# DESIGN AND VERIFICATION OF A SAFE AUTONOMOUS SATELLITE RENDEZVOUS MANEUVER

BY

NICOLE CHAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Associate Professor Sayan Mitra

# ABSTRACT

A fundamental maneuver in autonomous space operations is known as rendezvous, where an active spacecraft navigates towards and maneuvers within close proximity of a free-flying passive spacecraft. Any mistake during autonomous space flight can be extremely costly, yet these systems are difficult to verify due to limitations of testing spacecraft. In this thesis, we present a benchmark model formulation for the rendezvous mission, two control solutions to achieve this mission, and a rigorous method to demonstrate that the resulting system's behavior remains safe.

The benchmark model provides both a nonlinear description of the spacecraft's motion and a linearized approximation, and the mission objectives, or equivalently, our set of safety properties. We present a set of control solutions, which includes a hybrid, or switched, version of linear quadratic regulator (LQR)—a fundamental approach in the theory of optimal control for linear systems. We formulate a novel hybrid controller, dubbed state-dependent linear quadratic (SDLQ) control, which extends the former controller in a way that may improve its ability to generate only safe trajectories. With these choices of dynamical models and controllers, we obtain a collection of models that are shown to robustly achieve safety properties of interest using a suite of hybrid verification tools. We utilize several existing tools, each developed for different classes of hybrid models, and we implement a new tool called SDVTool which improves upon one of the former tools. We present experimental results that illustrate the promise (and ongoing challenges) of this approach; that is, applying a class of simulation-based verification algorithms to our proposed set of benchmark models and safety requirements to design and rigorously demonstrate safety of the autonomous satellite maneuver. We will demonstrate both successful, safe scenarios and incomplete or unsafe examples.

*To my family, for their unwavering love.*

# ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my advisor Professor Sayan Mitra, without whom this work would not have been possible. By supporting and mentoring me through my time at Illinois, he has provided me with an incredible opportunity to grow, learn, and explore both within and outside of research. His patience and passion for pursuing deeper, innovative problems in technology inspires the leader I strive to become someday. I would like to thank Dr. Richard Scott Erwin for his mentorship during my time at the Air Force Research Laboratory's Space Scholars program, which contributed greatly to the work presented in this thesis.

I am grateful for my labmates and friends Ritwika Ghosh, Chuchu Fan, Bolun Qi, Matthew Potok, and most notably, Hussein Sibai, who has spent countless hours and energy beyond working together to comfort and guide me through these last few years. I am also thankful for Carol Wisniewski for her genuine dedication to her students and making CSL a welcoming space.

I am deeply grateful and blessed to have lifelong friends who refuse to let thousands of miles disrupt their constant support and love: Jenifer Wong, Kayla Niu, and Wellington Lee; and to have found family in my roommates: Kelly Zhao and Sarah Garrow. I must also thank Eric Figueroa—who both inspired and enabled me to pursue this path.

I thank my family in Tucson; they never fail to provide perspective on what is most valuable in life and what I am incredibly blessed to have: home.

Finally, I would like to acknowledge the generous support of the Air Force Research Lab and the National Science Foundation.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

The National Research Council published a report on NASA's roadmaps for 2011-2021 and listed relative guidance algorithms (or the autonomous control of spacecraft to perform fundamental maneuvers on-orbit) as one of the highest-priority technologies for enabling next-generation space missions [1]. The model of an autonomous rendezvous, proximity operations, and docking (ARPOD) mission presented in [2] captures the essence of relative guidance problems in a generic, reusable scenario. This scenario and its components are essential in applications such as on-orbit transportation of personnel [3], resupply for personnel [4], assembly of space stations [5], and repair and refueling of spacecraft [6].

In particular, *rendezvous* refers to navigating a primary spacecraft towards a secondary free-flying object (which we fix to be a satellite) and maneuvering within close proximity of the target without collision. This presents a challenging constrained guidance problem. An overview of general state-of-the-art solutions to this particular problem is given in [7]. While such approaches have led to some successes, recent examples suggest they are not fully mature technologies nor guaranteed to behave safely. For example, NASA's DART spacecraft was designed to rendezvous with the MUBLCOM satellite [8]. In 2005, approximately 11 hours into a 24-hour mission, DART's propellant supply depleted due to excessive use of thrusters, and it began a mission abort sequence. In the process it collided with MUBLCOM; it met only 11 of 27 mission objectives, rendering the loss of a $110 million project. Several other incidents in [9] highlight the consequences of failures in space applications and demonstrate the need for more rigorous testing before deployment.

Formal verification is a means of providing proofs of safe behavior given the model of a system. Hybrid models, which capture systems with both discrete and continuous variables and dynamics, provide an expressive framework for describing complex cyber-physical systems. Verification of hybrid systems presents a big challenge, but recent advances in this area motivate and enable its application to the problem of autonomous spacecraft navigation. Although formal verification has played an important role in design and safety analysis of spacecraft hardware and software (see, for example, [10] and the references therein), it has not yet been used for model-based design and system-level verification and validation.

## 1.2 Thesis Overview

We present a suite of hybrid models for the rendezvous portion of the ARPOD mission [2] that can serve as benchmarks for verification tools and as building-blocks for more complex missions. These models include two controllers we design to achieve rendezvous, as well as mission objectives (or safety properties) such as physical limitations of the spacecraft, its interaction with the target, and the concept of *passive safety*—the ability to maintain both propulsion-free and collision-free motion in the event of a system failure. We verify these benchmark problems with a series of hybrid verification tools, including well-established (C2E2 and SpaceEx) and recently released (DryVR) tools and one introduced from analyzing the ARPOD model (SDVTool).

These rendezvous mission problems can be configured to any combination of the following choices:

- nonlinear (NLin) orbital dynamics, or linearized (Lin) dynamics using the Clohessy-Wiltshire-Hill (CWH) equations [11];
- switched LQR (SwLQ) with state-dependent switching signal, or state-dependent linear quadratic (SDLQ) control with time-dependent switching signal;
- *passive abort* (Pass) or without. The *passive abort* scheme adds a mode to the hybrid control strategy, in which the spacecraft will shut off its thrusters when it detects any system failure, thus motivating the need to check for passive safety.

The set of constraints for the rendezvous mission lends itself naturally to two partitions of the state-space, each with a different subset of active constraints. As a consequence, the first controller is chosen to be a switched, two-mode control scheme with statically computed linear quadratic regulator (LQR) in each mode. We refer to this as the switched LQR (SwLQ) scheme.

Unlike traditional optimal control solutions, we propose a cost function with dynamic, state-dependent coefficients. The state-dependent LQ (SDLQ) scheme extends switched LQR so that the quadratic cost function is periodically re-evaluated for new coefficients and solved.

We test the benchmarks with state-of-the-art hybrid verification tools: C2E2 [12, 13] and SpaceEx [14]. C2E2 can handle nonlinear models, and both tools are capable of handling linear models containing continuous dynamics described by ordinary differential equations (ODEs). The SDLQ configuration does not admit a closed-form control solution, so we additionally utilize a data-driven tool called DryVR [15]—capable of analyzing black-box models—to verify SDLQ models.

C2E2 proved to be the more versatile tool in our experiments, yet it did not perform well particularly for models that include the passive abort mode (Pass). To address this issue, we apply a different algorithm for computing reachable states of the system—a necessary component behind each of these verification tools—and implement this in a new software tool we call SDVTool.

We successfully show that several of the models do behave safely within the rendezvous mission constraints, under a set of mission parameters (i.e. time horizon and initial conditions) that we define. Overall, we believe that our results and approaches establish the feasibility of system-level verification of autonomous space operations, and they provide a foundation for the analysis of more sophisticated maneuvers in the future.

## 1.3 Related Work

There are few academic works on system-level verification of autonomous spacecraft. A survey of general verification approaches and how they may apply to small satellite systems is presented in [16]. Architecture and Analysis Design Language (AADL) and verification and validation (V&V) over

AADL models for satellite systems have been reported in [17].

A feasibility study for applying formal verification of autonomous satellite maneuvers is presented in [18]. That approach relied on creating rectangular abstractions (dynamics of the form $\dot{x} \in [a, b]$) of the satellites dynamics through hybridization and verification using PHAVer [19] and SpaceEx [14]. The generated abstract models have simple dynamics but hundreds of locations, and also the analysis is necessarily conservative. In contrast, the approaches presented in this thesis work directly with the linear (nonlinear) hybrid dynamics.

The ARPOD challenge [2] has been taken up by several researchers in proposing a variety of control strategies. A two-stage optimal control strategy is developed in [20], where the first part involves trajectory planning under a differentially flat system and the second part implements model predictive control on a linearized model. A supervisor is introduced to robustly coordinate a family of hybrid controllers in [21]. Safe reachsets (i.e. ReachAvoid sets) are computed for the ARPOD mission in [22] and used to solve for minimum fuel and minimum time trajectories.

The results presented in this thesis are directly related to previous works: the tool SDVTool and SwLQ control solution are first presented in [23], the SDLQ controller in [24], and a related tool for model predictive control in [25].

## 1.4 Organization

In Chapter 2, we present the detailed formulation of the various benchmark autonomous rendezvous missions. An overview of the formal verification approach used is given in Chapter 3, and its detailed implementation in SDVTool is presented in Chapter 4. Chapter 5 discusses how the benchmark models are specified for the software tools and summarizes the resulting safety guarantees achieved.

# CHAPTER 2

# SPACECRAFT RENDEZVOUS MODEL

In this chapter, we present the detailed hybrid satellite rendezvous mission models. First we give the orbital dynamics of the spacecraft in Sections 2.1-2.2. Section 2.3 introduces the mission constraints, and we propose two controllers designed to achieve the constrained rendezvous mission in Sections 2.4-2.5. We give an overview of the hybrid automata modeling framework and how our benchmarks are represented as hybrid models in Section 2.6.

## 2.1 Nonlinear Relative Motion Dynamics

The dynamics of the two spacecraft in orbit—the *target* and the *chaser*—are derived from Kepler's laws. We consider the case for relative motion in space, where the two spacecraft are restricted to the same orbital plane, resulting in two-dimensional, planar motion. The so-called Hill's relative coordinate frame is used. As shown in Figure 2.1, Hill's frame is centered on the target spacecraft, with $+\hat{\mathbf{i}}$-direction pointing radially outward from the Earth, $+\hat{\mathbf{k}}$-direction normal from the orbital plane, and $+\hat{\mathbf{j}}$-direction completing a right-handed system. We further assume that the target moves on a circular orbit, and thus, the $\hat{\mathbf{j}}$-direction aligns with the tangential velocity of the target.

The restriction on the target's orbit implies that the target-centered frame rotates with constant angular velocity. We will assume the target is in geostationary equatorial orbit (GEO), with angular velocity $n = \sqrt{\frac{\mu}{r^3}}$, where $\mu = 3.986 \times 10^{14}$ m$^3$/s$^2$ and $r = 42164$ km. The chaser's position is represented by the vector $\vec{\rho} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}}$, and the acceleration provided by the chaser's thrusters is denoted $\vec{u} = F_x\mathbf{i} + F_y\mathbf{j}$. Note we will also frequently refer to the *separation distance* with the notation: $\rho := |\vec{\rho}| = \sqrt{x^2 + y^2}$. The following equations are derived using Kepler's laws and constitute the nonlinear model

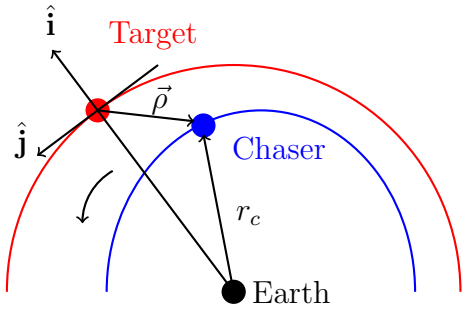



Figure 2.1: Hill's relative coordinate frame. Chaser's relative position vector is $\vec{\rho} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}}$. (Direction $+\hat{\mathbf{k}}$ out of the page not shown.)

(NLin) of the spacecraft dynamics:

$$\begin{aligned} \ddot{x} &= n^2 x + 2n\dot{y} + \frac{\mu}{r^2} - \frac{\mu}{r_c^3}(r+x) + \frac{F_x}{m_c}, \\ \ddot{y} &= n^2 y - 2n\dot{x} - \frac{\mu}{r_c^3} y + \frac{F_y}{m_c}, \end{aligned} \tag{2.1}$$

where $r_c = \sqrt{(r+x)^2 + y^2}$ and $m_c = 500$ kg is the mass of the chaser. The constant parameters of this set of equations are given in Table 2.1.

Table 2.1: Summary of constant parameters used to describe the continuous system dynamics in Equations (2.1)-(2.2).

| **Variable** | **Value** |
|---|---|
| $\mu$ | $3.986 \times 10^{14}$ m$^3$/s$^2$ |
| $r$ | 42164 km |
| $n$ | $7.29 \times 10^{-5}$ s$^{-1}$ |
| $m_c$ | 500 kg |

The three-dimensional case is also given in [2], where the satellites' relative altitude $z$ is considered. In this case, the chaser's relative position is $\vec{\rho} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}}$, external force or input vector $\vec{u} = F_x\mathbf{i} + F_y\mathbf{j} + F_z\mathbf{k}$, and dynamics $\ddot{z} = -n^2 z + \frac{F_z}{m_c}$. Notice this additional dimension evolves linearly and is decoupled from the two-dimensional system, so synthesizing a controller for the three-dimensional problem is a straightforward, independent extension of solving the two-dimensional control problem. We do not address this extension in this thesis.

The control and navigation problem for rendezvous is to choose $F_x, F_y$ to drive the system towards $\vec{x} = 0$, while adhering to the constraints that are

given in Section 2.3. Our solutions given in Sections 2.4-2.5 draw from theory for linear systems, hence we give a linear approximation in the next section. We will also observe the system under a passive abort, where rendezvous is no longer an objective and $F_x = F_y = 0$. In this case, the control portion is trivial but safety verification remains a crucial part of our analysis. An abort sequence is triggered when some system failure is detected. In order to keep this condition as general as possible, we assume that the event that triggers a passive abort occurs within some user-specified time interval. However, this could be customized to be a function of the state or some other parameter other than time if desired. We note that it is not expected for the system to satisfy a *passive safety* property (i.e. collision avoidance in Section 2.3) for an arbitrary choice of this interval.

## 2.2 Linear Dynamics

Linearization of Equations (2.1) about the system's equilibrium point at the origin results in the Clohessy-Wiltshire-Hill (CWH) equations [11], which are commonly used to capture the relative motion dynamics of two satellites within a reasonably close range. These equations for the linear model (Lin) are:

$$\begin{aligned} \ddot{x} &= 3n^2 x + 2n\dot{y} + \frac{F_x}{m_c}, \\ \ddot{y} &= -2n\dot{x} + \frac{F_y}{m_c}. \end{aligned} \tag{2.2}$$

Let the state vector be denoted by $\vec{x} = [x, y, \dot{x}, \dot{y}]^T$. The state-space form of these linear time-invariant (LTI) equations is:

$$\dot{\vec{x}} = A\vec{x} + B\vec{u},$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 2n \\ 0 & 0 & -2n & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m_c} & 0 \\ 0 & \frac{1}{m_c} \end{bmatrix}, \vec{u} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}.$$

## 2.3 Constraints and Safety Requirements

In this section, we enumerate the properties that define a safe and successful mission as given by [2], and how they are modeled for verification tools. The tools we use (C2E2, DryVR, and SpaceEx) require these sets of states be specified by affine inequalities (and more specifically, convex polyhedron). Bear in mind that the goal of rendezvous is to drive the system to state $\vec{x} = 0$, but the goal of passive abort is to avoid the region around $\vec{x} = 0$. These distinct objectives naturally result in mutually exclusive constraints.

**Max thrust** The thrusters are physically limited, so we assume that at any given time, the thrusters cannot provide more than 10N of force in any single direction. Then the constraint

$$|F_x|, |F_y| \leq 10 \text{ N}$$

is captured by the following unsafe sets of states:

$$\begin{aligned} F_x &< -10, & -F_x &< -10, \\ F_y &< -10, & -F_y &< -10. \end{aligned}$$

Notice that $F_{\{x,y\}}$ is not strictly part of the state space ($\vec{x}$), but soon we shall choose $F_{\{x,y\}}$ directly as a function of the state $\vec{x}$. But we could also introduce an augmented state vector $\vec{\tilde{x}} = \begin{bmatrix} \vec{x} \\ \vec{u} \end{bmatrix}$. This latter approach is implemented as discussed in Section 5.1.

**LOS cone and approaching velocity** When the chaser is in close-range rendezvous ($\rho = \sqrt{x^2 + y^2} < 100$ m), the chaser must remain within a line-of-sight (LOS) cone (see Figure 2.5), and its total velocity must remain under 5 cm/s, so $\sqrt{\dot{x}^2 + \dot{y}^2} \leq 5$ cm/s. Notice this unsafe set,

$$\{\vec{x} : \sqrt{x^2 + y^2} < 100 \cap \sqrt{\dot{x}^2 + \dot{y}^2} > 0.05\},$$

cannot be captured by affine inequalities exactly. Thus it is approximated by the intersection of two polytopes using a hexagon in $x, y$ that circumscribes the circle $\sqrt{x^2 + y^2} = 100$ and a hexagon in $\dot{x}, \dot{y}$ that inscribes the circle $\sqrt{\dot{x}^2 + \dot{y}^2} = 0.05$ (see Figure 2.2). The unsafe set

is completely contained in the approximated unsafe set. The detailed approximation is given in Table 5.1.

**Minimum separation distance** During a passive abort, the chaser must avoid collision with the target. The target, in the simplest case, can be treated as a point mass at the origin, where a small ball or box should be used to bound this point to account for limitations in numerical precision. We may also account for the target satellite's dimensions, which may range from the order of 1 m to 100 m, so the size of this bounding box will vary depending on the scenario. For our experiments, we use a box with a 1 m circumradius (see Figure 2.3 and Table 5.1).

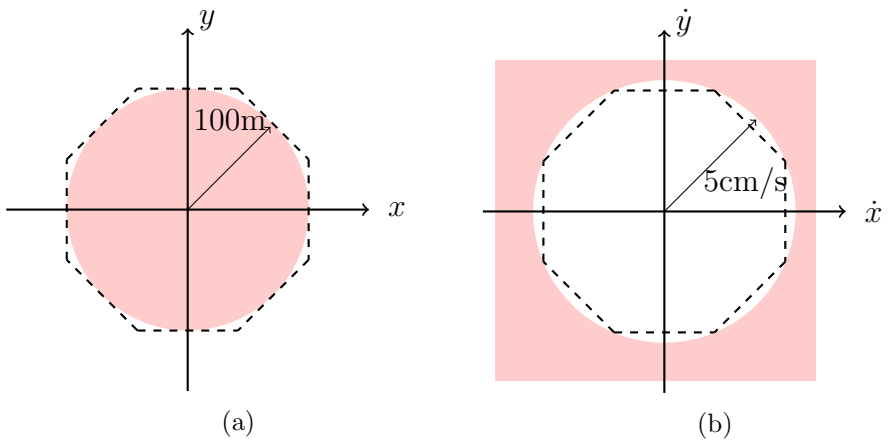



Figure 2.2: The dashed lines show the polytopic approximation of the unsafe set of states resulting from the LOS constraint in projections on $x, y$ in (a) and on $\dot{x}, \dot{y}$ in (b).

## 2.4 Switched Linear Quadratic Regulator

In this section, we construct the first of two controllers proposed to achieve rendezvous. This is a full-state feedback controller, namely, a linear quadratic regulator (LQR). LQR is a linear function of the state, of the form $\vec{u} = K_i\vec{x}$, where $i$ is introduced because we will design a set of LQR solutions. One of the constraints given in [2] is a maximum time bound on completing the ARPOD mission. We do not explicitly check for this property because we are only interested in a subset of the full mission at this time. However, it

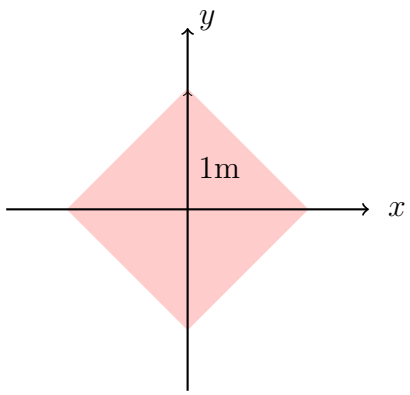



Figure 2.3: The unsafe set (shown in $x, y$-projection) resulting from collision avoidance constraint when the spacecraft is in a Passive Abort and not performing Rendezvous.

is clear that we cannot find a reasonable solution using a single LQR control law as this solution would need to meet the constraint on total velocity, even beyond the close-range rendezvous portion. In turn, completing the initial rendezvous at larger separation distances would take an unreasonably long time. We choose to solve for two LQR control laws: one for rendezvous at longer ranges and another for the more restrictive close-range. A third trivial control law in this set is that for the passive abort: $K_{passive} = 0$. We will refer to the resulting closed-loop system using each control law as a *mode*, and the reason will become apparent in Section 2.6.

In general, closed-loop feedback control is desirable because the system can measure and adjust for errors, and ultimately guarantee liveness (i.e. eventually the target will be reached). LQR is specifically chosen because it is constructed by minimizing a quadratic cost function, which we can choose so as to roughly satisfy our safety constraints. LQR is only directly applicable to (or guaranteed to be optimal for) linear systems, so we design the control for the linearized model in (2.2), but we will use the same control with nonlinear dynamics (2.1) when applying verification tools.

Each constant gain matrix $K_i \in \mathbb{R}^{2\times 4}$, $i \in \{1, 2\}$, is obtained by solving two independent infinite-horizon LQR problems, as follows:

$$\min_{\vec{u}} \tilde{J}_i,$$

subject to

$$\dot{\vec{x}} = A\vec{x} + B\vec{u},$$

where

$$\tilde{J}_i = \int_0^\infty (\vec{x}^T Q_i \vec{x} + \vec{u}^T R_i \vec{u}) dt, \tag{2.3}$$

where $Q_{\{1,2\}}$ and $R_{\{1,2\}}$ are chosen so as to improve performance (i.e. minimize flight time) when constraints are relaxed (i.e. when $\rho \geq 100$ m) and to restrict the system's behavior to fit tighter constraints otherwise (i.e. when the LOS cone and approaching velocity properties are enabled).

To achieve this, we apply a common approach to encoding restrictions on state and input variables into $Q_i$ and $R_i$ called Bryson's rule [26]. Bryson's rule dictates choosing $Q_i$ and $R_i$ to be matrices with diagonals such that $q_{jj} = \frac{1}{max(x_j^2)}$, $j \in \{1, ..., 4\}$ and $r_{jj} = \frac{1}{max(u_j^2)}$, $j \in \{1, 2\}$. Typically these terms correspond to the expected range of values for each variable, which then normalizes the cost of errors in each direction. We take these terms to refer to the largest *desired* value of each variable, given by the constraints. While the LQR gains are obtained with our constraints in mind, the resulting controller does not guarantee these constraints are never violated. This is why formal verification is still required. This design process is repeated for rendezvous at long and short ranges, corresponding with relaxed and tightened constraints respectively.

The solution to the optimization problem of Equation (2.3) is obtained by solving the continuous algebraic Riccati equation (ARE). Part of the motivation for relying on a LQR-like framework is that ARE solvers are well-studied and readily available. Given the controllability of the relative motion model (2.2) and a restriction of $Q_i$ and $R_i$ to symmetric, positive definite matrices, there exists a guaranteed unique optimal control solution $\vec{u} = -K_i \vec{x} = -R_i^{-1} B^T P_i$, where $P_i$ is a positive definite solution to the ARE. The solution is not only optimal with respect to (2.3) but ensures the closed-loop system is globally asymptotically stable (GAS) about the origin *in each mode*. This does not guarantee that the overall system is GAS under switching.

Given this switched LQR controller (SwLQ), the resulting nonlinear model

of the dynamics (or the *flow equations* for NLin-SwLQ) are:

$$\begin{aligned}\ddot{x} &= \left(3n^2 - \frac{k_{11}^i}{m_c}\right)x - \frac{k_{12}^i}{m_c}y - \frac{k_{13}^i}{m_c}\dot{x} + \left(2n - \frac{k_{14}^i}{m_c}\right)\dot{y}, \\ \ddot{y} &= -\frac{k_{21}^i}{m_c}x - \frac{k_{22}^i}{m_c}y - \left(2n + \frac{k_{23}^i}{m_c}\right)\dot{x} - \frac{k_{24}^i}{m_c}\dot{y},\end{aligned} \tag{2.4}$$

where

$$K_i = \begin{bmatrix} k_{11}^i & k_{12}^i & k_{13}^i & k_{14}^i \\ k_{21}^i & k_{22}^i & k_{23}^i & k_{24}^i \end{bmatrix}.$$

The flow equations for the Lin-SwLQ configuration can be written more compactly as:

$$\dot{\vec{x}} = (A - BK_i)\vec{x} \tag{2.5}$$

We note that a conventional switched LQR controller would require the solution of the following finite horizon problem (instead of the $\tilde{J}_i$ given in Equation (2.3)):

$$\min_{\vec{u}} \sum_{i=1}^{N} \int_{t_i}^{t_{i+1}} (\vec{x}^T Q_i \vec{x} + \vec{u}^T R_i \vec{u}) dt, \tag{2.6}$$

where $t_i$ denotes switching times with $N$-total switches ($N = 2$ for SwLQ). Thus, an optimal LQR solution would drive the state of the system to the origin by the specified time horizon $T_{N+1}$, while minimizing (2.6). The infinite-horizon formulation in (2.3) is easier to solve, but it loses the guarantee that the state reaches the origin in finite time. This is acceptable for the rendezvous maneuver as the goal is to drive the spacecraft within closer range, not to a terminal point. The infinite-horizon LQR result will still drive the spacecraft towards the origin, which can be observed by formulating invariant sets of states using Lyapunov functions. (It is sufficient to solve for a quadratic Lyapunov function using the Lyapunov equation for linear systems and then apply Lyapunov's first method to prove stability of the nonlinear system.) At each switching time $t_i$, the invariant set of states at $t_i$ is strictly contained in the invariant set at $t_{i-1}$. It follows that this switched controller with $i = \{1, 2\}$ (and the SDLQ in Section 2.5) is stable in the sense of

Lyapunov.

## 2.5 State-Dependent Linear Quadratic Control

We extend the two-stage, switched LQR from the previous section to $N > 2$ finitely many switches where the switches are brought about by a time-dependent switching signal. Additionally, the system dynamics (i.e. equations (2.4)-(2.5)) are not computed a priori but determined by a function of the current state; thus, we refer to the new control scheme as state-dependent linear quadratic (SDLQ) control.

At every switching time (i.e. after every $\delta$ interval), the state-feedback law is computed just as before, solving the LQR problem in Equation (2.3). In the switched LQ scheme, $Q_{\{1,2\}}$, $R_{\{1,2\}}$ are constant matrices, and $K_1$, $K_2$ are solved offline. Now, the cost coefficients are functions denoted $Q_i(\vec{x}(t_i))$ and $R_i(\vec{x}(t_i))$ and each $K_i$ is computed at time $t_i$.

In the example for this thesis, we choose a constant $R_i = R_j = R \in \mathbb{R}^{2\times 2}$, $\forall i, j \in \{0, 1, ..., N\}$, and $Q_i(\vec{x}(t_i))$ defined as follows:

$$\begin{aligned} Q_i(\vec{x}(t_i)) &= diag\left(\frac{1}{q_x^2}, \frac{1}{q_y^2}, \frac{1}{q_{\dot{x}}^2}, \frac{1}{q_{\dot{y}}^2}\right), \\ \text{where } q_x &= 5\left(|x(t_i)| + \epsilon\right)\left(1 + \frac{(|x(t_i)| + \epsilon)^2}{\rho^2}\right), \\ q_y &= 5\left(|y(t_i)| + \epsilon\right)\left(1 + \frac{(|y(t_i)| + \epsilon)^2}{\rho^2}\right), \\ q_{\dot{x}} &= \frac{40\left(|x(t_i)| + \epsilon\right)}{\rho}, \\ q_{\dot{y}} &= \frac{40\left(|y(t_i)| + \epsilon\right)}{\rho}, \end{aligned} \tag{2.7}$$

for some small $\epsilon > 0$ to avoid division by zero.

Notice $Q_i$ remains symmetric positive definite for all $\vec{x} \in \mathbb{R}^4$ and $R$ is also chosen to be symmetric positive definite. Then, as in the switched LQ control, we are guaranteed to find a stabilizing solution $\vec{u}_{[t_i, t_{i+1})} = -K_i\vec{x}$.

The current choice of (2.7) is motivated by satisfying the LOS constraint (see Section 2.3), hence it is a function only of $x$, $y$ and not of relative velocities $\dot{x}$, $\dot{y}$. Bryson's rule can be observed as a starting design choice in $q_x$, $q_y$ in the $(|x(t_i)| + \epsilon)$, $(|y(t_i)| + \epsilon)$ terms. In other words, the maximum

desired values for $x,\ y$ contract as the chaser moves towards the origin. The terms $(|\{x,y\}(t_i)| + \epsilon)/\rho$ approximate $|\cos(\theta)|$ and $|\sin(\theta)|$, where $\theta$ is the angular position of the total displacement vector $\vec{\rho}$. These terms could be used to enforce the $\theta$ restriction in the LOS region more directly.

Although the computation of the controller is different from SwLQ (i.e. now requires periodic re-computation of Equation (2.3)), the description of the dynamics for NLin-SDLQ and Lin-SDLQ looks identical to that given previously in Equations (2.4)-(2.5) for NLin-SwLQ and Lin-SwLQ, just with $i$ extended to $i = 1, 2, ..., N$.

## 2.6 Hybrid Models

In this section, we introduce the notion of a hybrid automaton model and unify all the parts discussed thus far under this modeling framework.

A *hybrid automaton* is a tuple $\langle \mathcal{V}, \Theta^{\mathcal{H}}, \mathcal{A}, D, \mathcal{T} \rangle$, where:

**(a)** $\mathcal{V} = X \cup \mathcal{L}$ is the set of state variables, with the state space taken to be the valuation $val(\mathcal{V})$. $X$ is the set of continuous variables ($val(X) = \mathbb{R}^4$ for our problem), and $\mathcal{L}$ is a discrete variable with countable set $val(\mathcal{L})$ referred to as *locations* or *modes*.

**(b)** $\Theta^{\mathcal{H}} \subseteq val(\mathcal{V})$ is the set of initial states.

**(c)** $\mathcal{A}$ is a set of transition labels.

**(d)** $D \subseteq val(\mathcal{V}) \times \mathcal{A} \times val(\mathcal{V})$ is the set of transitions—each characterized by a *guard* and *reset map*. These give conditions under which a transition is enabled and jump dynamics, respectively.

**(e)** $\mathcal{T}$ is a set of trajectories (continuous dynamics) for $X$ or flow equations, which are described by ODEs in our problem. For each $l \in val(\mathcal{L})$, there is a unique set of trajectories $E_l$ and an *invariant* $I_l \subseteq val(X)$, which indicates when the system may evolve according to $E_l$. If $x \in val(X)$ and $x \notin I_k$, then the system must be in some other mode $l \in val(\mathcal{L}) \neq k$.

In general, trajectories do not have to be described by ODEs but must satisfy certain properties given in [27]. A transition may be taken as long as

its guard is satisfied. When the mode invariant aligns with a guard so that the transition must be taken as soon as it is enabled, this is called an *urgent* transition. Such transitions may preserve determinism. However, it is possible for multiple transitions to be enabled at a given time, so nondeterminism can still arise from not knowing which transition is taken.

For the benchmark spacecraft problem, we have $X = \{x, y, \dot{x}, \dot{y}\}$, and given $K_i$, all trajectories in $E_i \in \mathcal{T}$ are described by Equations (2.4) for NLin and Equations (2.5) for Lin dynamics. The SwLQ controller lends $val(\mathcal{L}) = \{Mode1, Mode2\}$, with respective invariants: $I_1 = \{\vec{x} : \rho \geq 100\}$ and $I_2 = \{\vec{x} : \rho \leq 100\}$. For either the passive abort scenario Pass or the SDLQ controller, we introduce an additional continuous variable so that $X = \{x, y, \dot{x}, \dot{y}, \tau\}$, where $\tau$ is a timer variable with $\dot{\tau} = 1$. The Pass scenario introduces an additional mode $Passive \in val(\mathcal{L})$, with invariant $I_{Pass} = \{(\vec{x}, \tau) : \tau \geq t_1\}$, and incoming transitions with guard $t_1 \leq \tau \leq t_2$. As previously mentioned, we choose to consider a passive abort nondeterministically within a time interval $[t_1, t_2]$.

Hybrid models for the Lin-SwLQ, Lin-SwLQ-Pass and Lin-SDLQ configurations are shown in Figures 2.4-2.7. These examples should sufficiently illustrate how each of the 8 possible configurations can be formulated as hybrid models. To summarize Chapter 2, we have presented a set of 8 benchmark formulations of an autonomous satellite rendezvous mission that result from choosing NLin or Lin dynamics, SwLQ or SDLQ control, and with or without Pass. The systematic analysis of such hybrid models is presented next in Chapter 3.

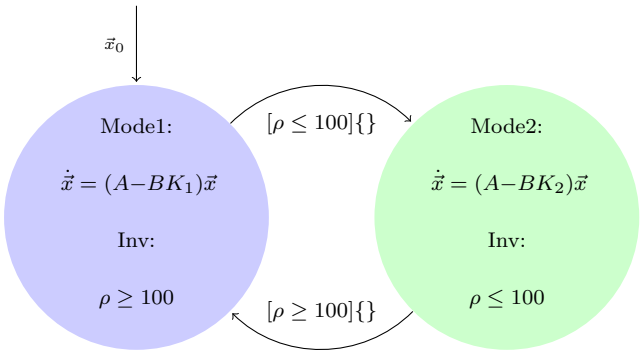



Figure 2.4: Hybrid model for spacecraft rendezvous, using the Lin-SwLQ configuration. The invariants in Mode 1 and Mode 2 are defined exclusively by the chaser's position, as shown by corresponding colors in Figure 2.5. Recall a hybrid automaton is the tuple $\langle \mathcal{V}, \Theta^{\mathcal{H}}, \mathcal{A}, D, \mathcal{T} \rangle$. Here, $\Theta^{\mathcal{H}}$ is depicted with the arrow labeled by $\vec{x}_0 \in val(X)$, pointing to initial mode $Mode1 \in val(\mathcal{L})$. The two members of $\mathcal{A}$ are not explicitly given here, but these transitions in $D$ are labeled by guards in "[]" and resets in "{}". Notice these transitions are urgent. The trajectories $\mathcal{T}$ are specified by the ODEs and invariants for each mode.

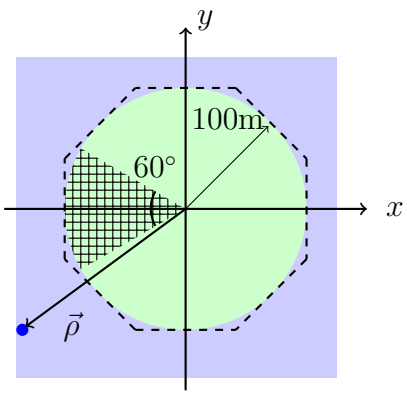



Figure 2.5: The invariants for Mode 1 and Mode 2 in Figures 2.4-2.6 are shown in corresponding colors (blue and green). However, they are approximated using the polytope shown by the dashed lines in the model inputted to the verification software tools. The LOS region (gridded) indicates the safe region of operation whenever the chaser is in Mode 2.

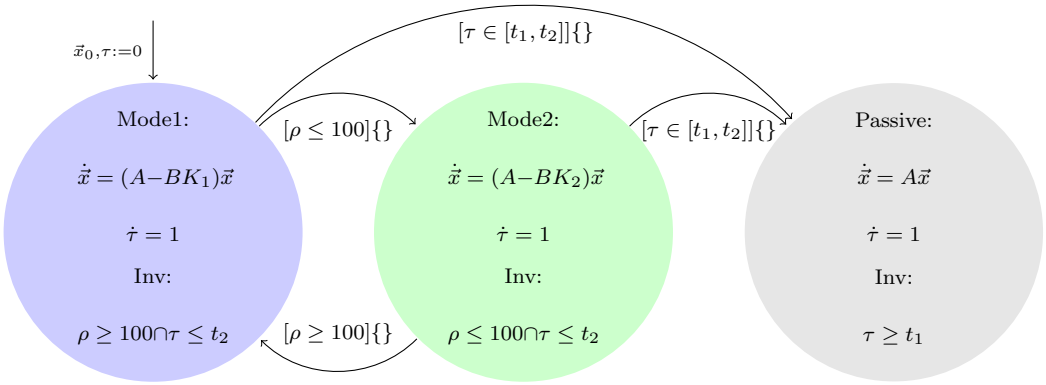



Figure 2.6: Hybrid model for spacecraft rendezvous with passive abort, using the Lin-SwLQ-Pass configuration. Transitions to Passive occur nondeterministically within an interval of time.

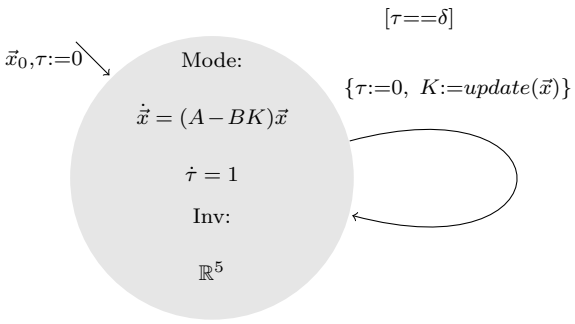



Figure 2.7: Hybrid model for spacecraft rendezvous using the Lin-SDLQ configuration. The subroutine $update(\vec{x})$ computes the SDLQ solution.

# CHAPTER 3

# VERIFICATION APPROACH

In this chapter, we present the underlying verification algorithm in the software tools we use. We introduce the notion of reachability analysis—the building block to our verification approach—and give an overview of how each tool differs in this regard.

## 3.1 Safety Verification Problem

To rigorously demonstrate that a hybrid system ($\mathcal{H} = \langle \mathcal{V}, \Theta^{\mathcal{H}}, \mathcal{A}, D, \mathcal{T} \rangle$) can never exhibit unsafe behaviors, we study the reachability problem: Given a set of initial states $\Theta^{\mathcal{H}} \subset val(\mathcal{V})$ and a set of unsafe states $\mathcal{U} \subset val(\mathcal{V})$, does there exist a trajectory from some $v \in \Theta^{\mathcal{H}}$ to some $v \in \mathcal{U}$? If so, then we conclude the system is unsafe. However, in general, the reachability problem is undecidable [28, 29, 30], so existing approaches typically provide a means of overapproximating *reachsets* $\mathcal{R} \subset val(\mathcal{V})$—sets of reachable states of the system—over a bounded time horizon $T \in \mathbb{R}_{\geq 0}$.

### 3.1.1 Related Works

Several recent works propose simulation-based approaches to computing such reachsets [31, 32, 33, 34, 12, 13, 15, 35]. The toolbox Breach [31, 36] facilitates the use of sensitivity analysis to measure how neighboring trajectories can evolve comparatively, thereby providing the knowledge to construct approximate reachsets. These approximations are not guaranteed to be sound for nonlinear systems, but contraction metrics in [37] provide a means of sensitivity analysis for nonlinear systems. A generalized approach that extends contraction metrics for nonlinear systems [32] to compute reachsets serves as a basis for some of the tools we will apply and discuss in Section 3.3.

An on-the-fly algorithm to compute this metric for nonlinear systems is presented in [34] and implemented in the C2E2 software tool [12, 13]; whereas a data-driven approach is implemented in the DryVR software tool [15]. In building SDVTool, we use the approach called Parsimonious in [35], which is a simulation-based method for reachability of linear systems but does not rely on sensitivity analysis. Instead, the reachable sets of states are formulated by applying the superposition principle—hence the restriction to linear systems—to a finite number of simulation traces. These tools mentioned will be used in our experiments in Chapter 5. There are also several works using simulations to reason about safety properties that do not rely on directly computing reachsets [38, 39, 40].

### 3.1.2 Preliminaries

With some abuse of notation, we say a *reachset* $\mathcal{R}(\Theta^{\mathcal{H}}, [t_1, t_2]) \subset val(\mathcal{V})$ contains all the states that are reachable from $\Theta^{\mathcal{H}}$ at $t = 0$ over time interval $[t_1, t_2]$. We will also refer to a *reachtube* $R^T$, a sequence of compact sets $\{(O_i, t_i)_{i=0}^N\}$, such that $\mathcal{R}(\Theta^{\mathcal{H}}, [t_{i-1}, t_i]) \subseteq O_i$, $t_0 = 0$, and $t_N = T$. Notice $O \supseteq \mathcal{R}$ is an overapproximation of the set of reachable states. We assume $t_i - t_{i-1} = \delta$ is uniform for all $i$. Intuitively, we can think of the reachtube $R^T$ as a reachset over $T$: $\mathcal{R}(\Theta^{\mathcal{H}}, [0, T])$, or as a collection of reachsets: $\cup_i(\mathcal{R}(\Theta^{\mathcal{H}}, [t_{i-1}, t_i]))$. Denote a trajectory of the hybrid system as a function $\xi : val(\mathcal{V}) \times \mathbb{R}_{\geq 0} \to val(\mathcal{V})$, such that $\xi(v_0, t)$ is the state of the system at time $t$ starting from $v_0$. Furthermore, a simulation trace of a trajectory is denoted by $\psi = \{\xi(v_0, t) : 0 \leq t \leq T\}$.

### 3.1.3 Problem Approach

The safety verification problem is solved if:

1. we show that all reachable states are completely disjoint from the set of unsafe states, i.e. $R^T \cap \mathcal{U} = \emptyset$ (program returns **Safe**), or

2. we provide an unsafe counterexample trajectory $\xi(v_0, t) \in \mathcal{U}$, such that $v_0 \in \Theta^{\mathcal{H}}$ and $t \leq T$ (returns **Unsafe**).

Since $R^T$ is an overapproximation of the true reachable set of states, it is possible that a solution in this sense does not exist. In other words, a state $v \in R^T$ may intersect with $\mathcal{U}$ but is not truly reachable so there exists no trajectory such that $\xi(v_0, t) = v$. Thus, in practice the safety verification algorithm may terminate with an additional output **Unknown**.

The high-level verification algorithm is given in Algorithm 1, and a more detailed version of it (that includes the implementation of line 1) can be found in Figure 1 of [32]. If the computed reachtube cannot be shown to be **Safe**, a finite number of trajectories are randomly generated (up to a user-defined limit SimBnd). If one of these simulated trajectories provides a counterexample to safety, an **Unsafe** result is given. Otherwise, it terminates with **Unknown**. The next sections will discuss the approach to achieving line 1 of Algorithm 1.

**Algorithm 1:** Safety verification of hybrid automaton $\mathcal{H}$. A **Safe** result guarantees no state in $\mathcal{U}$ is reachable by $\mathcal{H}$ over time horizon $T$ with initial set of states $\Theta^{\mathcal{H}}$ or $\Theta$, if optionally specified.

**Input**: $T$,$\mathcal{U}$,$\mathcal{H}$,$(\Theta)$
**Output**: SafeFlag
1 $R^T \leftarrow computeReachtube(\mathcal{H}, T, (\Theta))$
2 **if** $R^T \cap \mathcal{U} \neq \emptyset$ **then**
3  **for** $j = 1$ :SimBnd **do**
4  $\vec{x}_0 \leftarrow$ a randomly sampled point in $\Theta^{\mathcal{H}}$ (or $\Theta$)
5  $\psi \leftarrow Simulate(\vec{x}_0, T, \mathcal{H})$
6  **if** $\psi \cap \mathcal{U} \neq \emptyset$ **then**
7  **return** *Unsafe*
8  **end**
9  **end**
10  **return** *Unknown*
11 **end**
12 **return** *Safe*

**Proposition 1.** *Algorithm 1 is sound.*

*Proof.* The proof follows from Proposition 2 in Section 3.2, where we are given that $R^T$ (computed in line 1) contains all reachable states from some state in $\Theta^{\mathcal{H}}$ within time $T$. If Algorithm 1 terminates **Safe** (line 12), then $R^T \cap \mathcal{U} = \emptyset$ must hold, as the **if**-block (lines 2-11) is skipped. Thus we conclude that all reachable states are safe. Algorithm 1 only returns **Unsafe**

(line 7) when a trajectory $\psi$ from some initial state in $\Theta^{\mathcal{H}}$ up to time $T$ contains an unsafe state. Therefore, $\psi$ itself is a proof of contradiction for safety. □

## 3.2 Computing Reachtubes

Recall that a hybrid automaton $\mathcal{H} = \langle \mathcal{V}, \Theta^{\mathcal{H}}, \mathcal{A}, D, \mathcal{T} \rangle$ has state variables $\mathcal{V} = X \cup \mathcal{L}$, with $X$ the set of continuous variables and $\mathcal{L}$ a discrete variable. From here on out, we restrict the definitions given in Section 3.1 to continuous variables $X$. In most domains and applications, we are only concerned with studying the state variables $\vec{x} \in \mathbb{R}^n = val(X)$, so the discrete mode is decoupled from our safety properties and reachtube computations. By choosing $\Theta, \mathcal{U}, \mathcal{R}(\Theta, [t_1, t_2]), R^T \subset \mathbb{R}^n$ and $\xi(\vec{x}_0, t) : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, we are still able to express all the necessary parts of the problem. (Notice we introduce initial set $\Theta \subset val(X)$ to distinguish from $\Theta^{\mathcal{H}} \subset val(\mathcal{V})$ as part of the tuple for $\mathcal{H}$.) The approaches to reachset computation presented in the next sections rely on fixed dynamics, or knowledge of the full hybrid state $v \in val(\mathcal{V})$ since $l \in val(\mathcal{L})$ gives the dynamics $E_l$. So while $\Theta$ is sufficient for specifying the verification problem, the underlying algorithms still require knowledge of the set $\Theta^{\mathcal{H}}$, which is implicitly constructed by $\Theta$ as follows:

$$\Theta^{\mathcal{H}} = \bigcup_{\vec{x}_0 \in \Theta} \{v = (\vec{x}_0, l_i) : \vec{x}_0 \in I_{l_i}\},$$

where $I_{l_i}$ is the invariant set of states for mode $l_i \in val(\mathcal{L})$.

The overarching algorithm for computing reachtubes in the tools C2E2, DryVR, and SDVTool is given by Algorithm 2. Given a time horizon $T$ and an initial set of states $\Theta \subset \mathbb{R}^n$ (where $\theta_i \subseteq \Theta$ denotes the subset of states in mode $l_i$), a reachset $\mathcal{R}(\theta_i, T)$ is computed using the flow equations of mode $l_i$ (or $E_{l_i}$). Then the convex hull of all states reachable by taking an enabled transition $d_j$ from a subset $R_j \subseteq \mathcal{R}(\theta, T)$ is taken to be $(\theta', l')$. For each possible transition, a new reachset $\mathcal{R}(\theta', T)$ is computed under the new mode $l'$. After all possible transitions within the time bound are accounted for, the union of the reachsets in each mode forms the reachtube $R^T$.

Notice this algorithm is necessarily conservative. Nondeterministic transitions result in more reachset subcomputations. Reachset subcomputations

**Algorithm 2:** *computeReachtube*: Reachtube computation for a hybrid automaton $\mathcal{H}$.

**Input**: $\mathcal{H}, T, \Theta$
**Output**: $R^T$

$R^T \leftarrow \emptyset$
$\theta_1, ..., \theta_k \leftarrow$ Partition $\Theta$ such that $\Theta^{\mathcal{H}} = \bigcup_{i=1:k} (\theta_i, l_i)$
$Q.push((\theta_1, l_1), ..., (\theta_k, l_k))$
**while** $Q$ *not empty* **do**
    $(\theta, l) \leftarrow Q.pop()$
    $\mathcal{R}(\theta, T) \leftarrow computeReachset(\theta, l, E_l, T)$
    $\mathcal{R}(\theta, T) \leftarrow \mathcal{R}(\theta, T) \cap$ invariant $I_l$
    **for** *each transition* $d_j \in D$ *from mode* $l$ **do**
        **if** $d_j$ *is enabled by* $R_j \subseteq \mathcal{R}(\theta, T)$ **then**
            $(\theta', l') \leftarrow$ state after taking transition $d_j$ from $R_j$
            $Q.push((\theta', l'))$
        **end**
    **end**
    $R^T \leftarrow R^T \cup \mathcal{R}(\theta, T)$
**end**
**return** $R^T$

always cover the full time horizon length $T$, so this generally includes reachsets for $t > T$ since transitions do not occur (if at all) until after some time has elapsed. Thus the overapproximated regions of a reachtube can blow up quickly for a number of reasons: large uncertainty in initial set, long time horizon, nondeterministic transitions, number of transitions, and representation of the reachset. This behavior increases the likelihood of overapproximated regions intersecting with the unsafe set of states in Algorithm 1, barring a **Safe** or **Unsafe** result. This problem is alleviated in the software tools by partitioning the initial sets into smaller subsets before computing the reachtube. In the Pass benchmark models, we further reduce the effects of nondeterminism by restricting the transition guard to the Passive mode (see $[t_1, t_2]$ in Figure 2.6) to a relatively small interval of time.

**Proposition 2.** *Algorithm 2 returns guaranteed overapproxmations of reachable states of* $\mathcal{H}$ *from* $\Theta$ *within* $T$ *time.*

*Proof sketch.* The proof follows from Proposition 3 in Section 3.3, which guarantees that all reachable states from an initial hybrid state $(\theta, l)$ that can evolve according to the dynamics of mode $l$ (i.e. the ODEs described

by $E_l$) up to time $T$ are contained in the overapproximated reachset $\mathcal{R}(\theta, T)$ computed in line 6.

First notice that the reachtube $R^T$ must include all initial states $\Theta$. This is satisfied because $\Theta = \bigcup_{i=1:k} \theta_i$ from line 2, and each $\theta_i$ is checked in the main loop, where $\mathcal{R}(\theta, T) \supseteq \theta$ by Proposition 3 in line 6 and because $\theta \cap I_l$ by line 2 in line 7. The inner **for**-loop (lines 8-13) does not change $\mathcal{R}(\theta, T)$, which is completed contained in $R^T$ by line 14. Thus after $k$ iterations of line 14 corresponding to checking each $\theta_i$, we see that $\Theta \subseteq R^T$ for any $T \geq 0$.

Without loss of generality, assume $k = 1$. During the first execution of the **while**-loop, we have $\theta = \theta_1 = \Theta$ and line 6 computes a reachset $\mathcal{R}(\theta, T)$ that contains all reachable sets of states from $\theta$ under the dynamics of mode $l$ up to time $T$ by Proposition 3. Line 7 preserves only the valid subset of $\mathcal{R}(\theta, T)$—the continuous states that are possible when the discrete mode is $l$, as defined by invariant $I_l$. It is clear that all reachable states from $\theta$ with mode $l$ is still contained in $\mathcal{R}(\theta, T)$ after line 7. Again, the **for**-loop does not affect this reachset, and the reachset $\mathcal{R}(\theta, T)$ is completely contained in $R^T$ by line 14.

If no transitions can be taken from $\mathcal{R}(\theta, T)$, the algorithm does not enter the **for**-loop (lines 8-13) so nothing is pushed to $Q$, which is now empty after line 5. The **while**-loop terminates and the returned reachtube $R^T$ overapproximates the reachable states of the system $\mathcal{H}$ up to time $T$.

If a transition to mode $l_j$ can be taken from $\mathcal{R}(\theta, T)$, we take $R_j$ to be the subset of $\mathcal{R}(\theta, T)$ that intersects with the transition guard. Then for every $v \in R_j$ and $v'$ is the state after taking transition $d_j$, $v' \in (\theta', l_j)$. Thus for $l' = l_j$, we obtain an overapproximation of reachable states immediately after transition $d_j$ in $(\theta', l')$ in line 10. When the **for**-loop terminates and all possible transitions have been checked, $Q$ contains a set of $\{(\theta'_i, l'_i)\}$, whose union forms a new "initial" set of states corresponding to an overapproximation of all reachable states after one transition has been taken. In fact, checking all the sets currently in $Q$ is equivalent to recursively calling *computeReachtube*$(\mathcal{H}, T, \tilde{\Theta})$, with $\tilde{\Theta} = \bigcup_i \theta'_i$. The remaining executions of Algorithm 2 can be re-casted as a series of recursive function calls, and the last recursive call corresponds to an initial set from which no transitions can be taken, in which case we have already previously concluded the returned value of $R^T$ is an overapproximation of all reachable states from the input

argument $\Theta$ for time $T$. We can invoke an inductive argument to show that the initial call to *computeReachtube*, Algorithm 2 itself, satisfies Proposition 2.

□

The upcoming sections discuss the underlying algorithms for computing reachsets (line 4 in Algorithm 2). This is the component that the various tools we consider implement differently. Notice this subroutine *computeReachset* takes the dynamics of a single mode $l$ as an input argument. In fact, the over-approximated reachset it returns assumes the system can evolve according to only one set of ODEs $E_l$, so the hybrid nature of the system is not accounted for at this step. Hence, the primary purpose of Algorithm 2 is to collect all reachable states considering all possible sets of continuous dynamics.

## 3.3 Computing Reachsets with Discrepancy Functions

In this section, we present the simulation-based reachability computation algorithm that is implemented in two of the tools we test: C2E2 [12, 13] and DryVR [15]. The discussion in the preceding sections have directly involved hybrid automaton models, but for the remainder of this chapter, we will only work with a fixed set of ODEs given by $E_l$ and no notion of a discrete mode.

A continuous function $\beta : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a *discrepancy function* if, for any pair of states $\vec{x}_1, \vec{x}_2 \in \mathbb{R}^n$ and any time $t \geq 0$, the following holds:

$$||\xi(\vec{x}_1, t) - \xi(\vec{x}_2, t)|| \leq \beta(\vec{x}_1, \vec{x}_2, t), \tag{3.1}$$

$$\lim_{\vec{x}_1 \to \vec{x}_2} \beta(\vec{x}_1, \vec{x}_2, t) = 0. \tag{3.2}$$

Given a valid discrepancy function, we can compute $\mathcal{R}(\theta, T)$ by choosing a $\vec{x}_0 \in \theta$ to generate a simulation trace $\psi$ over time horizon $T$. Then the simulation trace $\psi$ is expanded by $\beta$ (by way of the Minkowski sum), so that the result is an overapproximation of the reachset from $\Theta$. This is given in Algorithm 3.

**Proposition 3.** *Algorithm 3 is sound and relatively complete.*

The proof of Proposition 3 is given in Theorem 11 of [32].

**Algorithm 3:** *computeReachset*: Compute the overapproximation of reachable states from initial set $\theta$ that evolves according to $E_l$ using a discrepancy function

**Input**: $\theta, T, E_l, \beta$
**Output**: $\mathcal{R}(\theta, T)$
1 $\vec{x}_0 \leftarrow$ some point in $\theta$
2 $\psi \leftarrow Simulate(\vec{x}_0, T, E_l)$
3 $\mathcal{R}(\theta, T) \leftarrow \psi \oplus \beta$
4 **return** $\mathcal{R}(\theta, T)$

Candidate discrepancy functions can be obtained using a global Lipschitz constant or using a matrix norm for linear systems. However, typically these approaches give discrepancy functions that blow up exponentially with time and therefore are not useful for verifying problems with long time horizons. Approaches for computing tighter discrepancy functions are implemented in C2E2 and DryVR.

In C2E2, the discrepancy function is computed using an automatic on-the-fly approach that relies on bounds of the Jacobian matrix of the dynamics of the continuous components of the hybrid system [13]. In DryVR, the discrepancy function is computed using PAC learning of linear separators [41] on simulations of a black-box model of the continuous dynamics [15] .

The algorithm in C2E2 relies on white-box models, where the continuous dynamics are captured by ODEs. In other words, given a vector of continuous state variables $\vec{x} \in \mathbb{R}^n$, its dynamics can be described by $\dot{\vec{x}} = f(\vec{x}(t))$, where $f$ is Lipschitz and possibly nonlinear. In particular, the discrepancy function computation in C2E2 evaluates the condition number of $J(\vec{x})$, the Jacobian matrix of $f$ evaluated at $\vec{x}$. In the case of ill-conditioned matrices, such as what we have in the Passive mode (the $A$-matrix representation of (2.2)), the overapproximation error blows up. Ill-conditioned systems may arise from extremely large and small coefficients appearing together in $J(\vec{x}_0)$, which may be alleviated to an extent by scaling the state variables. In this case, linearization of the system resulted in a singular matrix, so scaling is ineffective. In order to address this problem, we implement an alternative model-based reachset computation method called Parsimonious in SDVTool, which is presented next.

## 3.4 Computing Reachsets using Parsimonious

The simulation-based reachability computation algorithm implemented in SDVTool is named Parsimonious and presented in [42]. Like C2E2, this algorithm requires a white-box model of the continuous system dynamics. Moreover, it is restricted to linear ODEs. Then for a continuous state vector $\vec{x} \in \mathbb{R}^n$, the flow equations are described by: $\dot{\vec{x}}(t) = \tilde{A}(t)\vec{x}(t)$.

For an $n$-dimensional system, $n+1$ simulations are performed. From these simulations, special sets called *generalized star sets*, are generated to represent the exact reachsets. Generalized star sets can efficiently represent a variety of data structures, such as convex polyhedra. Part of the motivation for implementing Parsimonious is to improve the performance and robustness of C2E2, so we choose to restrict our implementation in SDVTool to hyperrectangles for reachset representation, same as C2E2. Thus, we will only introduce the notion of a generalized star set for hyperrectangles.

Let the generalized star set be represented by a pair $\langle \vec{x}_0, V \rangle$, where $\vec{x}_0 \in \mathbb{R}^n$ is the state at the center of the hyperrectangle and $V = \{v_1, ..., v_n\} \subseteq \mathbb{R}^n$ is a standard basis with the vectors scaled to the radius of each dimension of the hyperrectangle. Then the set defined by $\langle \vec{x}_0, V \rangle$ is

$$\{\vec{x} \in \mathbb{R}^n \mid \exists \alpha_1, \ldots, \alpha_n \in [-1, 1], \vec{x} = \vec{x}_0 + \sum_{i=1}^{n} \alpha_i v_i\}.$$

The reachability algorithm will take this representation of the initial set of states of the system and transform the center state and basis, such that a subsequent reachset at time $t_i$ is represented by the generalized star set $\mathcal{R}_i^* = \langle \vec{x}(t_i), V_i \rangle$, with $V_i = \{v_1^i, ..., v_n^i\}$.. These sets are zonotopes but not necessarily hyperrectangles, so they are further over-approximated with hyperrectangles in SDVTool as follows:

$$\mathcal{R}_i = \{\vec{x} \ : \ \vec{x} \leq \vec{x}(t_i) + \sum_{j=1}^{n} \max(-v_j^i, v_j^i)$$
$$\text{and } \vec{x} \geq \vec{x}(t_i) + \sum_{j=1}^{n} \min(-v_j^i, v_j^i)\}.$$

**Algorithm 4:** *computeReachset*: Compute the overapproximation of reachable states from initial set $\theta$ that evolves according to $E_l$ using Parsimonious with hyperrectangles

**Input**: $\theta = \langle \vec{x}_0, V \rangle, T, E_l$
**Output**: $\mathcal{R}(\theta, T)$

$\psi \leftarrow Simulate(\vec{x}_0, T, E_l)$
**for** *each* $v_i \in V$ **do**
  $\psi_i \leftarrow Simulate(\vec{x}_0 + v_i, T, E_l)$
  $\tilde{v}_i \leftarrow \psi_i - \psi$ // Note $v_i \in \mathbb{R}^n$ but $\tilde{v}_i \in \mathbb{R}^{n \times k}$, where there are $k$-simulation points over $T$.
**end**
$\tilde{V} \leftarrow \{\tilde{v}_1, ..., \tilde{v}_n\}$
$\mathcal{R}(\theta, T) \leftarrow \bigcup_{j=1:k} \{\vec{x} \ : \ \vec{x} \leq \psi^j + \sum_{i=1}^{n} \max(-\tilde{v}_i^j, \tilde{v}_i^j)$ and $\vec{x} \geq \psi^j + \sum_{i=1}^{n} \min(-\tilde{v}_i^j, \tilde{v}_i^j)\}$ // $\psi^j = \xi(\vec{x}_0, t_j) \in \psi$ and $\tilde{v}_i^j$ is the $j$-th column of $\tilde{v}_i$
**return** $\mathcal{R}(\theta, T)$

# CHAPTER 4

# SDVTOOL

In this chapter, we will present the details of the implementation of SDVTool. The verification algorithm that SDVTool implements is given in Chapter 3 and the component that distinguishes this tool from existing tools is discussed in Section 3.4.

## 4.1 Architecture and Overview

SDVTool is implemented in Matlab and requires an additional installation of the MPT3 toolbox [43]. The user interacts with the tool by providing an input file that is structured as described in Section 4.2. The implementation of the verification algorithm presented in Chapter 3 is broken up into three main components. The first (grey box in Figure 4.1) corresponds to the high-level procedure given by Algorithm 1. The second (white box in Figure 4.1) is the compute reachtube method in Algorithm 2. Finally the subroutine compute reachset is called within that, which is given in Algorithm 4. Notice in Algorithm 1 that safety is determined by checking the intersection of reachtube $R^T$ with unsafe set $\mathcal{U}$, but in Figure 4.1 safety is checked within the reachtube computation loop. From Algorithm 2 line 14, it is clear that $R^T$ is constructed from the union of all reachset subcomputations $\mathcal{R}(\theta, T)$, so checking the intersection of each $\mathcal{R}(\theta, T)$ with $\mathcal{U}$ is equivalent to checking $R^T \cap \mathcal{U}$.

Dependence on the MPT3 toolbox is introduced in the current version of SDVTool and is motivated by the recent development of another verification tool called CODEV [1], which was introduced to handle a class of hybrid models that use model predictive control (MPC). The original version of

[1] https://bitbucket.org/nchan2/codev

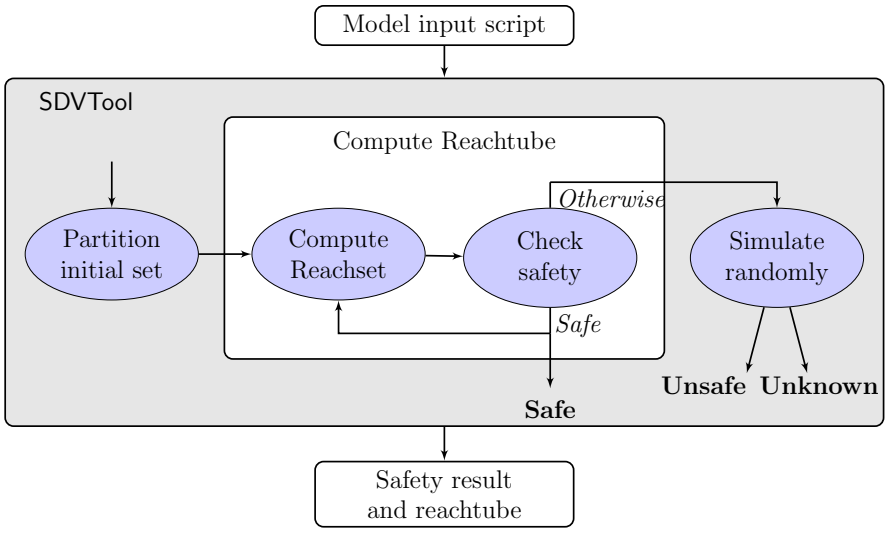



Figure 4.1: Overview of SDVTool's structure.

SDVTool can be obtained from this link [2], but we present the more robust, current version of SDVTool [3] in this thesis.

The POLYHEDRON-data type provided by MPT3 toolbox is used to represent the hyperrectangular reachsets computed. In particular, the output of the program is a flag indicating **Safe** (`safeFlag=1`), **Unsafe** (`safeFlag=0`), or otherwise **Unknown** (`safeFlag=-1`), and a corresponding Reachtube `reach`—a temporally sorted array of POLYHEDRON-objects—that is empty if a safe result cannot be obtained. The output `reach` can be plotted in Matlab as usual: `plot(reach)`, if the model is 2-dimensional. Otherwise, the `projection` method may be used to plot in dimensions $i, j$ as follows: `plot(reach.projection([i,j]))`.

## 4.2 Input Specification

In order to fully specify the safety verification problem, we need the hybrid model definition and additional parameters for the safety verification problem. The input file is set up to return a struct with a function handle `flowEq` to the plant dynamics (i.e. the dynamics that do not change across discrete modes); a struct `MPCsol` (naming inherited from CODEV) that contains the invariant regions and corresponding control laws (i.e. components

[2] https://tinyurl.com/verifysat
[3] https://bitbucket.org/nchan2/sdvtool2

of the dynamics that change across discrete modes); a set of POLYHEDRON objects `unsafeStates` describing the sets of unsafe states; a POLYHEDRON object `initStates` describing the initial set of states for the verification; and a struct `vPar` containing remaining parameters needed for verification (i.e. time horizon and simulation step size).

We illustrate the input structure usage with the example for Lin-SwLQ benchmark model. Let us refer to the output of the input file struct as `inPar`. The dynamics given by Equations 2.2 are specified in `inPar.flowEq`, with an input argument accounting for $F_x, F_y$. This argument is the matrix `inPar.MPCsol.Fi{i}`—$K_i$ from Equation 2.5—which corresponds to the mode given by invariant region defined by POLYHEDRON `inPar.MPCsol.Pn{i}`. Each unsafe set given in Section 2.3 (or its complement) is given as a POLYHEDRON member of the array `inPar.unsafeStates.region`, with a corresponding flag in `inPar.unsafeStates.safe` that is set to 0 when the region is an unsafe set, a subset of $\mathcal{U}$, and to 1 when the region is a safe set. The remaining components define the verification problem: `inPar.initStates` is a POLYHEDRON representing $\Theta$, `inPar.vPar.Thorizon` is an array $[0, T]$, and `inPar.vPar.simStep` is a fixed simulation step size corresponding with $\delta$ from Section 3.1.

## 4.3 Main Verification Routine

The main program is `verify.m`. After parsing the input file, the `poly2ball` routine takes the initial set `inPar.initStates` and set of mode invariants `inPar.MPCsol.Pn` and returns subsets of the initial set which correspond to $\theta_i$ in line 2 of Algorithm 2. We instantiate a Tree object (`CTree`) within the `verify.m` main loop, so as to track the various reachtubes that are generated for each $\theta_i$. This implementation using a tree is inherited from CODEV, where each $\theta_i$ might be further partitioned later in the program execution, but is admittedly unnecessary for SDVTool where the partitioning only occurs once. The remainder of this routine follows Algorithm 1 very closely.

## 4.4 Compute Reachtube Routine

The block for computing reachtubes is implemented in `computeReach.m`. This routine implements Algorithm 2. Again, a tree data structure is used to track the recursive reachset computations necessary for constructing the full reachtube $R^T$. Due to partitioning in the main loop, the input set of initial states to the `computeReach` method in the main loop is always constrained to a single discrete mode. So upon the first execution of line 7 in Algorithm 2, this reachset $R_1 := \mathcal{R}(\theta, T)$ is considered the root of a Tree labeled `RTree`. For each enabled transition, a new reachset is computed (say $R_2, R_3, ...$) by way of adding the subset of initial states resulting from taking the transition to the queue of sets $Q$ (line 11) that will be passed to *computeReachset* (line 6) upon some future iteration of the while-loop. These reachsets $R_2, R_3, ...$ are the children nodes of root $R_1$. Clearly these reachsets may generate their own children.

Thus the Tree data structure allows us to keep track of which reachsets spawned which other reachsets. The pseudocode given in Algorithm 2 uses a queue in line 4 because the ordering of the reachset computations does not matter. From the perspective of the underlying Tree, the problem of when to call *computeReachset* is equivalent to tree exploration. A depth-first search (DFS) is currently implemented in `computeReach`.

We could preserve the most informative model by storing a POLYHEDRON representation for each reachset or node of the tree, but this is not space-efficient so we further overapproximate reachsets by representing the union of sibling sets with one data structure. Storing an actual union of POLYHEDRON-objects results in a cell array of POLYHEDRON, which does not reduce space; so, the convex hull of sibling sets is taken instead and stored in a single POLYHEDRON object—freeing the space the sibling nodes has previously occupied. This is implemented by the `reachUnion` member function of the custom TREENODE class. Once all the reachset subcomputations are completed, the reachtube $R^T$ will look like there is only a single (overapproximated, convex) reachset at each level of the tree. The reachtube $R^T$ looks like an array of POLYHEDRON objects, each corresponding to one of these nodes of the tree.

The *computeReachset* method given in Algorithm 4 is implemented in `computePost.m`. The only difference is an additional call to `checkSafety.m`.

If this result is unsafe, it is passed all the way back to the main loop (setting line 2 of Algorithm 1), ending the reachset/reachtube computations and initiating the random search for a counterexample simulation trace.

# CHAPTER 5

# EXPERIMENTAL RESULTS

In Section 5.1, we will discuss the precise parameters used for the rendezvous mission benchmark models and for the proposed controllers. Then we demonstrate a variety of software verification tools on a subset of the possible benchmark model configurations in Section 5.2.

## 5.1 Setup

### 5.1.1 Mission Parameters

Our benchmark rendezvous mission is preceded and succeeded by other mission phases of the ARPOD formulation in [2]. Our choice of initial set of states and time horizon keeps in line with the full ARPOD scenario. We choose a time horizon of $T = 4$ hr and initial set:

$$\Theta = \{\vec{x} \in \mathbb{R}^4 \mid \exists \alpha_1, \ldots, \alpha_4 \in [-1, 1],\ \vec{x} = \vec{x}_0 + [\alpha_1, \alpha_2, \alpha_3, \alpha_4] * \vec{r},$$
$$\vec{x}_0 = [-900\text{m}, -400\text{m}, 0\text{m/s}, 0\text{m/s}]^T,\ \vec{r} = [25\text{m}, 25\text{m}, 0\text{m/s}, 0\text{m/s}]^T\}.$$

This gives an initial separation distance that is approximately 1000 m (the transition guard defined in [2] from the preceding mission stage,) and a reasonable relative angle $\tan^{-1}(\frac{y}{x})$ because the initial position given for the full ARPOD mission starts at a position along the $-\hat{\mathbf{j}}$-direction. The transitions between mission phases do not rely on the relative velocity of the chaser, so we choose to start the rendezvous phases from a zero relative velocity. The radius of the hyperrectangle $\Theta$ can be interpreted as having bounded uncertainty in the chaser's initial position but the chaser's initial velocity is precisely fixed. We use experimental data from [2] to set a reasonable time bound for the rendezvous portion of the total ARPOD mission.

For the Pass mission model configuration, we choose a small interval at

[120, 125 min] for the transition guard $[t_1, t_2]$ in Figure 2.6. We experimentally observe this is an interval that ensures the chaser satellite will transition to Mode 2 before transitioning to Passive in that particular model (Lin-SwLQ-Pass).

### 5.1.2 Unsafe Sets

In Section 2.3, we described the mission constraints and the need to approximate these with affine inequalities. Table 5.1 gives these precise affine approximations. Furthermore, they are separated so that the union of each of these sets describes the unsafe sets of states. Each of these properties are checked independently of one another in the software tools because most of the tools only check one convex polytope at a time.

Table 5.1: Description of each unsafe property that results from the mission constraints presented in Section 2.3.

| **Property** | **Description** | **Prop.** | **Description** |
|---|---|---|---|
| Thrust1 | $F_x < -10$ | VEL1 | $-\dot{x} < -3$ |
| Thrust2 | $-F_x < -10$ | VEL2 | $\dot{x} < -3$ |
| Thrust3 | $F_y < -10$ | VEL3 | $-\dot{y} < -3$ |
| Thrust4 | $-F_y < -10$ | VEL4 | $\dot{y} < -3$ |
| LOS1 | $x < -100$ | VEL5 | $-\dot{x} - \dot{y} < -3\sqrt{2}$ |
| LOS2 | $x \tan(30°) - y < 0$ | VEL6 | $-\dot{x} + \dot{y} < -3\sqrt{2}$ |
| LOS3 | $x \tan(30°) + y < 0$ | VEL7 | $\dot{x} - \dot{y} < -3\sqrt{2}$ |
| PASS | $(x + y < 1) \cap (x - y < 1) \cap$ $(-x - y < 1) \cap (-x + y < 1)$ | VEL8 | $\dot{x} + \dot{y} < -3\sqrt{2}$ |

### 5.1.3 Controller Parameters

In Section 2.4, we presented the framework for constructing the SwLQ controller. Here, we give the particular values of the cost functions from Equation 2.3 we used in our experiments, with units for $x, y$ in [m], $\dot{x}, \dot{y}$ in [m/min],

and $F_x, F_y$ in [kg·m/min$^2$]:

$$Q_1 = \begin{bmatrix} \frac{1}{1000^2} & 0 & 0 & 0 \\ 0 & \frac{1}{866^2} & 0 & 0 \\ 0 & 0 & \frac{1}{20^2} & 0 \\ 0 & 0 & 0 & \frac{1}{20^2} \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} \frac{1}{100^2} & 0 & 0 & 0 \\ 0 & \frac{1}{100^2} & 0 & 0 \\ 0 & 0 & \frac{1}{3^2} & 0 \\ 0 & 0 & 0 & \frac{1}{3^2} \end{bmatrix}$$

$$R_1 = R_2 = \begin{bmatrix} \frac{1}{28800^2} & 0 \\ 0 & \frac{1}{28800^2} \end{bmatrix}.$$

Using Matlab's `lqr` function, the following control law gains are obtained:

$$K_1 = \begin{bmatrix} 28.83 & -0.10 & 1449.98 & -0.005 \\ 0.08 & 33.26 & -0.005 & 1451.50 \end{bmatrix}$$

$$K_2 = \begin{bmatrix} 288.03 & -0.13 & 9614.99 & 0 \\ 0.13 & 288.0 & 0 & 9615.0 \end{bmatrix}.$$

In Section 2.5, we presented the construction of the novel SDLQ controller. Equation 2.7 gave the parameterized cost function weights in $Q$. For the remaining cost function weight $R$, we choose a constant matrix as follows:

$$R = \begin{bmatrix} \frac{1}{36000^2} & 0 \\ 0 & \frac{1}{36000^2} \end{bmatrix}.$$

Recall a new LQR control law is computed every $\delta$-time (see transition in Figure 2.7). We choose $\delta = 0.5$ min. This LQR computation is also completed using Matlab's `lqr` function.

### 5.1.4 Explicit Variables

By our choice of full-state feedback controllers, $F_x, F_y$ (or $\vec{u}$) become linear functions of $\vec{x}$, so there are no explicit variables modeling $F_x, F_y$ in the proposed hybrid benchmark models as observed in Figures 2.4-2.7. In other words, the following systems describe equivalent behaviors:

1. $\dot{\vec{x}} = A\vec{x} + B\vec{u},\ \vec{u} = -K\vec{x}$,

2. $\dot{\vec{x}} = (A - BK)\vec{x}$,

and we have focused on that given in system 2, since we are working with flow equations or trajectories that are described by ODEs.

Our verification problem involves constraints on $\vec{u}$. In order to check these properties in the software tools, the constraints would take the form of affine inequalities $-K\vec{x} < c$. However, the syntax of C2E2 is limited to 1- or 2-dimensional affine inequalities. Thus we revise the dynamical system described by system 1 to be:

$$\dot{\vec{x}} = A\vec{x} + B\vec{u},$$
$$\dot{\vec{u}} = -K\dot{\vec{x}} = -KA\vec{x} - KB\vec{u}.$$

This is equivalent to the original system 1 given the appropriate initial conditions. This gives an augmented 6-dimensional state vector $\vec{\tilde{x}} = [x, y, \dot{x}, \dot{y}, F_x, F_y]$. This augmented system is required to verify thrust constraints in C2E2. While it is not required in SpaceEx, the 6-dimensional model may still be used in SpaceEx to obtain reach sets for the $F_x, F_y$ variables.

## 5.2 Experiments

In this thesis, we only explore the application of a subset of verification tools to a subset of the proposed benchmark satellite rendezvous models. In particular, Table 5.2 describes the possible model configurations and which tools we have applied to each.

### 5.2.1 Experiments for the SwLQ Controller

We use the Lin-SwLQ model to compare tool performance between SDVTool, C2E2, and SpaceEx. First, all the tools returned a **Safe** result for all the properties in Table 5.1 except for PASS (corresponding with collision avoidance during the Passive mode), under the parameters given in Sections 5.1.1 and 5.1.3. Note that C2E2 and SpaceEx re-execute their verification algorithms

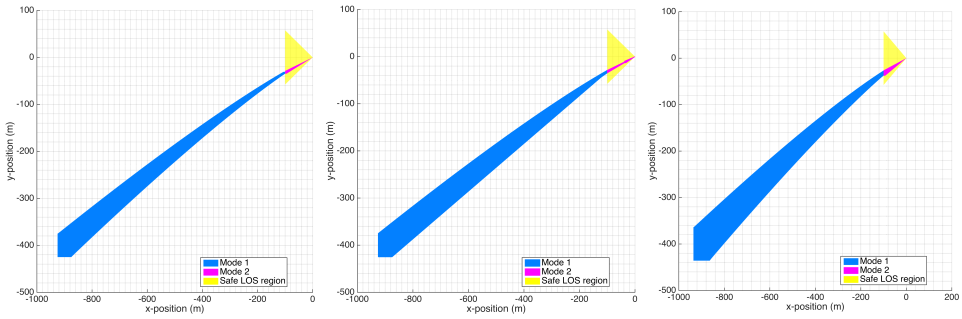
Table 5.2: Summary of the experimental setups (model configurations and software tools) tested in this thesis.

| **Tool** | SwLQ | SwLQ-Pass | SDLQ | SDLQ-Pass |
|---|---|---|---|---|
| SDVTool (Lin) | ✓ | ✓ | | |
| SpaceEx (Lin) | ✓ | ✓ | | |
| C2E2 (Lin) | ✓ | | | |
| C2E2 (NLin) | ✓ | | | |
| DryVR (Lin) | | | ✓ | |
| DryVR (NLin) | | | | |

for *each* property, whereas SDVTool checks all properties in one execution of the program. Table 5.3 summarizes the runtimes taken to obtain these results. The runtime listed for a property, say LOS, is taken to be the average runtime to check each sub-property, e.g. LOS1, LOS2, and LOS3. The reach-tubes returned by each tool are plotted for the relative position variables $x, y$ in Figure 5.1.

Table 5.3: Time (seconds) taken to check each safety property across different tools. "N/V" indicates that the property was not successfully checked.

| **Tool** | **Thrust** | **LOS** | **VEL** | **PASS** | **All** |
|---|---|---|---|---|---|
| SDVTool | | | | | 5.5 |
| SpaceEx | 6.2 | 6.4 | 6.5 | 6.5 | |
| C2E2 | 85 | 20 | 17.8 | N/V | |




(a) (b) (c)

Figure 5.1: Reachable positions for rendezvous using SwLQ without any transition to Passive as computed by: (a) SDVTool (Lin), (b) SpaceEx (Lin), and (c) C2E2 (NLin).

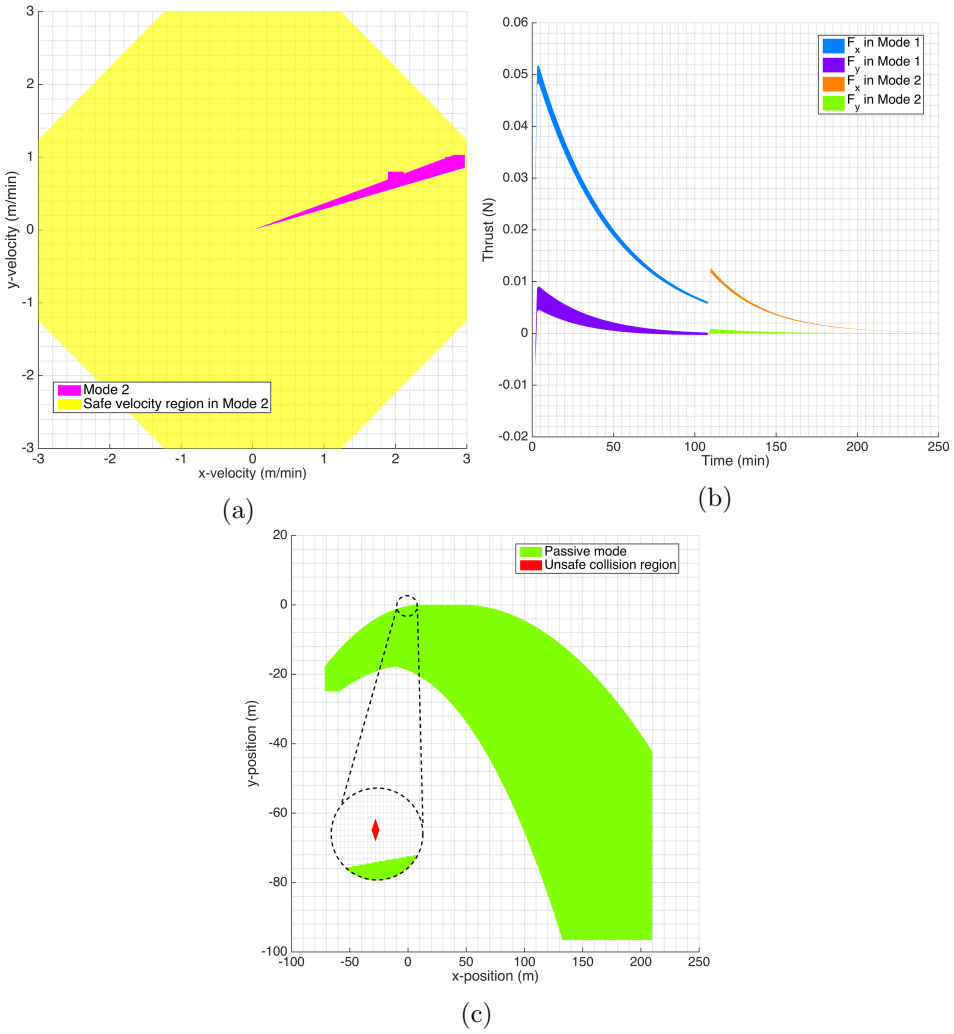



Figure 5.2: Reachsets computed by SDVTool under the Lin-SwLQPass model for: (a) velocity in Mode 2, (b) thrust, and (c) positions reached after a transition to Passive during [120,125 min].

The Lin-SwLQ-Pass model is verified to be **Safe** by both SDVTool and SpaceEx. The reachtubes look similar so we only include the plots from SDVTool to show what the reachable states look like in other dimensions in Figure 5.2.

In C2E2, checking Pass terminates in an **Unknown** result (due to the state-space explosion problem mentioned in Section 3.3). However, we successfully show for both Lin and NLin models without Pass that the remaining safety properties are achieved. The reachtube for relative position using the NLin model is given in Figure 5.1c to further demonstrate that the linear model provides a good approximation of the nonlinear dynamics in this benchmark.

Lastly we present an experiment that focuses on evaluating the limitations of the controller with respective to passive safety. We use SDVTool to run successive tests on the Lin-SwLQ-Pass model for all the safety properties, changing only the initial set of states and the transition guard to Passive. Due to the parameters given by the encompassing ARPOD mission in [2], we expect our rendezvous mission to start at Mode 1 with a separation distance around 1000 m and at an angle in the third quadrant of the relative coordinate frame. We maintain the same assumptions as before that the initial velocities are zero.

This resulting set of initial states $\Theta$ (see the colored arc in Figure 5.3) is partitioned into smaller sets $\theta_1, \theta_2, ...$, such that $\cup_i \theta_i = \Theta$. For each initial set $\theta_i$, we run the verification program $N$ times. On the $j^{th}$ run, the Passive transition guard is set to $[t_1, t_2] = [t_{j-1}, t_j]$ and $t_0 = 0$, $t_N = 270$ min. In particular we choose 5 min increments so $N = 54$. The results give us an idea of how long (within 5 minutes) a rendezvous mission from an initial state (within the neighborhood of its respective $\theta_j$) can still safely initiate a passive abort. Should a failure occur beyond this elapsed time, an active abort would be necessary (or a different control strategy other than the SwLQ presented here).

Figure 5.3 depicts the results of this experiment by assigning a color to the largest value of $t_j$ for which the algorithm returns a safe result. For example, if we look at the initial state $[-900, -400, 0, 0]$ or a small neighborhood of this point, the color is a deep red indicating that if we perform rendezvous starting from this initial state, we can successfully rendezvous *and* abort the mission at any time before the 270 min bound tested. If we look at a neighborhood around $[-200, -900, 0, 0]$, the color is a light blue indicating that we can only guarantee that the rendezvous can safely abort at any time before 100 min. If a transition to Passive occurs after 100 min, this may result in a violation of the passive safety property *or* a rendezvous property (e.g. max thrust) may be violated after 100 min.

### 5.2.2 Experiments for the SDLQ Controller

For the SDLQ controller, we do not have an explicit closed-form model description though we are able to represent it compactly in Figure 2.7. The

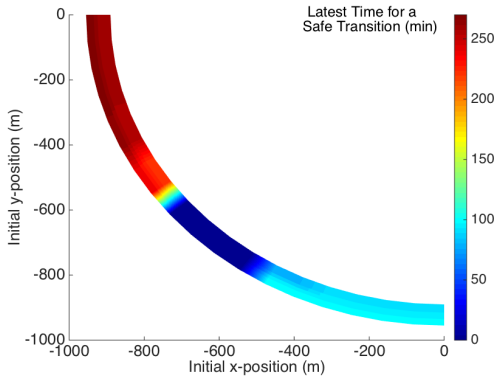


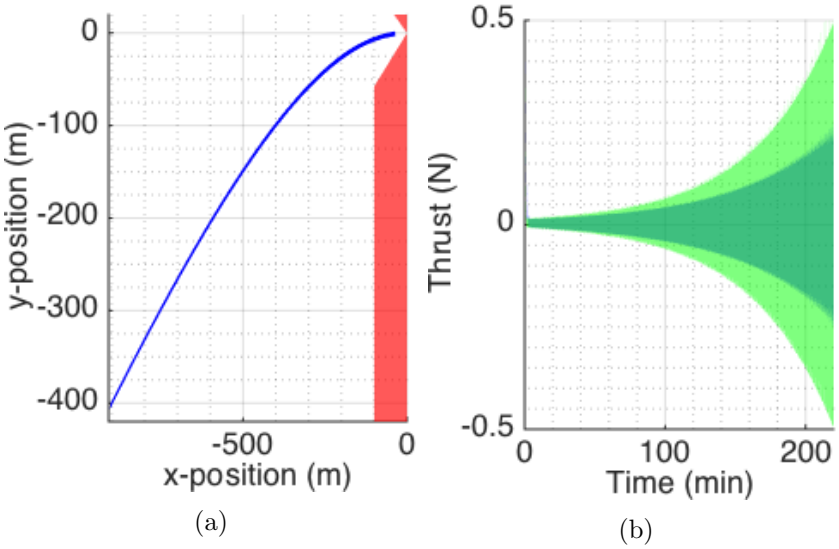

Figure 5.3: Initial positions (with zero initial velocities) of Lin-SwLQ-Pass that have been verified by SDVTool to be safe. They are safe for Passive transition times up to the time shown by the color map.

update function here "rewrites" the model description (the flow equations) periodically. The tools we have discussed thus far (SDVTool, C2E2, and SpaceEx) cannot handle models of this nature. However, we presented DryVR in Section 3.3 and noted that it did not rely on such white-box models, but could handle black-box models given a simulation artifact—we have implemented the simulation of the hybrid system in Figure 2.7 in Matlab.

Running the simulation artifact for the Lin-SDLQ model through DryVR provided **Safe** results (with reachtubes shown in Figure 5.4) for the thrust and LOS properties, and an **Unsafe** result for the total velocity (i.e. provided a counterexample simulation trace shown in Figure 5.5). This unsafe result is not surprising as the parameterized cost function proposed in Equation (2.7) was designed to enforce the LOS constraint and not necessarily the velocity constraint (notice dependence on $x, y$ but not on $\dot{x}, \dot{y}$).

The reachtubes obtained from checking thrust and LOS properties are plotted along the relative position dimensions and for thrust in each direction in Figure 5.4. The runtimes for these tests were on the order of hours. While this is not immediately practical, it establishes the feasibility of the approach and motivates more careful engineering and parallelization. The primary source of this long overhead is due to the frequent re-computation of the SDLQ control law. This slows down a single simulation run and DryVR requires the generation of multiple simulation traces. In DryVR, the random simulation search for counterexamples (lines 3-9 in Algorithm 1) occurs concurrently within the reachtube computation (line 1), so running the ver-

ification program for the total velocity (VEL) constraints completed on the order of seconds.




(a) (b)

Figure 5.4: (a) Reachable positions (blue) and unsafe positions (red). (b) Reachable thrusts: $F_x$ (blue) and $F_y$ (green).

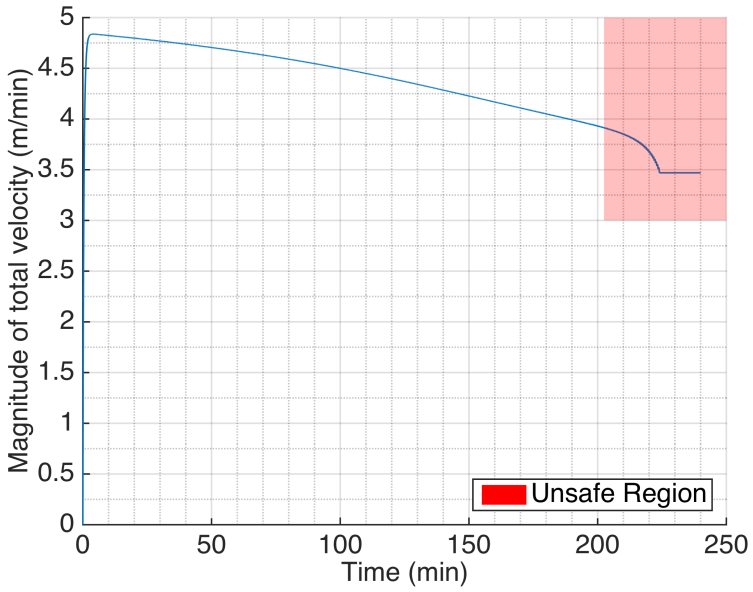



Figure 5.5: Simulated trajectory of total velocity starting from a state in $\Theta$ that violates the velocity constraint (red).

We additionally check the correctness of the discrepancy function learned in DryVR, that is, whether the learned function meets the property of dis-

crepancy functions stated in Equations (3.2). For this experiment, we randomly sample the initial states in $\Theta$, generate their simulation traces, and check whether any pair of these traces provide a counterexample to Equations (3.2). We do this for 600 samples and find no such counterexamples; thus, we conclude that the discrepancy function computed by DryVR in this example holds with high confidence.

Lastly, we perform an experiment to evaluate the performance of the SDLQ controller (rather than its capability to achieve safety) by running simulations using SDLQ, SwLQ, and a model predictive controller (MPC) that was used in the original benchmark formulation paper [2]. Specifically, we use the realistic performance metric of fuel consumption to compare performance between controllers. In this case, fuel consumption is taken to be:

$$J(T) = \int_0^T ||\vec{u}(t)||dt.$$

These simulations are performed under the same initial conditions and a terminating condition of achieving $\rho = 20$ m (which occurs at different times $t_f$ and terminal states $\vec{x}_{t_f}$ for each controller). The results are given in Table 5.4, and a plot of the cumulative fuel expended is in Figure 5.6. MPC minimizes the closing velocity best without sacrificing too much in completion time and fuel consumption. The SwLQ achieves the best completion time but at a high cost in fuel consumption. Finally, SDLQ achieves the best fuel costs with acceptable increase in completion time but unacceptable ranges of velocity that violate constraints.

Table 5.4: Comparison of terminal states, completion times, and total fuel cost.

| | $\vec{x}_{t_f}$ | $t_f$ | $J(t_f)$ |
|---|---|---|---|
| SDLQ | [-19.99, -0.23, 3.52, 0.09] | 223.7min | 168.6N |
| SwLQ | [-19.09, -5.97, 0.57, 0.18] | 166.4min | 531.8N |
| MPC | [-19.91, -0.01, 0.003, 0.0] | 180.0min | 213.0N |

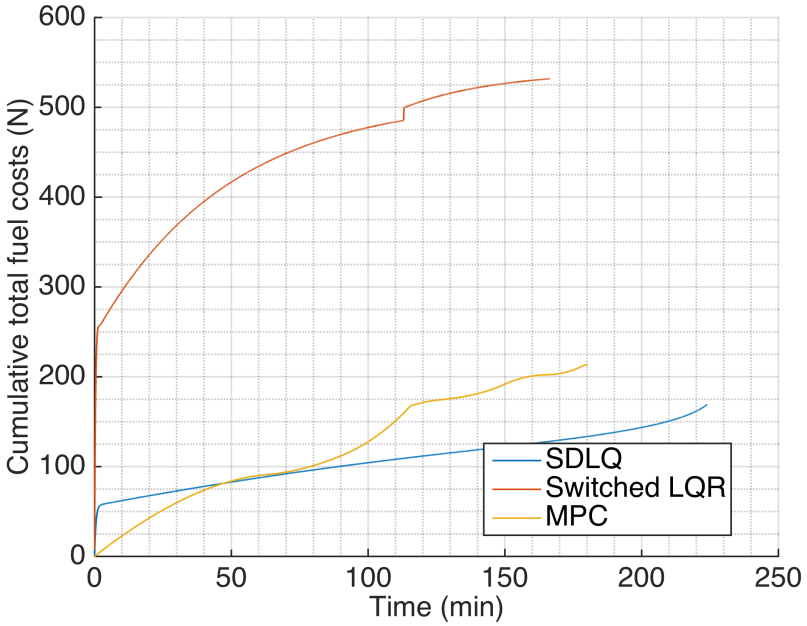



Figure 5.6: Cumulative fuel consumption for three different controllers obtained through simulation under same initial state and termination condition.

# CHAPTER 6

# CONCLUSIONS

In this thesis, we present a set of linear (Lin) and nonlinear (NLin), possibly nondeterministic (if Pass) benchmark models of an autonomous rendezvous maneuver for spacecraft with several safety requirements. We designed a switched LQR (SwLQ) controller and verified its safety across a subset of the provided models, a variety of initial conditions and parameter ranges, and using three different software tools (SDVTool, C2E2, and SpaceEx) for bounded model-checking of hybrid systems. This particular subset of experiments involving the passive abort maneuver (Pass) has shed light on the weakness of simulation-driven verification in handling ill-conditioned models, and motivated the development of SDVTool—a tool intended to test an alternative reachability computation algorithm that could improve the original tool in question: C2E2.

We also proposed a novel controller, the state-dependent LQ (SDLQ) control, which is a piecewise continuous state-feedback controller obtained by periodically recomputing a quadratic optimization problem that implicitly enforces system constraints. The design framework involves choosing an appropriate state-based function for the quadratic objective function, and then verifying that the resulting closed-loop system does not violate safety constraints. However this model proved to be intractable under these previously mentioned software verification tools. This provided the opportunity to demonstrate and highlight the advantages of a more recently developed data-driven reachability analysis tool, DryVR.

The results provide a foundation for verifying more sophisticated maneuvers (including the extended ARPOD mission) in future autonomous space operations. However, there are still many threads to address before obtaining a fully mature, safe controller for such applications. For example, we proposed a state feedback controller, but we ought to consider a situation where full state measurement is not possible, such as is the case for angles-

only rendezvous from the full ARPOD problem. Here, a simple bang-bang controller could be used but would result in unstable behavior similar to what was observed in the Passive mode in C2E2 (the resulting closed-loop system gives the same Jacobians which are used to compute discrepancy in this tool). The autonomous rendezvous problem can easily be made more challenging and realistic, which then prompts innovation and application of problems in observability, optimal and robust control, and the like. There is a concurrent need to address algorithmic verifiability of these sophisticated, innovative control solutions. C2E2 is one example of the ongoing effort to cover a large, growing class of nonlinear models, but we have already observed a facet of its limitations with the SDLQ models. It is clear that DryVR shows much promise in significantly extending the classes of models that can be automatically verified. It remains to be seen how we can apply rigorous safety-checking to optimal, constrained control schemes, such as model predictive control (e.g. if optimization constraints are relaxed). This safety-checking is especially important in the context of autonomous spacecraft navigation, as it poses a difficult control problem and its proposed solutions can become increasingly complex and creative.

# REFERENCES


[1] *NASA Space Technology Roadmaps and Priorities.* The National Academies Press, May 2012.

[2] C. Jewison and R. S. Erwin, "A spacecraft benchmark problem for hybrid control and estimation," in *2016 IEEE 55th Conference on Decision and Control (CDC).* IEEE, Dec 2016. [Online]. Available: https://doi.org/10.1109/cdc.2016.7798765

[3] D. Woffinden and D. Geller, "Navigating the road to autonomous orbital rendezvous," *Journal of Spacecraft and Rockets*, vol. 44, no. 4, pp. 898–909, 2007.

[4] D. Pinard, S. Reynaud, P. Delpy, and S. E. Strandmoe, "Accurate and autonomous navigation for the ATV," *Aerospace Science and Technology*, vol. 11, no. 6, pp. 490–498, Sep 2007. [Online]. Available: http://dx.doi.org/10.1016/j.ast.2007.02.009

[5] D. Zimpfer, P. Kachmar, and S. Tuohy, "Autonomous rendezvous, capture and in-space assembly: past, present and future," in *Proc. AIAA Space Exploration Conference*, Jan. 2005.

[6] K. Galabova, G. Bounova, O. de Weck, and D. Hastings, "Architecting a family of space tugs based on orbital transfer mission scenarios," in *AIAA Space 2003 Conference & Exposition.* American Institute of Aeronautics and Astronautics (AIAA), Sep 2003. [Online]. Available: http://dx.doi.org/10.2514/6.2003-6368

[7] J. A. Starek, B. Açıkmeşe, I. A. Nesnas, and M. Pavone, *Spacecraft Autonomy Challenges for Next-Generation Space Missions.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 1–48.

[8] NASA, "Overview of the dart mishap investigation results," Tech. Rep., 2006.

[9] W. E. Wong, V. Debroy, and A. Restrepo, "The role of software in recent catastrophic accidents," IEEE Reliability Society 2009 Annual Technology Report.

[10] G. J. Holzmann, "Mars code," *Commun. ACM*, vol. 57, no. 2, pp. 64–73, Feb. 2014. [Online]. Available: http://doi.acm.org/10.1145/2560217.2560218

[11] "Terminal guidance system for satellite rendezvous," *Journal of the Aerospace Sciences*, vol. 27, no. 9, pp. 653–658, Sep 1960. [Online]. Available: http://dx.doi.org/10.2514/8.8704

[12] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: A verification tool for stateflow models," in *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, 2015, pp. 68–82.

[13] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala, "Automatic reachability analysis for nonlinear hybrid models with C2E2," in *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-41528-4_29 pp. 531–538.

[14] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *CAV*, 2011, pp. 379–395.

[15] C. Fan, B. Qi, S. Mitra, and M. Viswanathan, "DRYVR: data-driven verification and compositional reasoning for automotive systems," *CoRR*, vol. abs/1702.06902, 2017. [Online]. Available: http://arxiv.org/abs/1702.06902

[16] S. A. Jacklin, "Survey of verification and validation techniques for small satellite software development," NASA Ames Research Center, Space Tech Expo, May 2015. [Online]. Available: http://ntrs.nasa.gov/search.jsp?R=20150010982

[17] M. Bozzano, R. Cavada, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, and X. Olive, "Formal verification and validation of aadl models," *Proc. ERTS*, 2010.

[18] T. T. Johnson, J. Green, S. Mitra, R. Dudley, and R. S. Erwin, "Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems," in *FM 2012: Formal Methods*. Springer Nature, 2012, pp. 252–266. [Online]. Available: https://doi.org/10.1007%2F978-3-642-32759-9_22

[19] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech," in *Hybrid Systems: Computation and Control*. Springer Nature, 2005, pp. 258–273. [Online]. Available: https://doi.org/10.1007%2F978-3-540-31954-2_17

[20] S. S. Farahani, I. Papusha, C. McGhan, and R. M. Murray, "Constrained autonomous satellite docking via differential flatness and model predictive control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, Dec 2016. [Online]. Available: https://doi.org/10.1109/cdc.2016.7798766

[21] B. P. Malladi, R. G. Sanfelice, E. Butcher, and J. Wang, "Robust hybrid supervisory control for rendezvous and docking of a spacecraft," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. Institute of Electrical and Electronics Engineers (IEEE), Dec. 2016. [Online]. Available: https://doi.org/10.1109%2Fcdc.2016.7798769

[22] B. HomChaudhuri, M. Oishi, M. Shubert, M. Baldwin, and R. S. Erwin, "Computing reach-avoid sets for space vehicle docking under continuous thrust," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. Institute of Electrical and Electronics Engineers (IEEE), Dec. 2016. [Online]. Available: https://doi.org/10.1109%2Fcdc.2016.7798767

[23] N. Chan and S. Mitra, "Verifying safety of an autonomous spacecraft rendezvous mission," in *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, ser. EPiC Series in Computing, G. Frehse and M. Althoff, Eds., vol. 48. EasyChair, 2017. [Online]. Available: https://easychair.org/publications/paper/S2V pp. 20–32.

[24] N. Chan and S. Mitra, "Verified hybrid lq control for autonomous spacecraft rendezvous," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec 2017, pp. 1427–1432.

[25] N. Chan and S. Mitra, "Codev: Automated model predictive control design and formal verification," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, ser. HSCC '18. New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3178126.3187003 pp. 281–282.

[26] M. A. Johnson and M. J. Grimble, "Recent trends in linear optimal quadratic multivariable control system design," *IEE Proceedings D - Control Theory and Applications*, vol. 134, no. 1, pp. 53–71, January 1987.

[27] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, *The Theory of Timed I/O Automata (Synthesis Lectures in Computer Science)*. Morgan & Claypool Publishers, 2006.

[28] E. Hainry, "Reachability in linear dynamical systems," in *Proceedings of the 4th Conference on Computability in Europe: Logic and Theory of Algorithms*, ser. CiE '08. Berlin, Heidelberg: Springer-Verlag, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-69407-6_28 pp. 241–250.

[29] G. Lafferriere, G. J. Pappas, and S. Sastry, "O-minimal hybrid systems," *Mathematics of Control, Signals and Systems*, vol. 13, no. 1, pp. 1–21, Feb 2000. [Online]. Available: https://doi.org/10.1007/PL00009858

[30] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94 – 124, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000098915811

[31] A. Donzé and O. Maler, "Systematic simulation using sensitivity analysis," in *Hybrid Systems: Computation and Control*, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 174–189.

[32] P. S. Duggirala, S. Mitra, and M. Viswanathan, "Verification of annotated models from executions," in *Proceedings of the Eleventh ACM International Conference on Embedded Software*, ser. EMSOFT '13. Piscataway, NJ, USA: IEEE Press, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2555754.2555780 pp. 26:1–26:10.

[33] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 116–126, Jan 2010.

[34] C. Fan and S. Mitra, "Bounded verification with on-the-fly discrepancy computation," in *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Shanghai, China, October 12-15, 2015, Proceedings*, 2015. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-24953-7_32 pp. 446–463.

[35] P. S. Duggirala and M. Viswanathan, *Parsimonious, Simulation Based Verification of Linear Systems*. Cham: Springer International Publishing, 2016, pp. 477–494.

[36] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 167–170.

[37] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, June 1998. [Online]. Available: http://dx.doi.org/10.1016/S0005-1098(98)00019-3

[38] A. Kanade, R. Alur, F. Ivančić, S. Ramesh, S. Sankaranarayanan, and K. C. Shashidhar, "Generating and analyzing symbolic traces of simulink/stateflow models," in *Computer Aided Verification*, A. Bouajjani and O. Maler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 430–445.

[39] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas, "Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '10. New York, NY, USA: ACM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1755952.1755983 pp. 211–220.

[40] E. M. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *Automated Technology for Verification and Analysis*, T. Bultan and P.-A. Hsiung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–12.

[41] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*. MIT press, 1994.

[42] P. S. Duggirala and M. Viswanathan, "Parsimonious, simulation based verification of linear systems," in *Computer Aided Verification*, S. Chaudhuri and A. Farzan, Eds. Cham: Springer International Publishing, 2016, pp. 477–494.

[43] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, http://control.ee.ethz.ch/ mpt. pp. 502–510.