrium is determined by observing the Jacobian

matrix at each step of the continuation process, where negative values indicate stable and positive values indicate unstable. In the context of dynamical systems, stability is defined by trajectories that converge to stable equilibria and diverge away from unstable equilibria. The following sections describe prototypical bifurcation cases and the different types of equilibria.

2.2.1 Saddle Node Bifurcation

Saddle node bifurcations occur when two equilibrium points annihilate each other for some value of parameter p. For example, consider the dynamical system

$$\dot{x} = p + x^2. \tag{2.9}$$

Figure 2.2 shows $\dot{x}$ vs. x for different values of parameter p. When $p < 0$, there are two solutions to $\dot{x} = 0$ given by $x^* = \pm\sqrt{-p}$, where x^* represents values of x at equilibrium points. When $p > 0$, the two solutions disappear. Therefore, a saddle node bifurcation occurs at $p = 0$. The stability of the solution points correlates to the sign of the derivative $f'(x) = 2x$. At the equilibrium in the negative x-domain, $f'(x^*)$ is negative and therefore the point is stable. At the equilibrium in the positive x-domain, $f'(x^*)$ is positive and therefore the point is unstable. As seen in Figure 2.2, stable and unstable equilibria are represented by filled and unfilled circles, respectively. The bifurcation point at $x^* = 0$ when $p = 0$ is half-stable, i.e. it attracts in one direction but repels in the other.

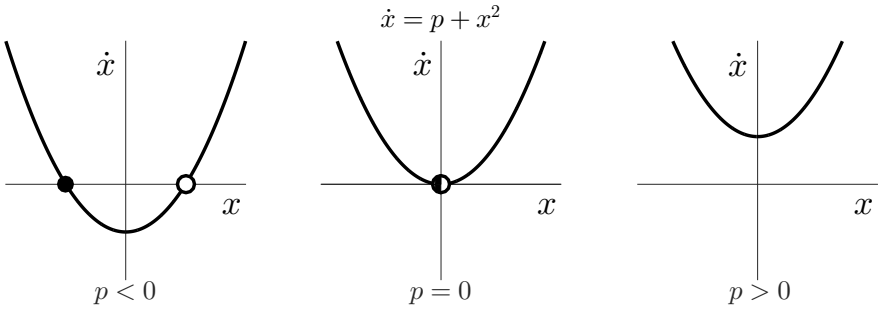

Figure 2.2: Output of $\dot{x} = p + x^2$ for varying p. A saddle node bifurcation occurs at $p = 0$.

Equation (2.9) presents an underdetermined problem since it contains one equation and two unknowns, p and x. Numerical continuation can be used to analyze the relationship between state x and parameter p by finding equilibrium solution branches for $\dot{x} = 0$. The bifurcation diagram resulting from continuation is shown in Figure 2.3 with p on the x-axis and x on the y-axis. Notice that this bifurcation diagram captures the various behaviors seen in Figure 2.2: two equilibrium branches exist when $p < 0$ but disappear when $p > 0$. The equilibrium in the negative x-domain is stable (displayed as solid blue) while the positive x-domain contains an unstable equilibrium (dashed red). The point where the saddle node bifurcation occurs is labeled as "SN." The exact values of x^* along the solution branches depend on the value of p.

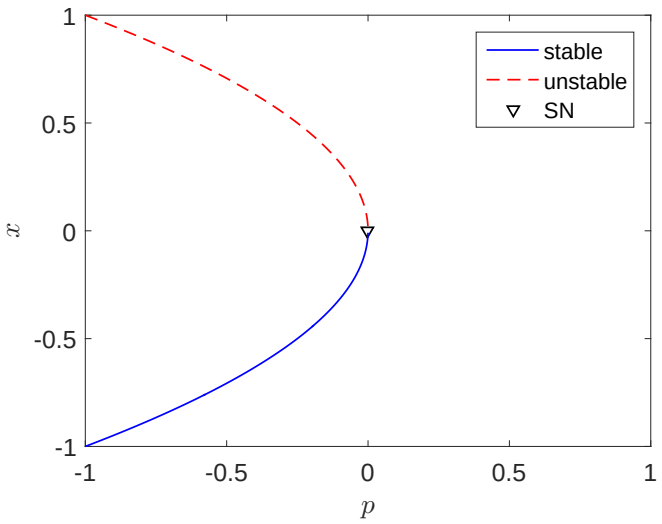

Figure 2.3: Bifurcation diagram for $\dot{x} = p + x^2$.

2.2.2 Pitchfork Bifurcation

Pitchfork bifurcations occur when a transition is made from a single equilibrium point to three equilibrium points for some value of parameter p. For example, consider the system

$$\dot{x} = px - x^3. \tag{2.10}$$

Figure 2.4 shows $\dot{x}$ vs. x for different values of parameter p. When $p < 0$, one stable equilibrium exists at $x^* = 0$. When $p > 0$, $x^* = 0$ remains an equilibrium but switches from stable to unstable and two additional stable equilibria appear. Therefore, a pitchfork bifurcation occurs at $p = 0$. The corresponding bifurcation diagram produced via numerical continuation is shown in Figure 2.5, where the location of the pitchfork bifurcation is labeled as the branch point "BP."

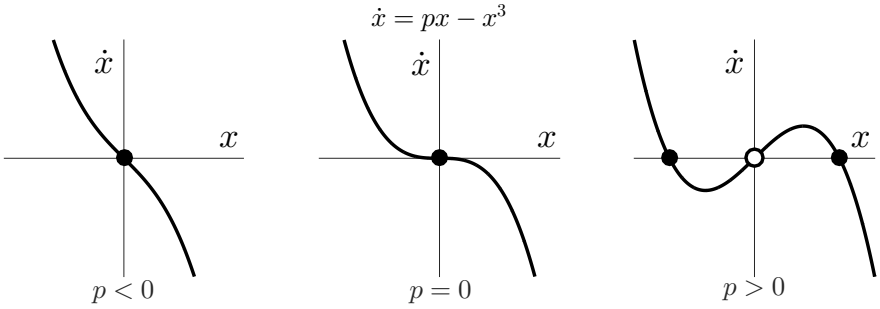

Figure 2.4: Output of $\dot{x} = px - x^3$ for varying p. A pitchfork bifurcation occurs at $p = 0$.

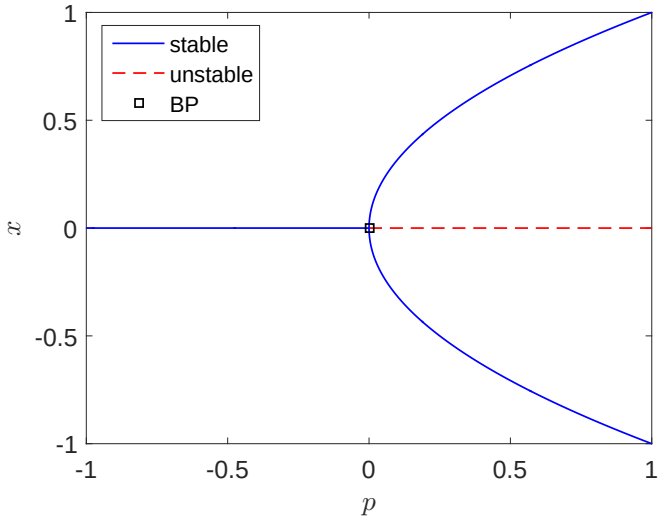

Figure 2.5: Bifurcation diagram for $\dot{x} = px - x^3$.

2.2.3 Transcritical Bifurcation

Transcritical bifurcations occur when two equilibrium points switch their stability. For example, consider the dynamical system

$$\dot{x} = px - x^2. \tag{2.11}$$

Figure 2.6 shows $\dot{x}$ vs. x for different values of parameter p. Two equilibrium points exist at $x = 0$ and $x = p$ for both positive and negative domains of p. When $p < 0$, $x = 0$ is stable and $x = p$ is unstable. However, when $p > 0$, $x = 0$ is unstable and $x = p$ is stable. Therefore, a transcritical bifurcation occurs at $p = 0$. The corresponding bifurcation diagram produced via numerical continuation is shown in Figure 2.7, where the location of the transcritical bifurcation is labeled as the branch point "BP."

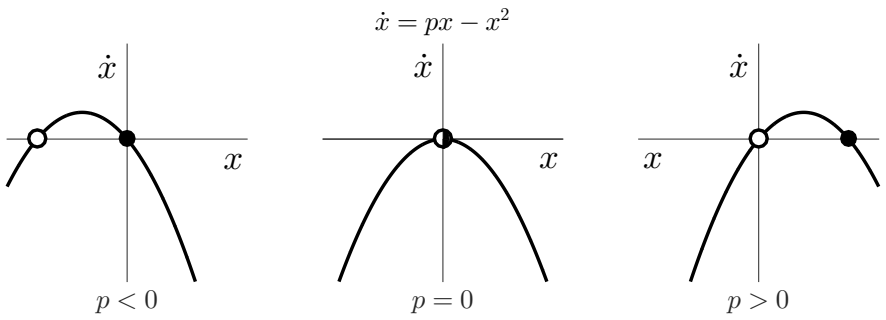

Figure 2.6: Output of $\dot{x} = px - x^2$ for varying p. A transcritical bifurcation occurs at $p = 0$.

2.2.4 Types of Equilibria

In one dimension, trajectories can only evolve monotonically towards a stable equilibrium or diverge to infinity [8]. Therefore, oscillatory behavior and overshoot are not possible in one-dimensional systems. This makes the behavior of one-dimensional systems relatively predictable if the locations of equilibrium points are known; trajectories will either converge to an adjacent, stable equilibrium or diverge.

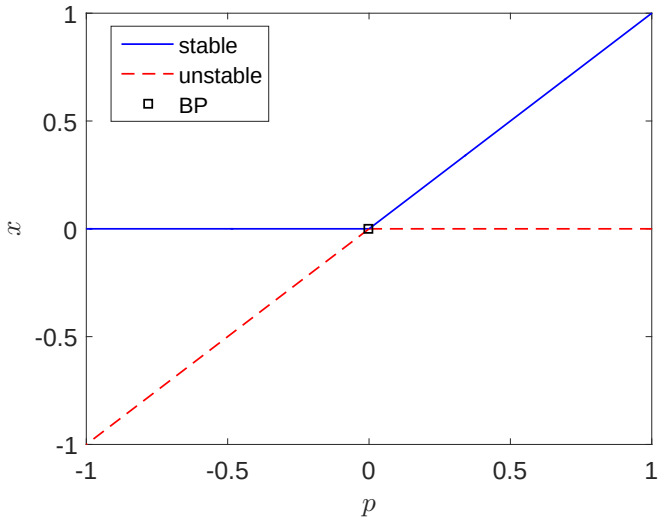

Figure 2.7: Bifurcation diagram for $\dot{x} = px - x^2$.

In higher dimensions, this monotonic behavior is no longer guaranteed and trajectories become much more difficult to predict. Different types of equilibria are possible and can be identified using the eigenvalues of the Jacobian matrix taken at the equilibrium point. The following equilibrium types are examples for two-dimensional systems, and sample phase portraits are shown in Figure 2.8 [12].

- *Nodes* have two real eigenvalues, both negative for a stable node and both positive for an unstable node. Trajectories are attracted or repelled.

- *Saddle points* have two real eigenvalues, one negative and one positive. Trajectories are attracted in one direction and repelled in the other. Saddle points are considered unstable since trajectories do not converge to some value as $t \to \infty$.

- *Centers* have two purely imaginary eigenvalues. Trajectories are neither attracted nor repelled, therefore centers are considered neutrally stable.

- *Spirals* have two complex eigenvalues. If the real parts of the eigenvalues are negative, the spiral is stable. If the real parts of the eigenvalues are positive, the spiral is unstable. Trajectories spin towards stable spirals and away from unstable spirals.

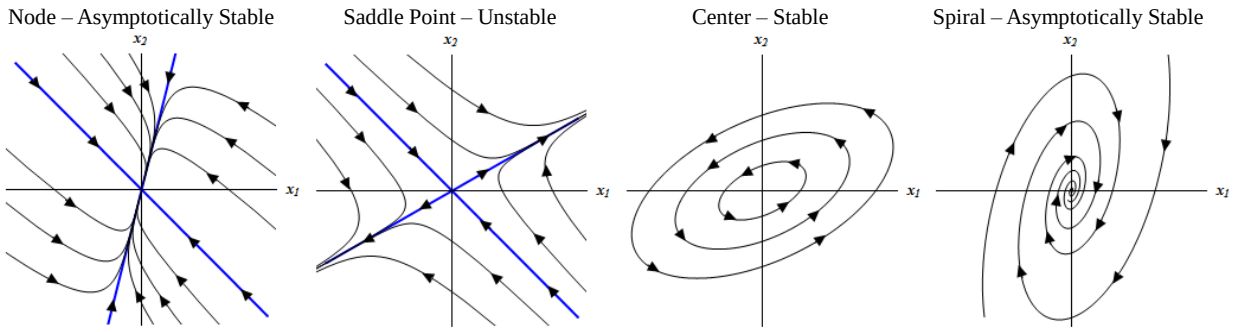

Figure 2.8: Phase portraits for different 2D equilibrium types.

2.3 Hybrid Automata

A dynamical system that contains both continuous and discrete components is referred to as a *hybrid system* [13]. A *hybrid automaton* is a formal model of such a system that captures the interactions between continuous and discrete components to represent system behavior. Typically, a hybrid automaton consists of control *modes* that contain locally relevant, continuous dynamics and discrete control *switches* that transition between the modes depending on the satisfiability of certain conditions.

A simple example of a hybrid automaton from [13] for a thermostat is shown in Figure 2.9. In this example, the state x represents temperature. At the initial temperature of $x = 20$, the heater-off control mode is active and the temperature falls according to $\dot{x} = -0.1x$. When the transition condition $x < 19$ is satisfied, a control switch is activated and the heater-on control mode becomes active, with the temperature now changing according to $\dot{x} = 5 - 0.1x$. Once the $x > 21$ transition condition is satisfied, the system returns to the heater-off mode. This cycle repeats as necessary to regulate temperature to the desired range.

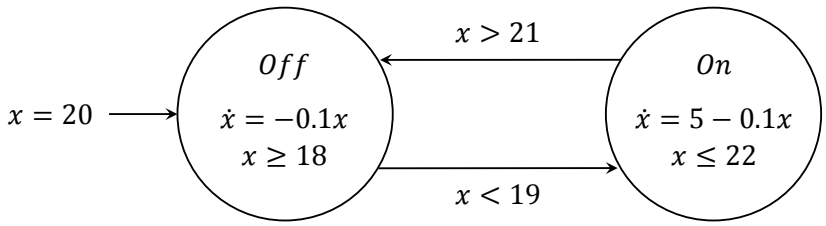

Figure 2.9: Hybrid automaton for a thermostat.

2.4 Verification and Validation tools

Two of the most commonly used types of V&V tools are model checkers and theorem provers. The following sections provide basic details on these methods.

2.4.1 Model checking

The term *model checking* describes techniques to automatically verify the correctness of finite-state systems [1, 2]. The inputs to a model checker are a mathematical model of the system to be verified and the property requirement, typically expressed as logic formulas, that are used to determine correctness. Model checkers then apply exhaustive search algorithms to the entire state-space to determine if the property requirement holds true for all states or is violated for at least one state. Figure 2.10 depicts the general structure of model checking techniques. For increasingly complex systems, an exponential growth of this exhaustive search process leads to a state-space explosion problem. Therefore, model checkers require abstraction of the target system to ease the computational process [3, 4]. This sacrifices fidelity to the original system and often necessitates a separate review process to ensure that the abstracted model adequately reflects the original system.

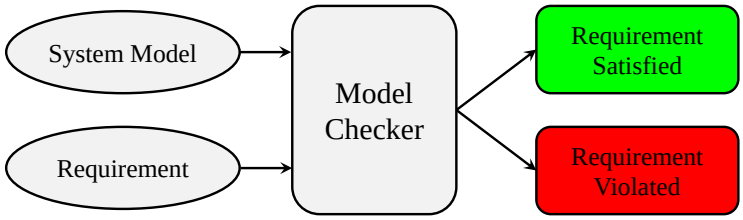

Figure 2.10: Top-level structure for model checking requirement verification.

2.4.2 Theorem proving

Theorem proving, or automated reasoning, is a commonly used verification method [5, 6]. Mathematical models of the system and the property specification must be formalized into a logic-based architecture. Typical theorem provers work by expanding on well-known axioms to deduce the validity of the specification for the system being verified. The logic formalization restricts the capabilities of theorem proving, with trade-offs between system complexity and automation potential for three levels of logical expression: propositional, first-order, and higher-order. Propositional logic is very limited in the types of systems that can be expressed, but is easily incorporated into automated verification schemes. First-order logic has fewer limitations on system expressions, but may require some user interaction. Higher-order logic can be used to express complex systems, but is highly user-interactive and therefore not suitable for automatic verification.

Chapter 3

HYBRID STABILITY AUTOMATA

This chapter presents the concept of *hybrid stability automata*. This replicates the structure of typical hybrid automata where a system is represented by a series of control modes that contain continuous subset dynamics and discrete transition switches between these modes that are activated when certain parameter criteria are met. Within each of these modes, referred to as *p-modes* for their dependence on system parameters, is a sub-level of modes and switches that are dependent on the system state, referred to as *x-modes*. Each x-mode is labeled with its stability, equilibrium solution, and basin of attraction or repulsion. If the system is within the basin of attraction of a stable x-mode, it will trend towards the equilibrium value. On the other hand, a system within the basin of repulsion of an unstable x-mode will trend away from the equilibrium value. Transitions between the x-modes rely on certain system state criteria.

Hybrid stability automata are useful for system analysis and safety verification since the safe and unsafe, i.e. stable and unstable, regions of the operational envelope are explicitly identified through the basin information. In this way, a hybrid stability automaton captures valuable information on the steady state time dynamics of a system without requiring time simulations. While bifurcation diagrams such as the examples presented in Section 2.2 provide useful insight on the existence and stability of equilibrium values, no explicit relation is made to safe/unsafe conditions within the system's state-space. Therefore, hybrid stability automata are better suited for automated safety analyses than bifurcations diagrams. Additionally, since existing V&V tools such as model checkers and theorem provers typically rely on logical expressions, the logic-based architecture of a hybrid stability automaton can potentially be leveraged to create an integrated V&V approach. For example, a model checker

may be compatible with the hybrid stability automaton of a system for which it otherwise could not handle.

The proposed process for generating a hybrid stability automaton begins by applying numerical continuation to obtain the information on system equilibria. Then, the basins for the equilibria are determined. Data is formulated into modes and a graph representation tool is used to produce the automaton visualization. An executable function for the automaton is also created. The processes for creating hybrid stability automata for single-parameter and multi-parameter systems, along with some basic examples, are detailed in this chapter.

3.1 Single-Parameter Process

The following sections describe the process for creating hybrid stability automata for one-dimensional, one-parameter systems. Aside from the user defined inputs in 3.1.1, these steps are performed automatically using an algorithm created for this work.

3.1.1 Step 1: Run Continuation Analysis

The first step of the hybrid stability automaton process is to apply the continuation analysis to the system dynamics. The following inputs are provided to the COSY continuation tool:

- System dynamics, $f(x, p)$

- Identification of *active continuation parameters* within the dynamics – These are the unknowns that numerical continuation is allowed to vary to find solution branches. System state x and parameter p are typically identified as active continuation parameters.

- Initial conditions for the active continuation parameters – This is intended to provide an initial point along the solution branch from which the continuation process will begin. A separate solver step can be applied to find an initial point, or a sufficiently

close guess can be made and adjusted automatically via the Newton corrector within COSY.

- Continuation algorithm options – At a minimum, this includes upper and lower limits for the values of an active continuation parameter (typically system parameter p). Other optional inputs include, but are not limited to, solution tolerance, limits for continuation step size, and maximum number of continuation steps.

The output is sets of active continuation parameter values (e.g. x and p) that solve $f = 0$. The eigenvalues of the Jacobian matrix at each point along the solution branch are included in the output, which are used to determine stability. *Bifurcation points*, where the number and/or stability of solutions change, are considered *events* that are used to separate domains of system parameter p. For example, if a single bifurcation occurs at $p = 0$, two *p-domains* are created: one for $p < 0$ and one for $p > 0$ to capture the different system dynamics that exist within these domains. These p-domains will be used in the definition of the p-modes. To illustrate this separation of p-domains, Figure 3.1 shows the bifurcation diagram resulting from a continuation analysis of the system $\dot{x} = px + x^2 - x^3$. The branch point "BP" and the saddle node bifurcation point "SN" are the events used to separate the dynamics into three p-domains, $p < -0.25$, $-0.25 < p < 0$, and $p > 0$, which are bordered by the dashed vertical lines in Figure 3.1.

3.1.2 Step 2: Identify Basins of Attraction

The *basin of attraction* for a stable equilibrium x^* is defined as the set of initial conditions x_0 such that

$$\lim_{t\to\infty} f(x_0, t) = x^*. \tag{3.1}$$

For hybrid stability automata, the *basin of repulsion* for an unstable equilibrium x^* is the set of initial conditions in which $f(x_0, t)$ diverges away from x^* and does not converge to

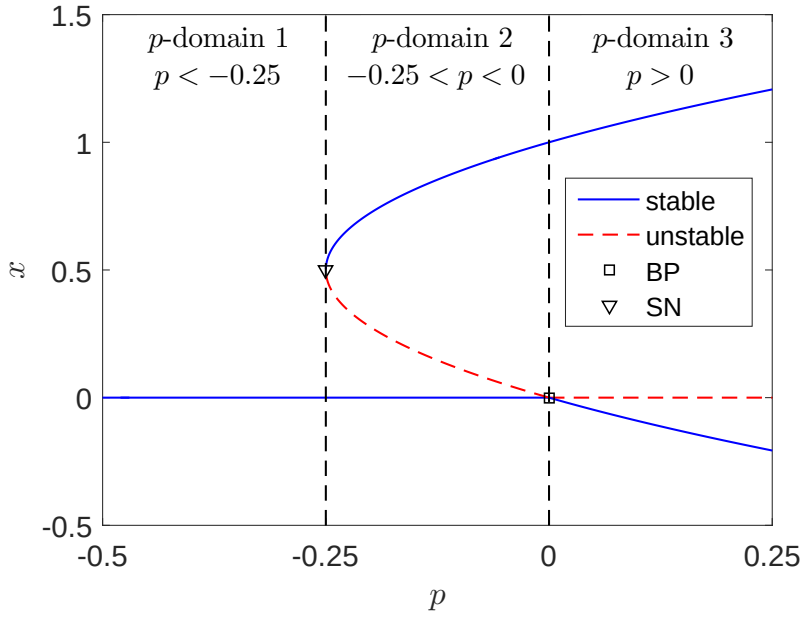

Figure 3.1: Bifurcation diagram with three different p-domains labeled.

any stable equilibrium as $t \to \infty$. Figure 3.2 shows examples of these basins overlaid on the prototypical bifurcation diagrams previously shown in Figures 2.3, 2.5, and 2.7. These basins are used to define x-modes for hybrid stability automata.

For the saddle node system in Figure 3.2, no basins exist when $p > 0$. However when $p < 0$, a basin of attraction appears for the stable equilibrium branch that extends from $x = -\infty$ to $x = x^{*2}$, where x^{*2} defines a value along the unstable equilibrium branch in the positive x-domain. This x^{*k} notation will be used for the remainder of this paper where k is the solution branch number starting from the lowest x-valued branch. A basin of repulsion also exists for $p < 0$ that begins at $x = x^{*2}$ and extends to $x = \infty$. Therefore the basin of attraction for the stable branch is $x \in (-\infty, x^{*2})$ and the basin of repulsion for the unstable branch is $x \in (x^{*2}, \infty)$. Any trajectories starting within the basin of attraction will converge to the stable branch over time while trajectories initiated in the basin of repulsion will diverge over time.

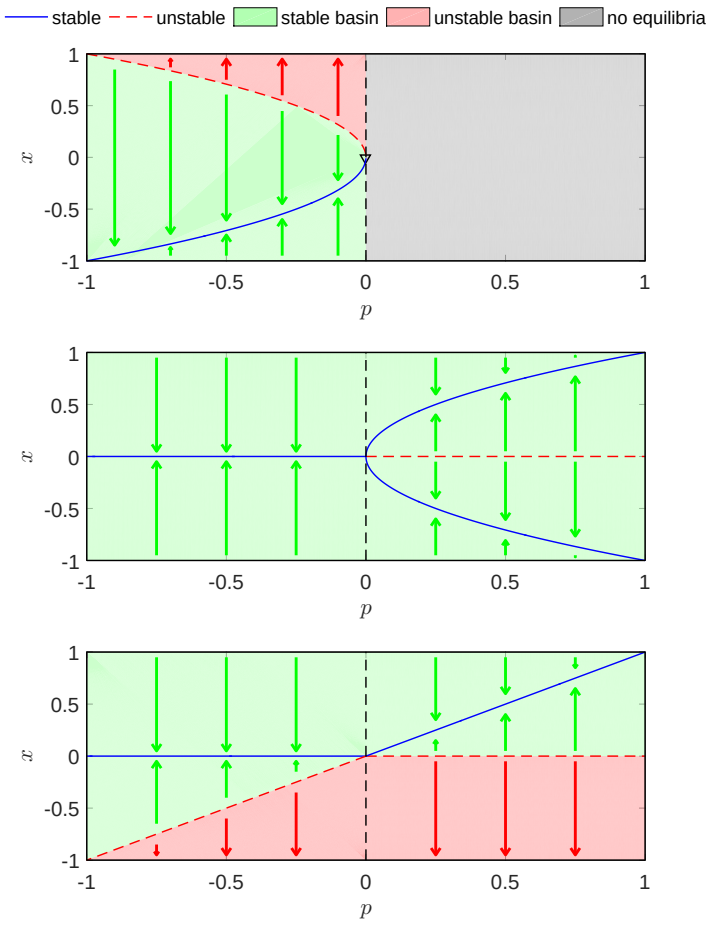

Figure 3.2: Bifurcation diagrams with identified basins of attraction and repulsion for example saddle node (top), pitchfork (center), and transcritical (bottom) bifurcations.

For the pitchfork system in Figure 3.2, when $p < 0$, only a stable branch exists with the basin of attraction $x \in (-\infty, \infty)$. When $p > 0$, trajectories converge to one of the two stable branches depending on which side of the unstable branch at $x^{*_2} = 0$ the initial conditions begin. Therefore, the basins of attraction for the two stable branches are $x \in (-\infty, 0)$ and $x \in (0, \infty)$. For the transcritical system in Figure 3.2, when $p < 0$, all x-values above the unstable $x^{*1} = p$ converge to $x^{*_2} = 0$ while values below will diverge. Therefore, the stable

basin is $x \in (x^{*_1}, \infty)$ and the unstable basin is $x \in (-\infty, x^{*_1})$. When $p > 0$, the stable and unstable basins are $x \in (0, \infty)$ and $x \in (-\infty, 0)$, respectively.

The automated process to determine the basins of attraction and repulsion from the numerical continuation results is as follows.

1. Beginning in the first p-domain, if no equilibrium exists, no basins are identified.

2. If only one equilibrium exists, its domain is $x \in (-\infty, \infty)$. The basin is attractive or repulsive depending on the stability of the equilibrium.

3. For multiple equilibria, the first equilibrium x^{*_1} has the lowest x-value. Since no equilibria exist below this, the lower limit of the basin for x^{*_1} is $x = -\infty$.

4. If x^{*_1} is unstable, then the upper limit of its basin is x^{*_1} since any trajectories starting above this value will trend towards the second equilibrium, x^{*_2}, which must be stable. If x^{*_1} is stable, then the upper limit of its basin is x^{*_2}, which is unstable. This method takes advantage of the fact that in one-dimensional systems, the stability of an equilibrium point is opposite the stability of adjacent equilibrium points [8]. For example, the stability ordering for a system with three equilibrium points is either stable-unstable-stable or unstable-stable-unstable.

5. Step 4 is repeated for each equilibrium until the last with the highest x-value, which has a basin with upper limit of $x = \infty$.

6. Steps 1-5 are repeated for each p-domain.

3.1.3 Step 3: Create Automaton

With the system equilibria and their basins defined within p-modes and sub-level x-modes, a hybrid stability automaton can be created. Two classes of automata can be automatically generated, as chosen by the user depending on the desired analysis goals. The first class covers

the *general* case, in which parameter p is not fixed to any particular value. Therefore, the system's equilibria and basins are represented as functions of p. Since the continuation output is solution data, but not the explicit functions, these functions are determined symbolically in a separate solver step. This imposes a restriction on general cases with system equations that may be too difficult for a symbolic solver. The second class, designated as *fixed-p*, is where the user specifies a value of interest for parameter p. Therefore, the system's equilibria and basins are represented as numerical values rather than functions and only the p-mode that contains the specified p is created. These values are determined directly from the continuation output data, therefore, avoiding the symbolic solving restrictions that exist in the general case. However, a trade-off exists between general and fixed-p representation. While more flexible in the types of systems it can handle, fixed-p representation captures a narrower view of the system's overall behavior since the resulting automaton is only relevant for the user-specified value of p.

GraphViz [14, 15, 16], a publicly available graph representation tool, is used to generate the automaton visualizations presented in this thesis. The input to GraphViz is a DOT file that contains all of the information to be represented. The mode data is therefore translated into DOT syntax and stored in a DOT file. This DOT file can then be directly processed with GraphViz to produce the automaton visualizations. The dot2tex [17] tool can be optionally applied to enable LaTeX-style formatting in the automata visualizations.

Parallel to the visualization, an executable function file is automatically generated. This function is structured as a list of if-statements defined by the transition switch conditions between p-modes and x-modes. Within each if-statement are relevant dynamics and stability information (i.e. whether the system diverges from an unstable equilibrium or converges to a stable equilibrium) that is displayed when the function is accessed.

3.2 Single-Parameter Examples

Example hybrid stability automata produced using the algorithm described in Section 3.1 are presented in the following sections.

3.2.1 Saddle Node Bifurcation

A general case hybrid stability automaton for the saddle node system, $\dot{x} = p + x^2$, is shown in Figure 3.3. As previously captured in bifurcation diagrams, the general automaton captures the lack of equilibria when $p > 0$ and the one stable, one unstable equilibria when $p < 0$. However, unlike bifurcation diagrams, the automaton explicitly defines the equilibria and basins of attraction/repulsion in each x-mode, thus identifying stable and unstable regions of the systems state-space. For example, it can be seen that when $p < 0$, a trajectory starting in the stable basin $x \in (-\infty, x^{*_2})$ will converge to $x^{*_1} = -\sqrt{-p}$ over time. A fixed-p case automaton where $p = -0.5$ is shown in Figure 3.4. Notice that only the $p < 0$ p-mode is depicted since the specified value lies within this domain. Also notice that the equilibrium and basin values are now numeric.

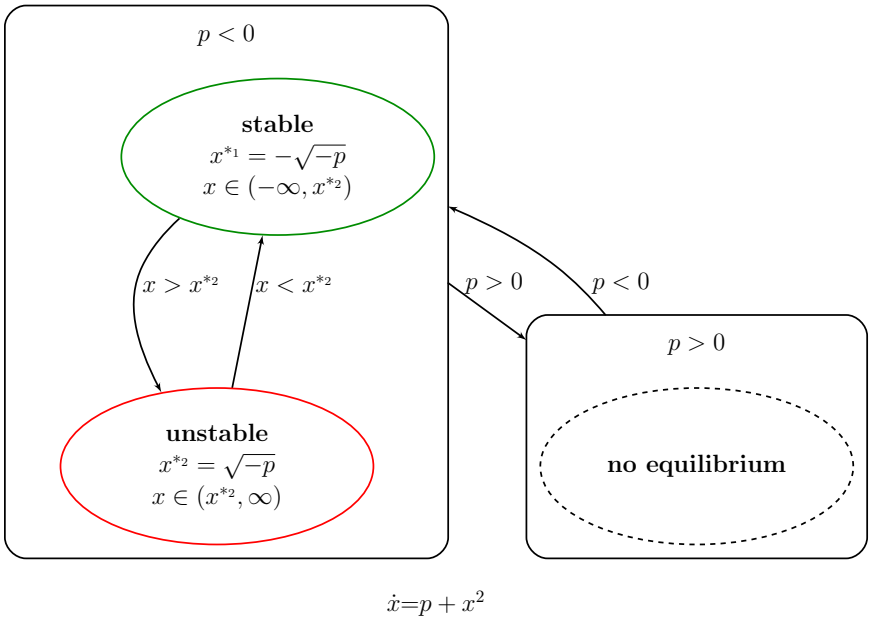

Figure 3.3: Hybrid stability automaton for saddle node bifurcation, general case.

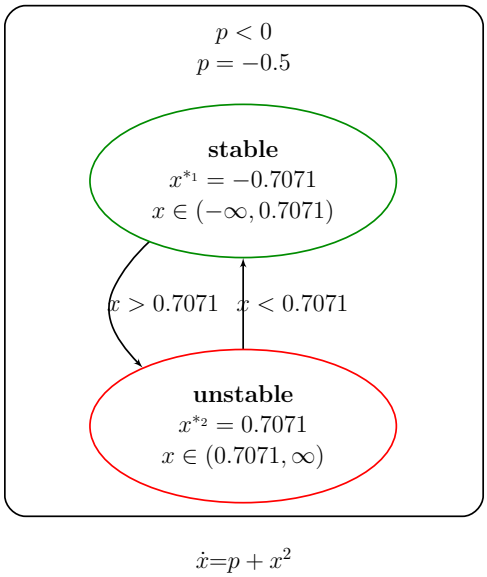

Figure 3.4: Hybrid stability automaton for saddle node bifurcation, fixed-p case ($p = -0.5$).

3.2.2 Pitchfork Bifurcation

Figure 3.5 shows example general and fixed-p hybrid stability automata for the pitchfork system, $\dot{x} = px - x^3$. In the fixed-p case p is specified as 0.5. The general automaton captures the single stable equilibrium when $p < 0$ and the two stable, one unstable equilibria when $p > 0$. The only unstable region of this system is $x = 0$ in the $p > 0$ p-mode, since values on either side of this will converge to one of the stable equilibria. Notice that in the saddle node case, trajectories either approach the one equilibrium or diverge to infinity based on whether x is above or below a single value. However, in the pitchfork case, the basins of attraction become more varied since multiple stable equilibria can exist simultaneously.

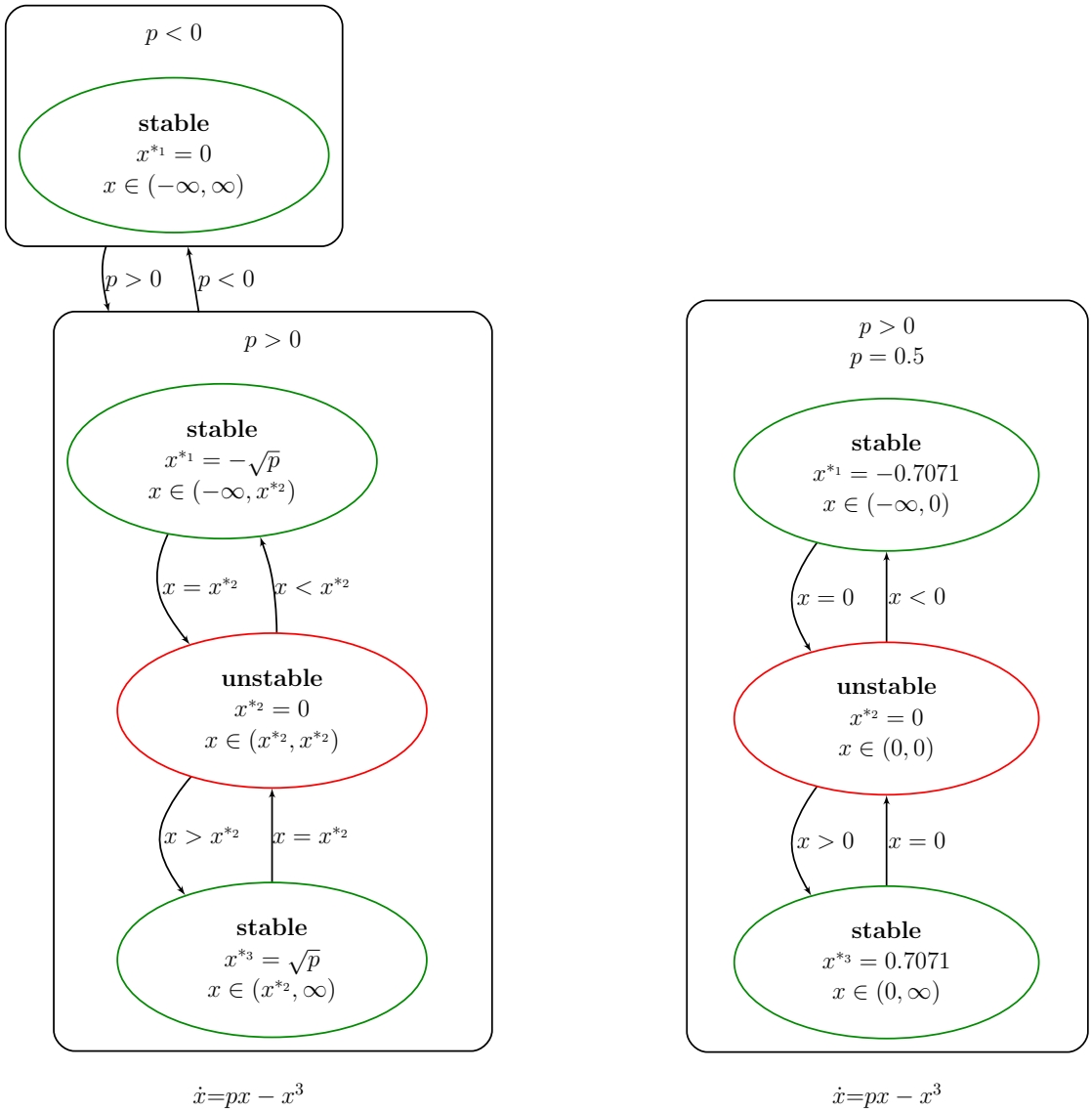

Figure 3.5: Hybrid stability automaton for pitchfork bifurcation, general case (left) and fixed-p case (right).

3.2.3 Transcritical Bifurcation

General and fixed-p hybrid stability automata for the transcritical system, $\dot{x} = px - x^2$, are shown in Figures 3.6 and 3.7. The characteristic switch in stability can be seen between the two equilibria, $x^* = 0$ and $x^* = p$.

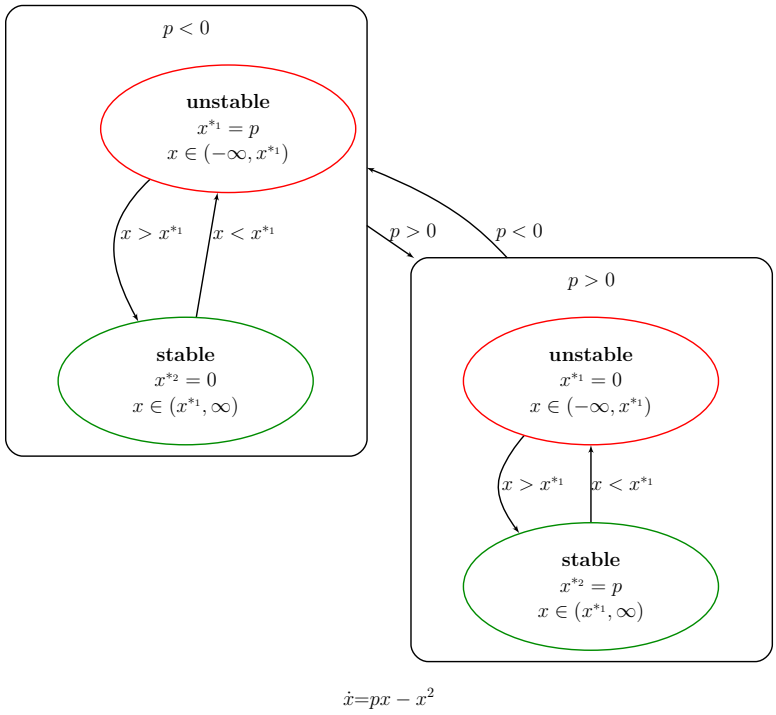

Figure 3.6: Hybrid stability automaton for transcritical bifurcation, general case.

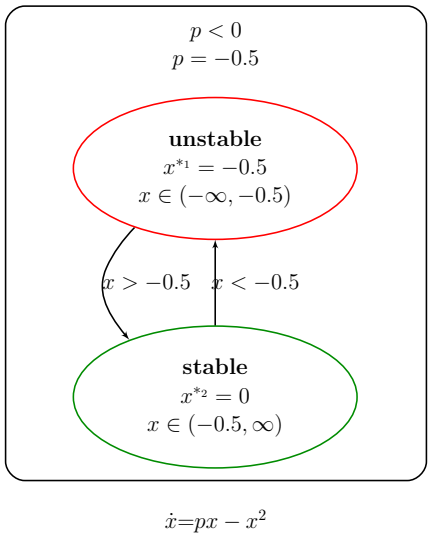

Figure 3.7: Hybrid stability automaton for transcritical bifurcation, fixed-p case ($p = -0.5$).

3.3 Multi-Parameter Process

The behavior of most physical systems is dependent on more than one parameter. These parameters are often analyzed in the context of operating conditions or trade studies. For example, it may be of interest to determine an aircraft's stability for different combinations of center of gravity and velocity. Therefore, it is valuable to enable multi-parameter analysis capabilities for hybrid stability automata. Since numerical continuation requires that the number of unknowns is greater than the number of equations by one, only a single parameter can be varied at a time in addition to the one state, x, for one-dimensional systems. Therefore, to apply the continuation analysis to multi-parameter systems, one parameter is activated while the remaining parameters must be set as constants. For example, activating p_1 while holding p_2,...,p_k constant. This process can then be repeated with different parameter designations of interest, such as varying p_2 while p_1, p_3,...,p_k are constant. The following steps describe the process to build a hybrid stability automaton for a one-dimensional, multi-parameter system. Aside from Step 1, this process is performed automatically.

1. The user defines the dynamic system equation, a *parameter designation matrix*, and initial points and limits for numerical continuation. The identification of the parameter to be varied and the values for the parameters to be held constant for each process are contained as entries within the parameter designation matrix, $M_{PD} \in \mathbb{R}^{m \times n}$, where m is the desired number of individual continuation processes and n is the number of different parameters.

2. The parameter designations from each row of M_{PD} are used to create subset dynamics from the original system dynamics.

3. Numerical continuation is performed on each subset dynamics.

4. The one-dimensional, one-parameter hybrid stability automaton process from Section 3.1 is applied to each subset. Data for p-modes and x-modes is stored within *top-modes*:

higher level modes of the automaton that are distinguished by the relevant parameter designations and subset dynamics.

5. A hybrid stability automaton visualization and an executable function are generated.

3.4 Multi-Parameter Example

To illustrate the multi-parameter process from Section 3.3, the system,

$$\dot{x} = p_1 + p_2 x + p_3 x^2 - p_4 x^3, \tag{3.2}$$

is investigated. Suppose it is of interest to analyze the system at three parameter designations,

$$\begin{aligned} 1: &\quad p_1 = p, \quad p_2 = 0, \quad p_3 = 0, \quad p_4 = -1 \\ 2: &\quad p_1 = 0, \quad p_2 = p, \quad p_3 = 0, \quad p_4 = 1 \\ 3: &\quad p_1 = 0, \quad p_2 = p, \quad p_3 = 1, \quad p_4 = 1, \end{aligned} \tag{3.3}$$

where p without a subscript denotes the active continuation parameter. The parameter designation matrix, M_{PD}, would then be

$$M_{PD} = \begin{bmatrix} p & 0 & 0 & -1 \\ 0 & p & 0 & 1 \\ 0 & p & 1 & 1 \end{bmatrix}. \tag{3.4}$$

These sets define three separate top-modes with the subset dynamics

$$\begin{aligned} 1: &\quad \dot{x} = p_1 + x^3 \\ 2: &\quad \dot{x} = p_2 x - x^3 \\ 3: &\quad \dot{x} = p_2 x + x^2 - x^3. \end{aligned} \tag{3.5}$$

Figures 3.8 and 3.9 show the resulting automata for general and example fixed-p cases.

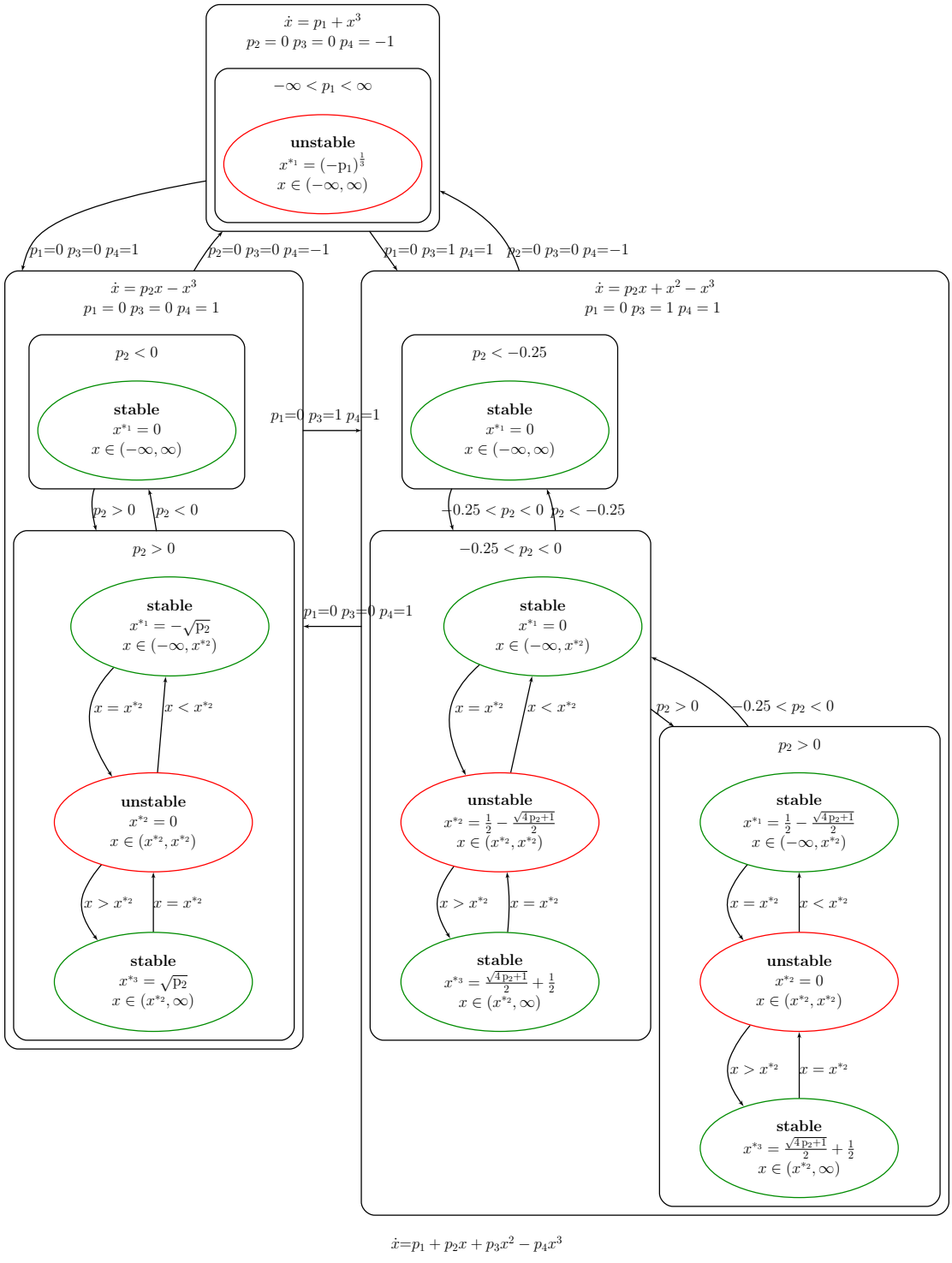

Figure 3.8: Hybrid stability automaton for a multi-parameter system, general case.

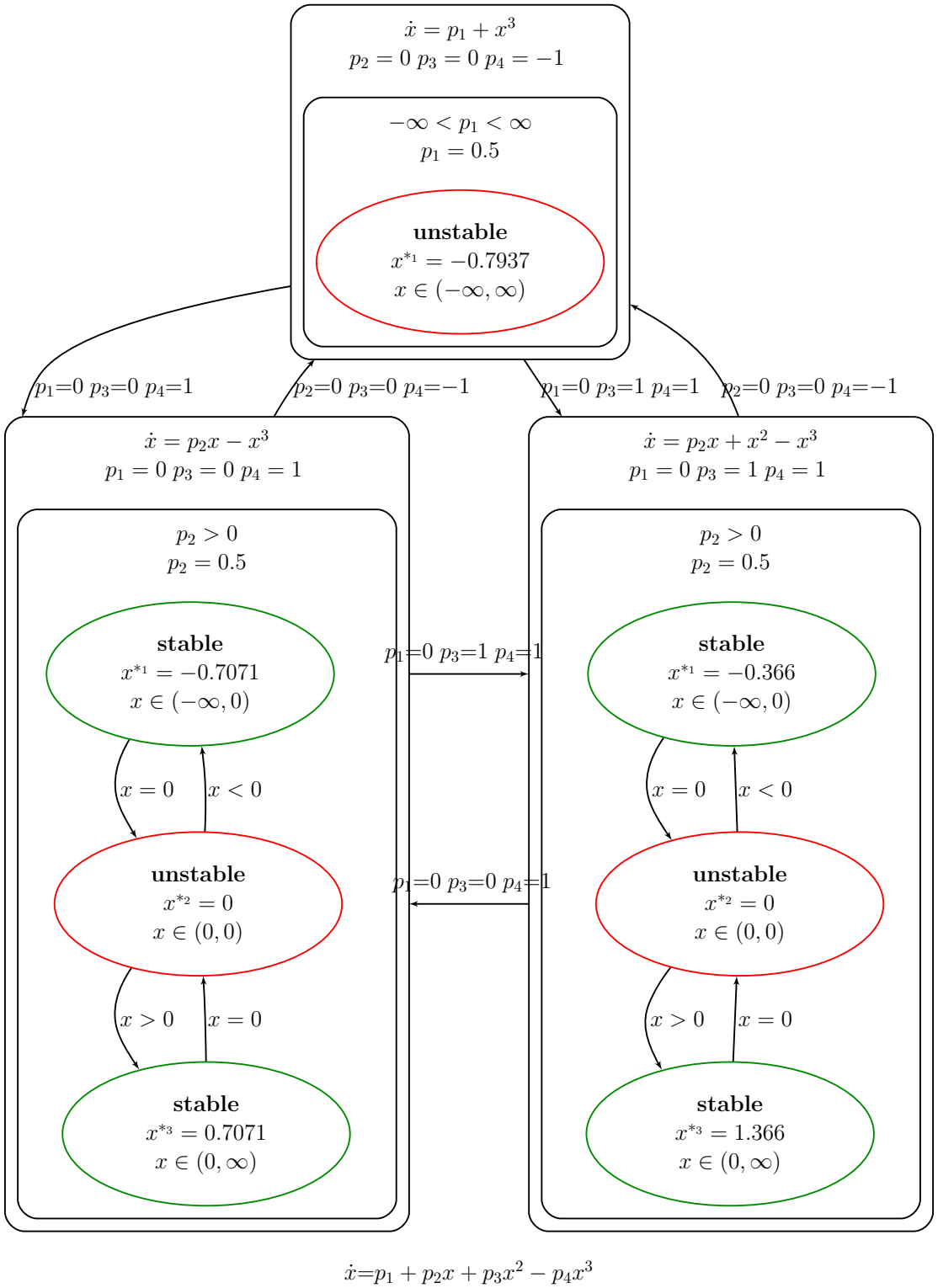

Figure 3.9: Hybrid stability automaton for a multi-parameter system, fixed-p case ($p = 0.5$).

Chapter 4

MULTI-DIMENSIONAL HYBRID STABILITY AUTOMATA

Producing a hybrid stability automaton for multi-dimensional systems, i.e. dynamical systems with two or more equations and states, is more challenging than producing a hybrid stability automaton for one-dimensional systems as discussed in Chapter 3. Numerical continuation is capable of handling multi-dimensional systems as long as the number of unknowns is one greater than the number of equations (e.g. a system with two state equations where the unknowns are the two states and one parameter). While this makes it possible to determine the equilibrium solutions and their stability properties, the guarantee that trajectories trend towards an adjacent stable equilibrium used to determine basins for one-dimensional systems no longer applies in two or more dimensions. Therefore, alternative means of finding the basins are required for multi-dimensional systems since they can no longer be deduced automatically from the numerical continuation output.

The multi-dimensional hybrid stability automaton creation process is separated into two steps. In the first step, numerical continuation is applied and the automaton is generated, similar to the one-dimensional process. However, the automaton mode definitions and switch conditions are now defined generically where D_{sxx} is a basin of attraction for a stable equilibrium with xx as some two-digit identifier. For multi-dimensional systems, the identification of a basin of attraction may not provide insight on the stability of regions outside of the basin. For example, if only a portion of a stable basin is identified, assumptions cannot be made on whether a point outside of the known portion is stable or unstable. To capture this uncertainty, the "unknown" mode type was incorporated into the automaton visualization with the domain D_u. This D_u contains all conditions outside of known basins of attraction, including any unstable equilibria. In the second step, a separate process is performed to

determine the basins of attraction for each D_{sxx} identified from the first step. Determining basins of attraction for stable equilibria has been the focus of much research and different techniques may be selected depending on the user's desired analysis goals.

This two-step process allows the automaton visualization to be generated automatically and independent of the basin identification. Therefore, if the user is only interested in finding the equilibria and their stability, the second step can be skipped entirely. If the basins are of interest, then the second step can be performed without impacting the results of the first step. An executable function is generated in either case, although it can only display stability information if basins have been identified. The following section details the basin identification step, which can only be applied to fixed-p hybrid stability automata (i.e. cannot be applied to the general case).

4.1 Basin Identification

The multi-dimensional hybrid stability automata structure allows for the incorporation of different basin identification methods. The work presented in this thesis focuses on the application of three techniques: Monte Carlo simulation, numerical continuation, and sum of squares (SOS). Table 4.1 summarizes some of the advantages and disadvantages of each of these methods, and the following sections provide more detail.

4.1.1 Monte Carlo Simulation

The *Monte Carlo method* describes computational techniques that acquire system properties by repeated numerical simulation [18, 19]. The sets of conditions to be simulated are often determined via random sampling. Euler and Runge-Kutta numerical integration methods [20] are commonly used to produce the simulations. Since the Monte Carlo method can be applied to any system that can be simulated, it has been one of the most widely used analysis techniques in fields such as safety V&V, particularly for systems that are not easily handled by other approaches. However, since the Monte Carlo method requires iterative simulation, it is typically computationally expensive, especially for systems with large state-

Table 4.1: Advantages and disadvantages of three basin identification methods.

Method	**Advantages**	**Disadvantages**
Monte Carlo simulation	• Simple and easy to automate • Works on most systems	• Does not guarantee lack of holes • Computationally intensive
Numerical continuation	• Computationally efficient	• Does not guarantee lack of holes • Requires some user interaction
Sum of squares	• Guarantees lack of holes	• Typically non-exhaustive • Works on limited types of systems • Requires laborious user interaction

spaces. Additionally, there is no guarantee that unsampled values will exhibit the same behavior as nearby sampled values (i.e. behavioral "holes" may exist). Increasing the number of samples can increase the level of confidence (but not guarantee) that no holes exist, at the cost of computation time.

To use the Monte Carlo method for basin of attraction identification, initial conditions for the system states are sampled and simulated. This sampling can be based on randomization or a fixed step size. For each simulation, if the state values at the final time are within some zero tolerance to a stable equilibrium, then the initial conditions are considered to be within the basin of attraction for that equilibrium. A MATLAB script that employs a Monte Carlo method was created to determine the basins of attraction for stable equilibria, D_{sxx}, for multi-dimensional hybrid stability automata using the `ode45` solver. The user-defined inputs to this script include:

- Hybrid stability automaton mode information – This is the data output from the automaton generation (first step in the multi-dimensional process), which is used to

identify the stable equilibria.

- The limits of interest for each of the system states – This is used as the outer boundary of the basin identification process. Values outside of this boundary are considered to be within the unknown domain D_u.

- Selection of random or fixed step sampling

- Sample size definition – If random sampling is selected, this is the total number of samples, uniformly distributed within the boundary limits. If fixed step sampling is selected, this is the number of evenly spaced segments for each state.

- The simulation time span

4.1.2 Numerical Continuation

Similar to how numerical continuation is used to approximate an equilibrium solution branch, it can be used to "trace" a basin of attraction by altering the continuation problem definition to search for values of x_0 that solve $f(x_0, t_f) = x^*$, where x^* is a stable equilibrium. This is done by incorporating a numerical integration step into the continuation process so that approximations can be made for final values x_f based on a given time span and initial conditions x_0. For example, the fourth order Runge-Kutta method,

$$\begin{aligned} x_{n+1} &= x_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \\ k_1 &= f\left(t_n, x_n\right), \\ k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right), \\ k_3 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2\right), \\ k_4 &= f\left(t_n + h, x_n + hk_3\right), \end{aligned} \tag{4.1}$$

where h is a step size and f is the set of state equations, can be incorporated into the continuation problem structure. Continuation is applied to equation (4.1), which accesses the state equations at each iteration, instead of being applied directly to the state equations themselves. However, the inclusion of this numerical integration step requires additional variables since it is desired to vary the initial conditions and the states. This also requires additional equations to satisfy the numerical continuation requirement that the number of variables is one greater than the number of equations. To ensure that the solution satisfies the basin of attraction condition at each continuation step, the constraint equation

$$G = x\left(t_f\right) - x^* - \epsilon = 0, \tag{4.2}$$

where ϵ is some zero tolerance, is enforced such that the final state values match the stable equilibrium values. Using equations (4.1) and (4.2), with the states x, initial conditions x_0, and zero tolerance ϵ as variables, numerical continuation can be applied. The resulting values of x_0 trace the basin of attraction.

Since this continuation-based basin identification method requires computation only along the basin's boundary, it can provide a significant reduction in process time compared to the Monte Carlo method, which requires computation at iterative points throughout the state-space. However, some intuition is required in choosing an effective initial point to start the continuation process in order to obtain the desired results. For example, if the initial point is within the basin instead of on the boundary, continuation may find an internal solution curve rather than the outer boundary. Therefore, the continuation-based method is more difficult to automate than the Monte Carlo method. Additionally, similar to the Monte Carlo method, there is no guarantee that holes do not exist within the basin boundary identified via numerical continuation.

4.1.3 Sum of Squares

Leveraging sum of squares programming to determine a basin of attraction for a stable equilibrium has been the focus of much research, [21, 22, 23] for example. A multivariate polynomial $F(x)$ is considered to be a *sum of squares* if there exists polynomials $f_1(x)$, ..., $f_m(x)$ such that

$$F(x) = \sum_{i=1}^{m} f_i^2(x). \tag{4.3}$$

It can be seen that such a real-valued function $F(x)$ is nonnegative for all values of x. The general approach for an SOS-based method of basin determination is to find a Lyapunov function that guarantees stability within its bounds without any behavioral holes. The Lyapunov stability theorem states that a dynamical system $\dot{x} = f(x)$ with an equilibrium at the origin is stable if a positive definite function $V(x)$ exists such that $\dot{V}(x)$ is nonpositive along system trajectories [24]. SOS can be used to find a $V(x)$ Lyapunov function by enforcing certain constraints while solving for $F(x)$ in equation (4.3). For example, in a three-state system, the constraints

$$\begin{aligned} V - \epsilon\left(x_1^2 + x_2^2 + x_3^2\right) &\geq 0, \\ -\frac{\partial V}{\partial x_1}\dot{x}_1 - \frac{\partial V}{\partial x_2}\dot{x}_2 - \frac{\partial V}{\partial x_3}\dot{x}_3 &\geq 0, \end{aligned} \tag{4.4}$$

where ϵ is some positive constant, may be applied to satisfy the Lyapunov stability conditions. It is important to note that equation (4.4) only defines candidate constraints, which may not be usable for every three-state system. SOSTOOLS [25] is a popular MATLAB tool that converts sum of squares programs into semidefinite programs and employs SeDuMi [26], a semidefinite programming solver, to find the desired $V(x)$ given the system state equations and appropriate constraints.

The SOS output Lyapunov function is typically quadratic. Therefore, this method will only capture a portion of a non-quadratic basin of attraction. Some research efforts have

focused on constructing composite polynomial Lyapunov functions to extend the shape flexibility of the captured region such as in [23], but requires increased user interaction. In general, the SOS basin determination method requires laborious user interaction, particularly for the selection of appropriate constraint equations, and is therefore difficult to automate. Additionally, the application of SOS techniques is restricted to certain polynomial systems.

4.2 Multi-Dimensional Examples

The following sections provide hybrid stability automata demonstrations for multi-dimensional systems. Single-parameter and multi-parameter cases are presented.

4.2.1 Multi-Dimensional, Single-Parameter Example

For this example, the multi-dimensional, single-parameter system,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -px_2 - x_1^3 \\ -x_2^2 - x_1 \end{bmatrix}, \tag{4.5}$$

is investigated. For reference, the bifurcation diagrams for x_1 vs. p, x_2 vs. p, and x_2 vs. x_1 that result from applying numerical continuation to equation (4.5) are shown in Figure 4.1. The output of the multi-dimensional hybrid stability automata visualization tool for the general and example fixed-p cases of this system are shown in Figure 4.2 and Figure 4.3, respectively.

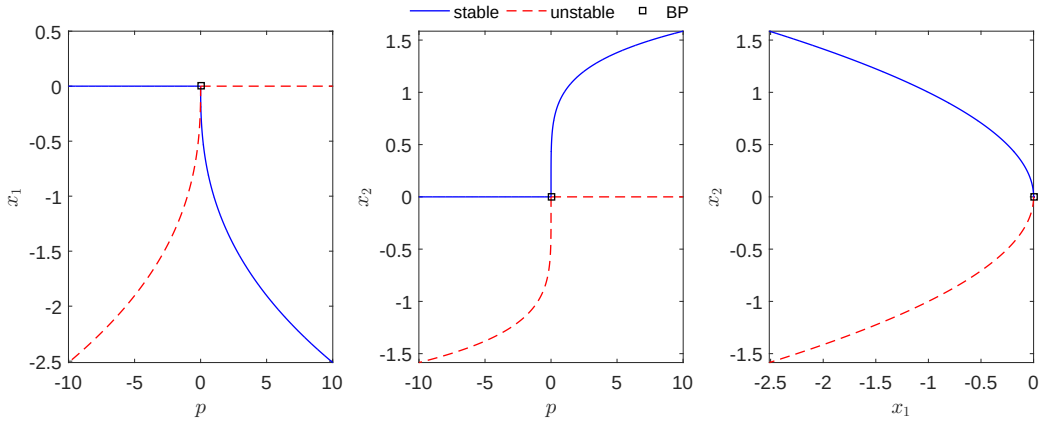

Figure 4.1: Bifurcation diagrams for a multi-dimensional, single-parameter system.

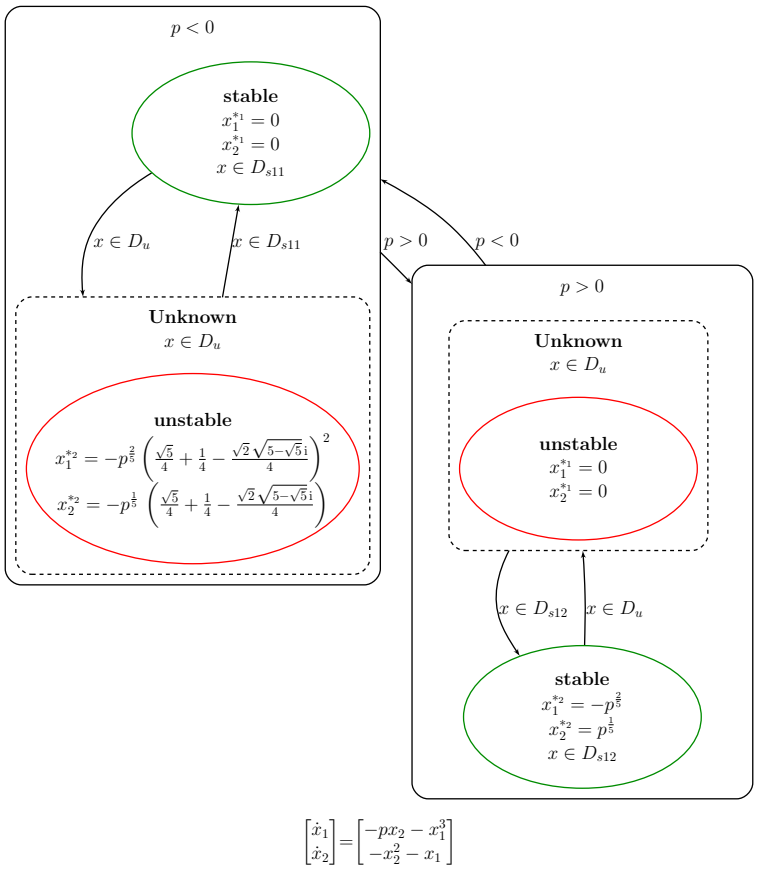

Figure 4.2: Example multi-dimensional hybrid stability automaton, general case.

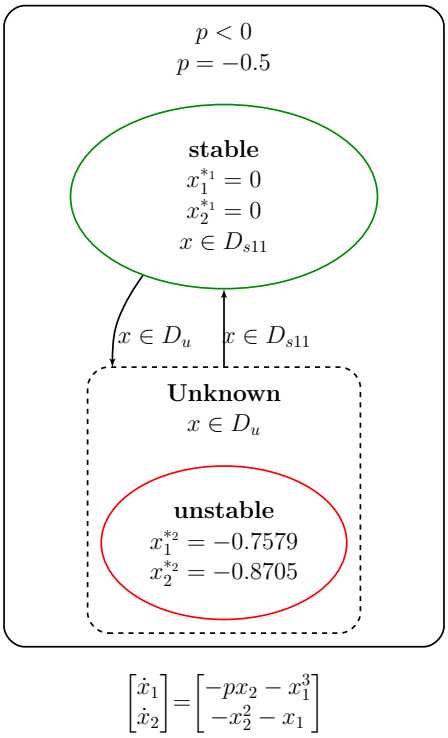

Figure 4.3: Example multi-dimensional hybrid stability automaton, fixed-p case.

Basin Identification

The Monte Carlo technique discussed in Section 4.1.1 for determining the basin of attraction is applied to the fixed-p automaton shown in Figure 4.3. In this case, the only stable equilibrium is at $x_1^* = 0$, $x_2^* = 0$, with the basin of attraction D_{s11}. The resulting simulated trajectories are shown in Figure 4.4, which uses fixed step sampling, limits $x_1, x_2 \in [-3 \quad 3]$, and a final time of 20. A phase plot of this system is also shown in Figure 4.4, in which it can be seen that the stable equilibrium is a spiral and the unstable equilibrium is a saddle point. Figure 4.5 shows the basin determination for D_{s11} using the Monte Carlo method, where initial conditions within the basin are depicted with green points. This information is then used to construct the basin of attraction shown in Figure 4.6.

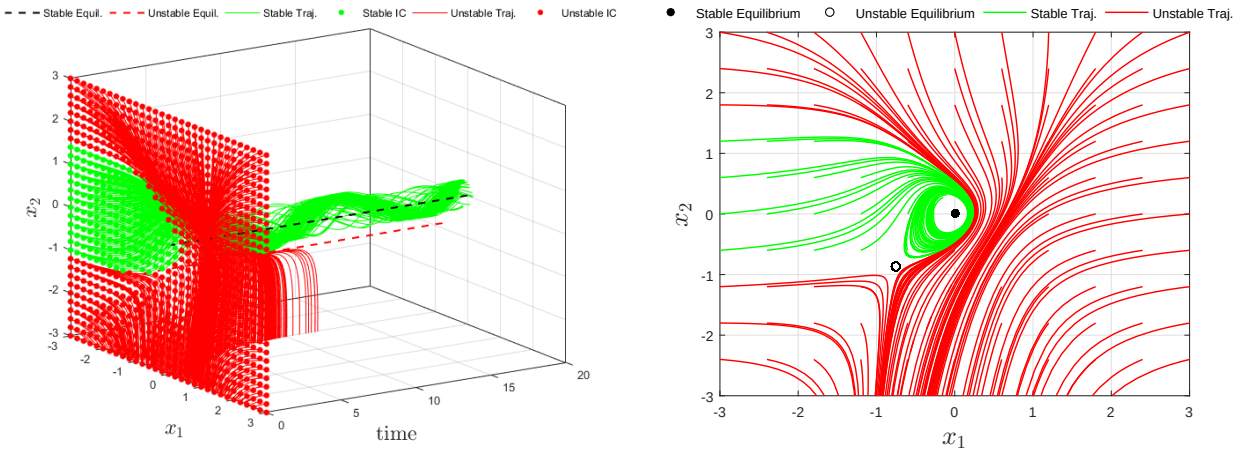

Figure 4.4: Monte Carlo method simulated trajectories (left) and phase plot (right).

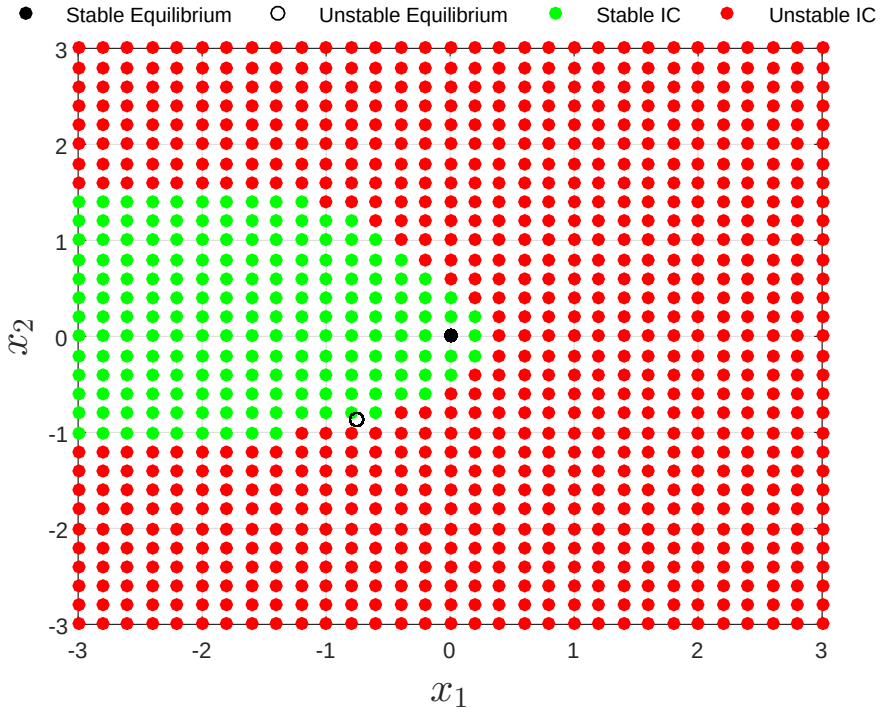

Figure 4.5: Monte Carlo method applied to find the D_{s11} basin of attraction.

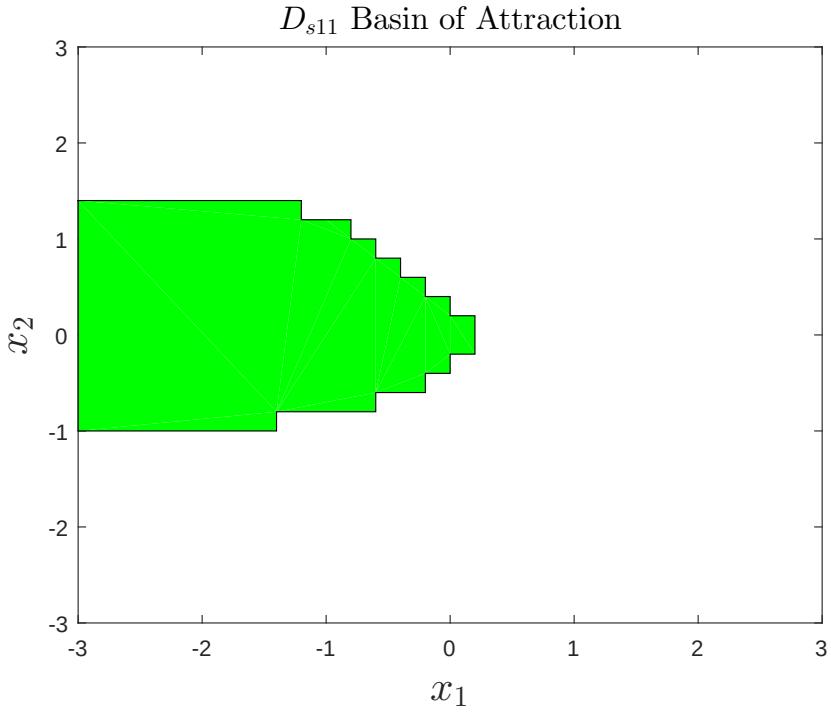

Figure 4.6: The D_{s11} basin of attraction, Monte Carlo method.

4.2.2 Multi-Dimensional, Multi-Parameter Example

The system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -p_1 x_2 + p_2 x_1 - x_1^3 \\ -p_3 \left(x_2^2 + x_1\right) + p_2 x_1 - p_4 x_1^2 \end{bmatrix} \tag{4.6}$$

is used to demonstrate a multi-dimensional, multi-parameter case with parameter designation

$$PD = \begin{bmatrix} p & 0 & 1 & 0 \\ 1 & p & 0 & 1 \end{bmatrix}. \tag{4.7}$$

The automated multi-parameter automaton generation follows the process outlined in Section 3.3, but with the multi-dimensional algorithm incorporated. The resulting hybrid stability automata for the general and example fixed-p cases are shown in Figures 4.7 and 4.8, respectively.

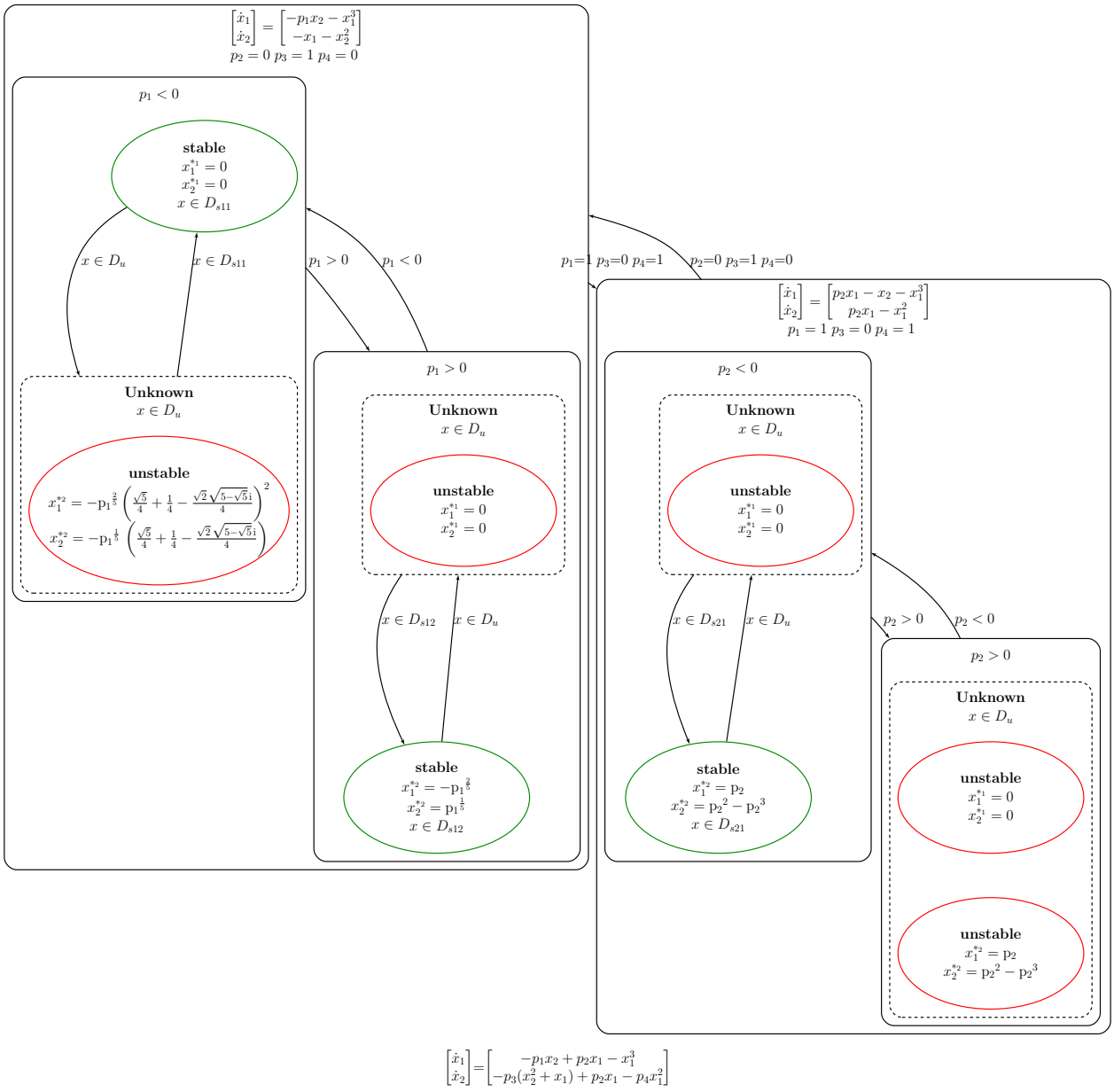

Figure 4.7: Example multi-dimensional, multi-parameter hybrid stability automaton, general case.

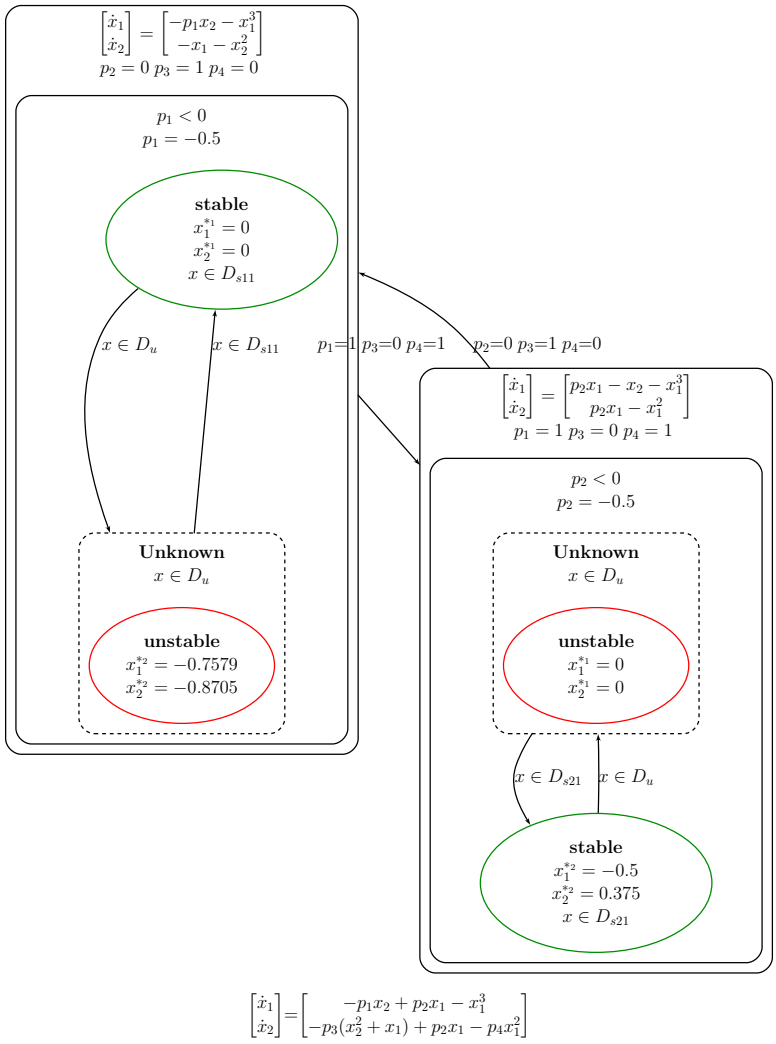

Figure 4.8: Example multi-dimensional, multi-parameter hybrid stability automaton, fixed-p.

Basin Determination

The fixed-p automaton, shown in Figure 4.8, contains two stable equilibria. The first top-mode, depicted on the left side of Figure 4.8, is equivalent to the single-parameter example discussed in the Section 4.2.1. Therefore the basin of attraction for the stable equilibrium at $x_1^* = 0$, $x_2^* = 0$, D_{s11}, is the same as what was shown in Figure 4.6. The second top-mode, which is defined by the dynamics

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} px_1 - x_2 - x_1^3 \\ px_1 - x_1^2 \end{bmatrix} \tag{4.8}$$

with $p = -0.5$, contains a stable equilibrium with basin of attraction D_{s21}. Both the Monte Carlo and numerical continuation basin determination methods are applied to find D_{s21}. The results are shown in Figure 4.9. It can be seen that the numerical continuation method traces a basin boundary similar to what is found via repeated simulation. Using this data, the D_{s21} basins of attraction determined by Monte Carlo and numerical continuation methods are shown in Figures 4.10 and 4.11, respectively. The computational advantage of numerical continuation is observed in its 27-second process time versus the 369-second process time for the Monte Carlo method (performed on an Intel Core i7-6700 3.40GHz processor with 16GB of RAM). However, the continuation method required additional user intuition in the selection of an initial point known to be near the basin's boundary.

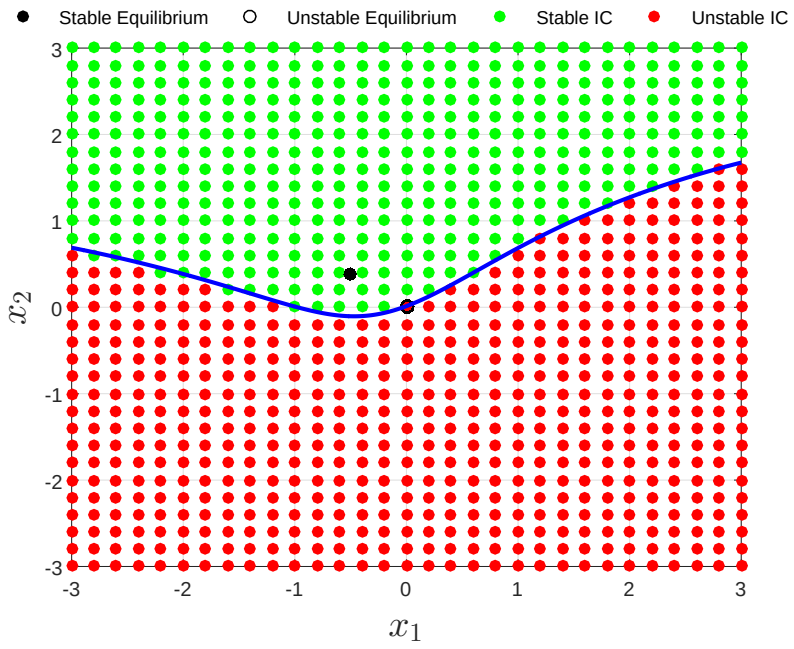

Figure 4.9: Monte Carlo and numerical continuation (solid blue line) results for finding D_{s21}.

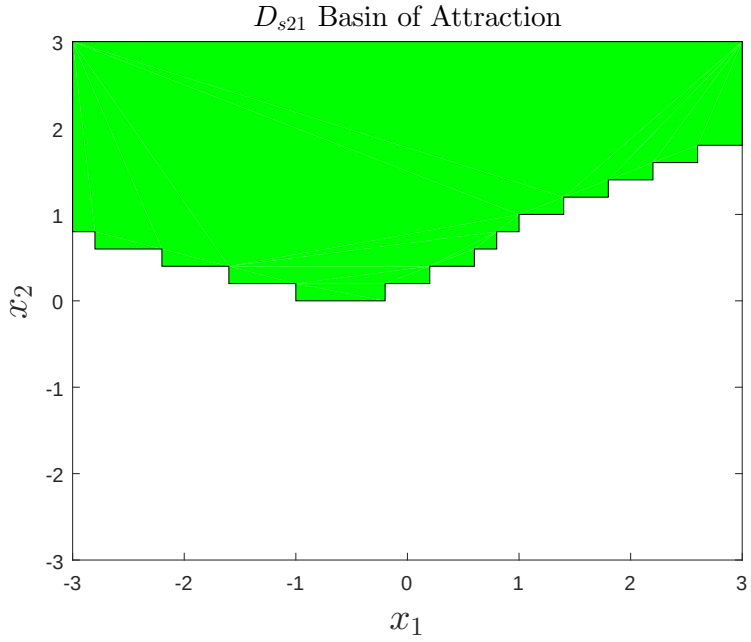

Figure 4.10: The D_{s21} basin of attraction, Monte Carlo method.

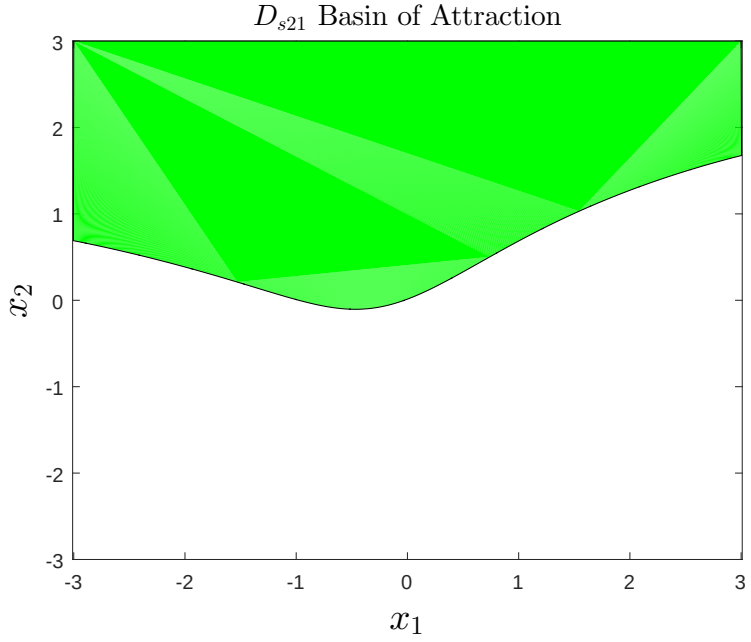

Figure 4.11: The D_{s21} basin of attraction, numerical continuation.

Chapter 5
APPLICATION EXAMPLES

In this chapter, the one-dimensional and multi-dimensional processes for creating hybrid stability automata from Chapters 3 and 4, respectively, are used to facilitate demonstration analyses on the space shuttle atmospheric reentry dynamics. Taken from [27], the equations of motion for the unpowered space shuttle during reentry are

$$\dot{h} = v \sin\gamma, \tag{5.1}$$

$$\dot{\phi} = \frac{v}{R_e + h} \cos\gamma \sin\psi / \cos\theta, \tag{5.2}$$

$$\dot{\theta} = \frac{v}{R_e + h} \cos\gamma \cos\psi, \tag{5.3}$$

$$\dot{v} = -\frac{D}{m} - g \sin\gamma, \tag{5.4}$$

$$\dot{\gamma} = \frac{L}{mv} \cos\beta + \cos\gamma \left(\frac{v}{R_e + h} - \frac{g}{v} \right), \tag{5.5}$$

$$\dot{\psi} = \frac{L}{mv \cos\gamma} \sin\beta + \frac{v}{R_e + h} \cos\gamma \sin\psi \sin\theta, \tag{5.6}$$

where altitude h, longitude ϕ, latitude θ, velocity v, flight path angle γ, and azimuth ψ are the system states. Drag D, lift L, and gravity g are defined using the equations

$$\begin{aligned} D &= \frac{1}{2} c_D S \rho v^2, \qquad & L &= \frac{1}{2} c_L S \rho v^2, \\ c_L &= a_0 + a_1 \alpha, \qquad & c_D &= b_0 + b_1 \alpha + b_2 \alpha^2, \\ g &= \frac{\mu}{(R_e + h)^2}, \qquad & \rho &= \rho_0 e^{-h/h_r}. \end{aligned} \tag{5.7}$$

The vehicle controls for the reentry scenario are angle of attack α and bank angle β. Parameters specific to shuttle reentry are listed in Table 5.1.

Table 5.1: Space shuttle reentry parameters.

μ	0.1407654×10^{17} ft/s^2	ρ_0	0.002378 lbs/ft^3
R_e	20902900 ft	h_r	23800 ft
S	2690 ft^2	m	6309.44 lbs
a_0	-0.20704	a_1	0.029244
b_0	0.07854	b_1	-0.61592×10^{-2}
b_2	0.621408×10^{-3}		

5.1 One-Dimensional Shuttle Reentry Analysis

In this section, a one-dimensional hybrid stability automaton is used for a space shuttle reentry analysis. For this demonstration, stable and unstable flight path angles are determined for certain velocities at different reentry conditions. Therefore, equation (5.5) is used with γ identified as the state and forward velocity v as the varied parameter. The reentry conditions of interest are angles of attack 10°, 20°, and 30° at an altitude of 200,000 ft and no bank angle. With system parameters $P = [v \quad \alpha \quad \beta \quad h]$, the parameter designation matrix for this analysis is

$$M_{PD} = \begin{bmatrix} p & 10 & 0 & 200000 \\ p & 20 & 0 & 200000 \\ p & 30 & 0 & 200000 \end{bmatrix}. \tag{5.8}$$

By defining $x = \gamma$ and substituting equation (5.7) in equation (5.5), the system to be analyzed becomes

$$\dot{x} = \frac{1}{2m}\left(a_0 + a_1\alpha\right) S\rho_0 e^{-h/h_r} v\cos\beta + \cos x\left(\frac{v}{R_e + h} - \frac{\mu}{v\left(R_e + h\right)^2}\right). \tag{5.9}$$

Applying the three parameter designations from equation (5.8) and substituting the parameters from Table 5.1 into equation (5.9) results in the dynamic subsets,

$$f(x,p) = \dot{x} = 9.7 \times 10^{-9} p + \cos x \left(\frac{p}{21102900} - \frac{31.62}{p} \right), \tag{5.10}$$

$$f(x,p) = \dot{x} = 4.3 \times 10^{-8} p + \cos x \left(\frac{p}{21102900} - \frac{31.62}{p} \right), \tag{5.11}$$

$$f(x,p) = \dot{x} = 7.6 \times 10^{-8} p + \cos x \left(\frac{p}{21102900} - \frac{31.62}{p} \right). \tag{5.12}$$

Using the multi-parameter process, which automatically performs the substitutions to obtain equations (5.10–5.12) from equation (5.9), a hybrid stability automaton for a fixed-p case where v = 17,000 ft/s was generated and is shown in Figure 5.1. Notice that in the first two top-modes, the vehicle dynamics contain a stable equilibrium in the negative domain of γ and an unstable equilibrium in the positive domain. A positive flight path angle cannot be maintained since the vehicle is unpowered and falling during atmospheric reentry, and will trend towards the stable, negative γ^*. Interestingly for this system, since γ is circular, trajectories starting in the unstable basin will eventually converge to the stable equilibrium. For example, in the first top-mode, a trajectory initiated at $\gamma = 90°$ trends away from $\gamma^* = 81.02$ but will eventually reach the stable equilibrium at $\gamma^* = 278.98°$, which is equivalent to the stable $\gamma^* = -81.02°$ depicted in Figure 5.1. Regardless, this is unsafe behavior since it describes the vehicle rotating around until reaching the stable equilibrium. In the third top-mode, no equilibrium exists for the specified velocity value of 17,000 ft/s. In this scenario, γ increases over time without convergence, which is physically interpreted as continuous spinning of the vehicle. Since this is undesirable behavior, the "no equilibrium" region is considered unsafe. The varying system behaviors can be seen in Figure 5.2, which shows example trajectories within the three top-modes simulated with the MATLAB `ode45` solver on the executable hybrid stability automaton function.

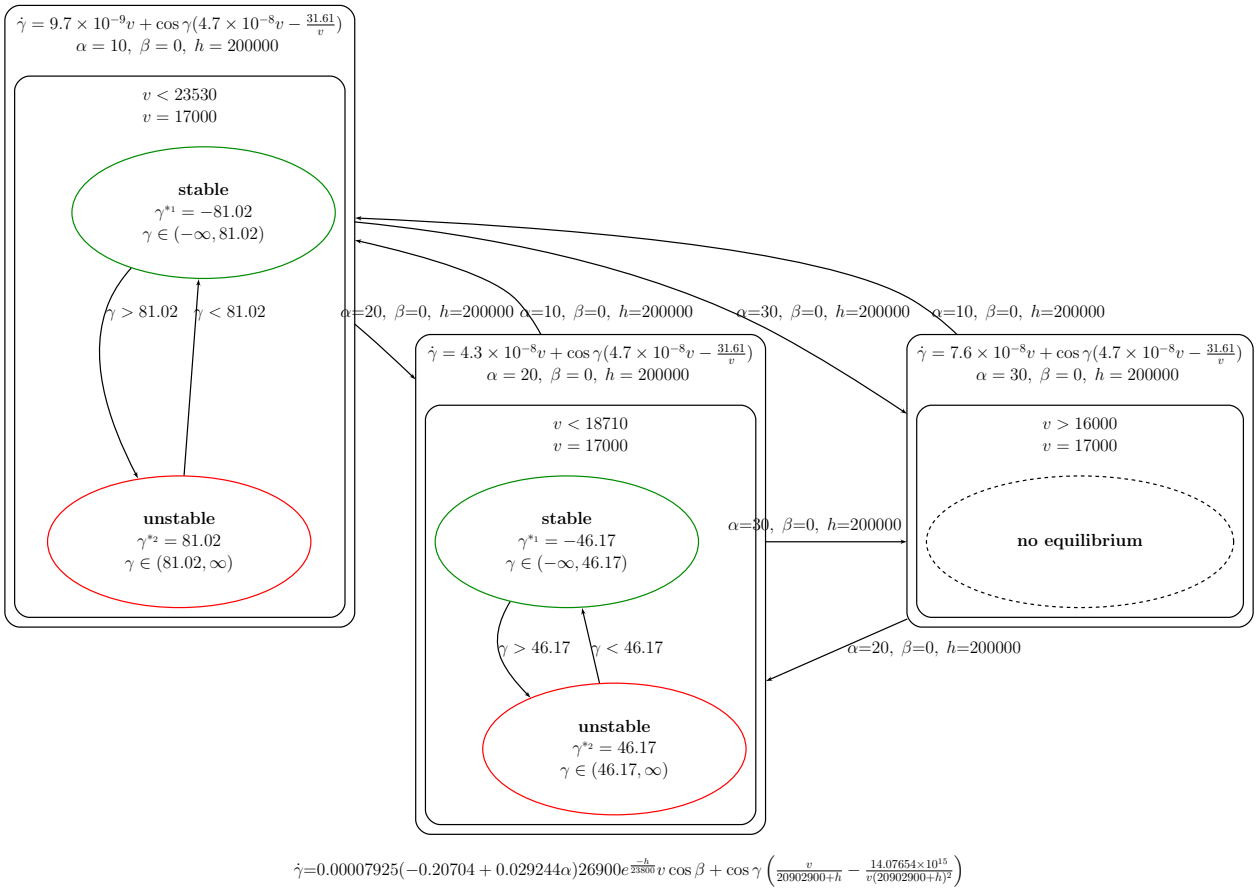

Figure 5.1: 1D, multi-parameter hybrid stability automaton for a shuttle reentry analysis.

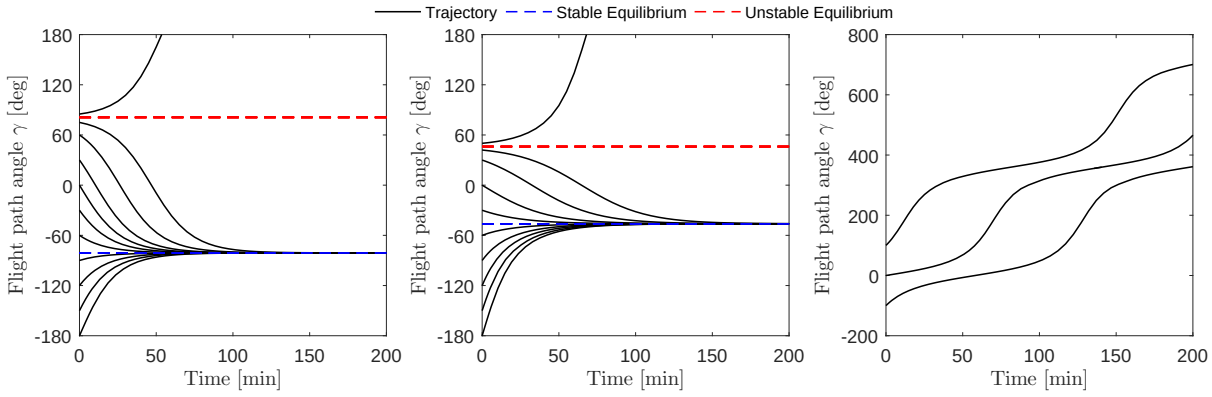

Figure 5.2: Example trajectories for the three top-modes of the 1D shuttle automaton.

5.2 Multi-Dimensional Shuttle Reentry Analysis

In this section, a multi-dimensional hybrid stability automaton is used for a space shuttle reentry analysis. For this demonstration, stable and unstable sets of flight path angles and velocities are determined for different reentry conditions. Therefore, equations (5.5) and (5.4) are used with γ, v identified as the states and angle of attack α as the varied parameter. The reentry conditions of interest are altitudes of 190,000 ft and 175,000 ft with no bank angle. With system parameters $P = [\alpha \quad \beta \quad h]$, the parameter designation matrix is

$$M_{PD} = \begin{bmatrix} p & 0 & 190000 \\ p & 0 & 175000 \end{bmatrix}. \tag{5.13}$$

By defining $x_1 = \gamma$, $x_2 = v$, and substituting equation (5.7), the system to be analyzed becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2m}\left(a_0 + a_1\alpha\right) S\rho_0 e^{-h/h_r} x_2 \cos\beta + \cos x_1 \left(\frac{x_2}{R_e+h} - \frac{\mu}{x_2(R_e+h)^2}\right) \\ -\frac{1}{2m}\left(b_0 + b_1\alpha + b_2\alpha^2\right) S\rho_0 e^{-h/h_r} x_2^2 - \frac{\mu}{(R_e+h)^2}\sin x_1 \end{bmatrix}. \tag{5.14}$$

Applying the parameter designations from equation (5.13) and substituting the parameters from Table 5.1 into equation (5.14) results in the dynamic subsets,

$$f(x,p) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1.73 \times 10^{-9}\left(-20.70 + 2.92p\right) x_2 + \cos x_1 \left(\frac{x_2}{21092900} - \frac{31.64}{x_2}\right) \\ -1.73 \times 10^{-11}\left(785.4 - 61.59p + 6.21p^2\right) x_2^2 - 31.64 \sin x_1 \end{bmatrix}, \tag{5.15}$$

$$f(x,p) = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 3.25 \times 10^{-9}\left(-20.70 + 2.92p\right) x_2 + \cos x_1 \left(\frac{x_2}{21077900} - \frac{31.68}{x_2}\right) \\ -3.25 \times 10^{-11}\left(785.4 - 61.59p + 6.21p^2\right) x_2^2 - 31.68 \sin x_1 \end{bmatrix}. \tag{5.16}$$

Using the multi-parameter process, which automatically performs the substitutions to obtain equations (5.15) and (5.16) from equation (5.14), a hybrid stability automaton for a fixed-p case where $\alpha = 10$ degrees was generated and is shown in Figure 5.3. Notice that in the first top-mode, which is defined by a higher altitude than the second top-mode, a stable

and unstable equilibrium exists for $\alpha = 10$. However, the unstable equilibrium disappears at the lower-altitude top-mode, leaving only the stable equilibrium. Also notice that the stable equilibrium to which the vehicle trends is lower in velocity at the lower altitude. This behavior is expected since reentry vehicles slow down as they lose altitude due to increased air resistance in lower regions of the atmosphere with greater air density.

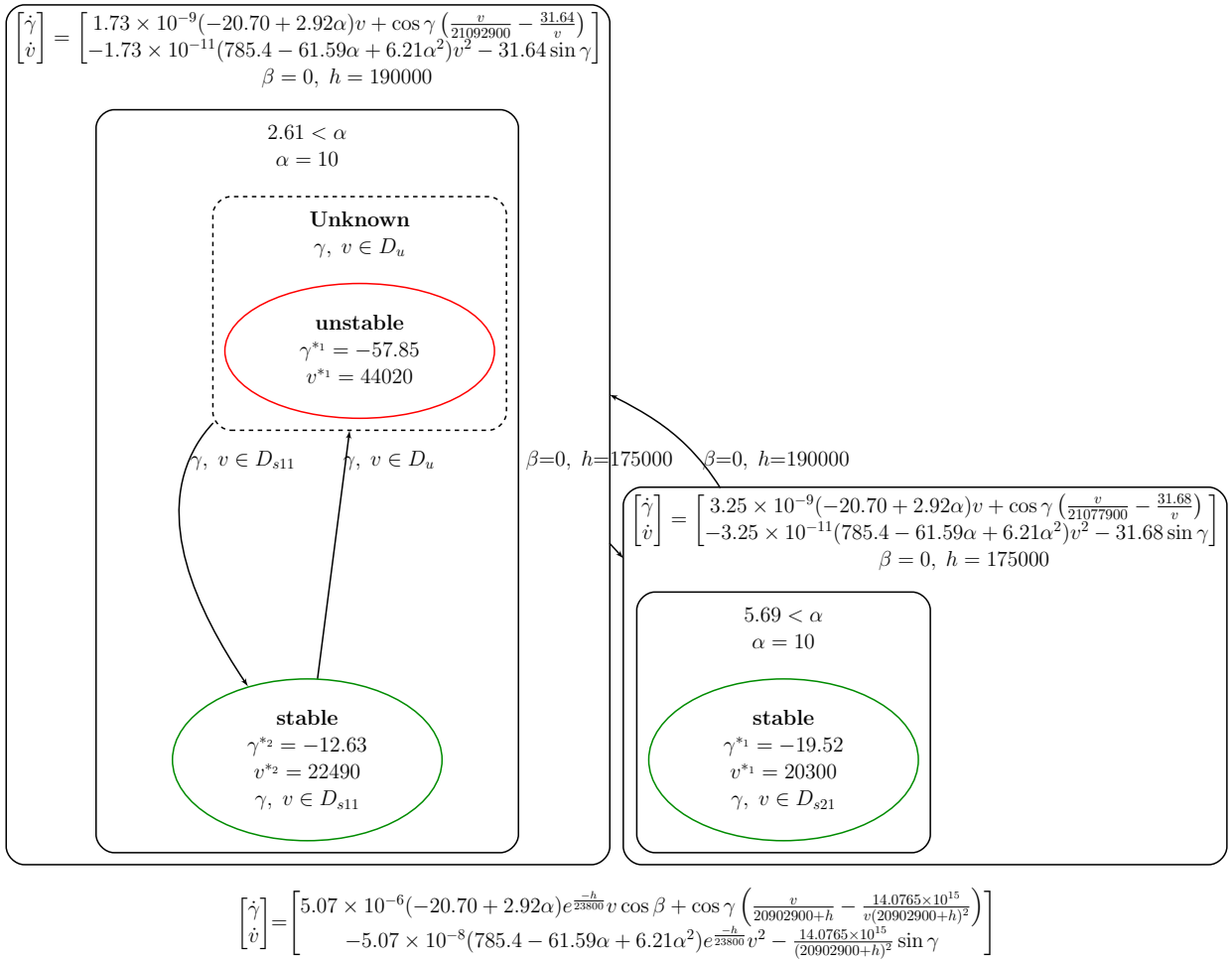

Figure 5.3: Multi-dimensional, multi-parameter hybrid stability automaton for a shuttle reentry analysis.

The automaton for this analysis contains two stable equilibria. To determine their basins of attraction, D_{s11} and D_{s21}, the Monte Carlo simulation method previously discussed in Section 4.1.1 was applied. Fixed step sampling was used with the state limits of $v \in [0 \quad 50000]$ ft/s and $\gamma \in [-120 \quad 90]$ degrees. The sampling results for D_{s11} and D_{s21} are shown in Figure 5.4. The approximated D_{s11} and D_{s21} basins of attraction generated from the sampling results are shown in Figure 5.5. The majority of operating conditions within the specified state limits belong to the basins of attraction of the stable equilibria. Even trajectories initiated near the unstable equilibrium in the first top-mode belong to D_{s11}, thus will eventually trend towards the stable equilibrium. An unstable region exists in both modes for flight path angles above 50 degrees. A velocity of zero is also unstable, since the vehicle needs forward motion to stabilize, although this scenario is unrealistic during reentry. Notice that the D_{s11} basin of attraction approximation shown in Figure 5.5 encapsulates the unstable equilibrium. This is an example of the Monte Carlo method's inability to guarantee a lack of holes, since the fixed step samples did not find the unstable equilibrium value.

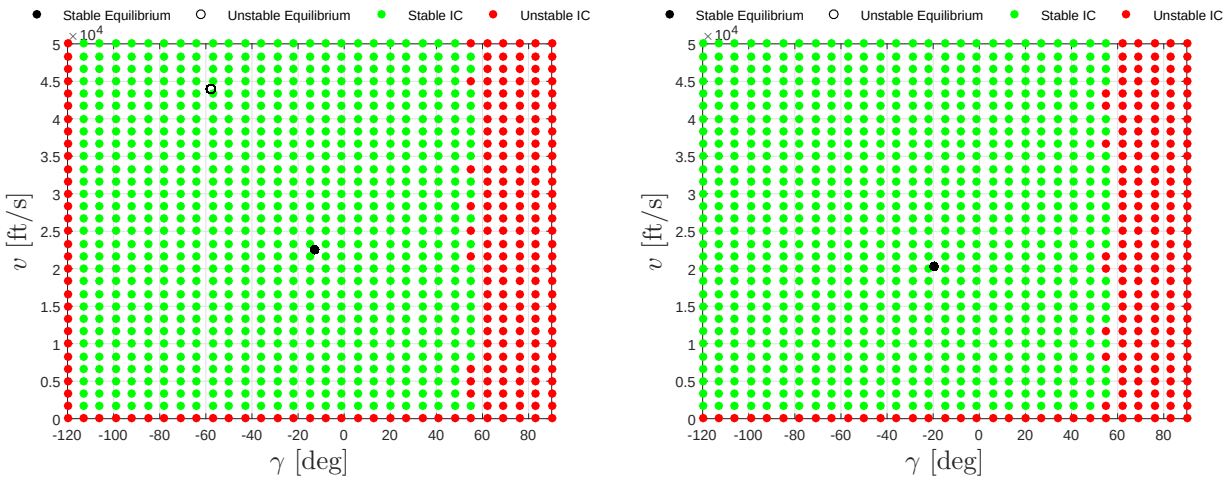

Figure 5.4: Monte Carlo results to find D_{s11} (left) and D_{s21} (right) basins of attraction.

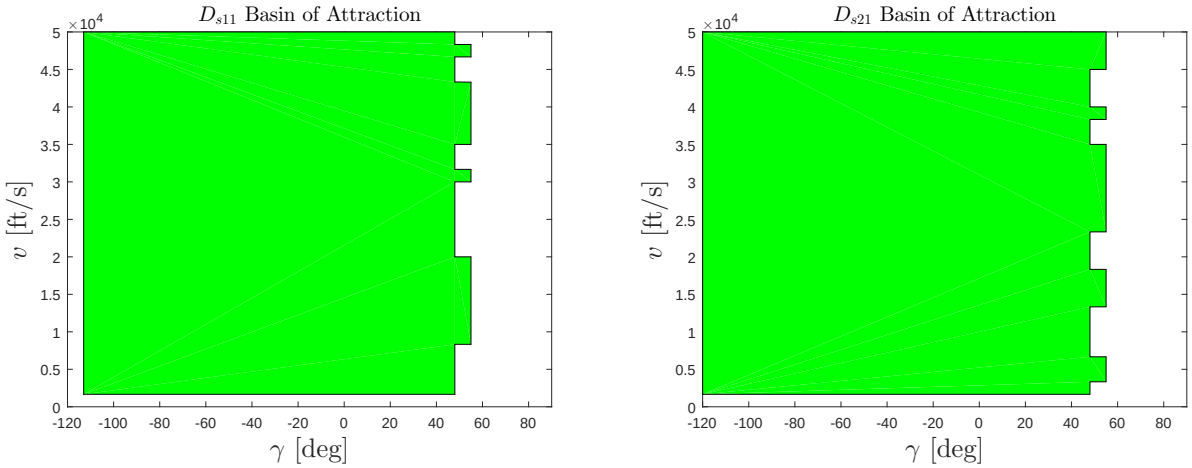

Figure 5.5: The D_{s11} (left) and D_{s21} (right) basins of attraction, Monte Carlo method.

Chapter 6

CONCLUSION

This thesis presented the concept of hybrid stability automata as a way to model a system that captures valuable stability information. Equilibria and their stabilities are obtained by employing a numerical continuation-based bifurcation analysis on the system's dynamics. The basins of attraction and repulsion are determined from the numerical continuation results and used to identify stable and unstable regions of the state-space. The processes to create hybrid stability automata for one-dimensional and multi-dimensional systems were detailed. Code was developed to apply these processes for creating automaton visualizations and executable functions. For one-dimensional systems, the basins of attraction are explicitly defined within the automaton visualization, providing a direct description of safe operating regions. For multi-dimensional systems, the basins are generically defined within the automaton and a separate basin identification step is applied to approximate the safe regions. The separation of this step allows for flexibility in choosing the desired method for determining the basins of attraction without affecting the automaton visualization. Of the options discussed, the Monte Carlo method was automated and applied due to its simplicity. However, this is not ideal since simulation repetition compromises the computational efficiency advantages targeted by hybrid stability automata. Promising results were produced for basin "tracing" via numerical continuation, and it is recommended that this method's potential benefits over existing basin determination techniques be explored in future research.

Hybrid stability automaton examples for prototypical systems were presented. Safety analyses were demonstrated for both one-dimensional and multi-dimensional automata on the space shuttle reentry dynamics. An analysis using a hybrid stability automaton can provide significant gains in process efficiency over existing methods. The construction of a

general or fixed-p automaton accesses the existing mode data from the underlying numerical continuation analysis, therefore only one computational process is required for a set of dynamics. Comparatively, an analysis using repeated simulations requires a computational process for each set of initial states being tested – a disadvantage that can compound since a large number of simulations may be required to capture the same information contained in one hybrid stability automaton.

Overall, modeling systems as hybrid stability automata was shown to be a useful way to capture stable and unstable regions. Since this directly correlates to safe and unsafe operating conditions, hybrid stability automata are conducive to safety verification. Additionally, since system modes are separated via logic-based switches, this type of representation has the potential to be integrated with model checkers and theorem provers. Therefore recommended future work includes this integration to enhance the capabilities of existing V&V tools to create a more advanced safety verification framework.

BIBLIOGRAPHY

[1] Clarke, E. M., Emerson, E. A., and Sistla, A. P., "Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications," *Association for Computing Machinery Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 8, No. 2, 1986, pp. 244–263.

[2] Grumberg, O. and Veith, H., *25 Years of Model Checking: History, Achievements, Perspectives*, Vol. 5000, Springer, 2008.

[3] Clarke, E. M., Grumberg, O., and Long, D. E., "Model Checking and Abstraction," *Association for Computing Machinery Transactions on Programming Languages and Systems (TOPLAS)*, Vol. 16, No. 5, 1994, pp. 1512–1542.

[4] Merz, S., "Model Checking: A Tutorial Overview," *Modeling and Verification of Parallel Processes*, Springer, 2001, pp. 3–38.

[5] Hasan, O., *Formalized Probability Theory and Applications using Theorem Proving*, IGI Global, 2015.

[6] Harrison, J., *Handbook of Practical Logic and Automated Reasoning*, Cambridge University Press, 2009.

[7] Allgower, E. L. and Georg, K., *Introduction to Numerical Continuation methods*, Society for Industrial and Applied Mathematics, 2003.

[8] Strogatz, S. H., *Nonlinear Dynamics and Chaos: with Applications to Physics, Biology, Chemistry, and Engineering*, Westview Press, 2014.

[9] Platzer, A., *Logical analysis of hybrid systems: proving theorems for complex dynamics*, Springer Science & Business Media, 2010.

[10] Spetzler, M. and Narang-Siddarth, A., "Continuation Analysis of Nonlinear Systems with Equality Constraints on States, Parameters, and Eigenvalues," *AIAA Guidance, Navigation, and Control Conference*, 2015.

[11] Spetzler, M. G. and Narang-Siddarth, A., "Increased Functionality of Continuation-Based Nonlinear System Analysis," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 6, 2016, pp. 1206–1222.

[12] Dawkins, P., "Differential Equations - Notes: Phase Plane," [online] http://tutorial.math.lamar.edu/Classes/DE/PhasePlane.aspx.

[13] Henzinger, T. A., "The Theory of Hybrid Automata," *Verification of Digital and Hybrid Systems*, Springer, 2000, pp. 265–292.

[14] Gansner, E. R. and North, S. C., "An Open Graph Visualization System and its Applications to Software Engineering," *Software Practice and Experience*, Vol. 30, No. 11, 2000, pp. 1203–1233.

[15] Gansner, E., Koutsofios, E., and North, S., "Drawing Graphs with DOT," Technical report, AT&T Research, 2006, [online] http://www.graphviz.org/Documentation/dotguide.pdf.

[16] Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., and Woodhull, G., "Graphviz and Dynagraph Static and Dynamic Graph Drawing Tools," *Springer Graph Drawing Software*, 2004, pp. 127–148.

[17] Fauske, K. M., "dot2tex - a GraphViz to LaTeX converter," [online] https://dot2tex.readthedocs.io/en/latest/.

[18] Rubinstein, R. Y. and Kroese, D. P., *Simulation and the Monte Carlo Method*, John Wiley & Sons, 2016.

[19] Robert, C. P., *Monte Carlo Methods*, Wiley Online Library, 2004.

[20] Islam, M. A., "A Comparative Study on Numerical Solutions of Initial Value Problems (IVP) for Ordinary Differential Equations (ODE) with Euler and Runge Kutta Methods," *American Journal of Computational Mathematics*, Vol. 5, No. 3, 2015, pp. 393.

[21] Parrilo, P. A., *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*, Ph.D. thesis, California Institute of Technology, 2000.

[22] Papachristodoulou, A. and Prajna, S., "On the construction of Lyapunov functions using the sum of squares decomposition," *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, Vol. 3, IEEE, 2002, pp. 3482–3487.

[23] Tan, W. and Packard, A., "Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming," *IEEE Transactions on Automatic Control*, Vol. 53, No. 2, 2008, pp. 565–571.

[24] Khalil, H. K., *Noninear Systems*, Prentice-Hall, New Jersey, 1996.

[25] Papachristodoulou, A., Anderson, J., Valmorbida, G., Prajna, S., Seiler, P., and Parrilo, P. A., *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2013, [online] http://www.eng.ox.ac.uk/control/sostools.

[26] Sturm, J. F., "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization methods and software*, Vol. 11, No. 1-4, 1999, pp. 625–653.

[27] Betts, J. and Kolmanovsky, I., "Practical Methods for Optimal Control using Nonlinear Programming," *Applied Mechanics Reviews*, Vol. 55, 2002, pp. B68.