AUTOMATIC SIMULATION-DRIVEN REACHABILITY USING MATRIX MEASURES

BY

CHUCHU FAN

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, 2016

Urbana, Illinois

Adviser:

Associate Professor Sayan Mitra

ABSTRACT

Simulation-driven verification is a promising approach that provides formal safety guarantees for otherwise intractable nonlinear and hybrid system models. A key step in simulation-driven algorithms is to compute the reach set over-approximations from a set of initial states through numerical simulations. This thesis introduces algorithms for this key step, which relies on computing piece-wise exponential bounds on the rate at which trajectories starting from neighboring states converge or diverge. We call this *discrepancy function*. The algorithms rely on computing local bounds on the matrix measure of the Jacobian matrices. We discuss different techniques to compute the matrix measures under different norms: regular Euclidean norm or Euclidean norm under coordinate transformation, such that the exponential rate of the discrepancy function is locally minimized. The proposed methods enable automatic reach set computations of general nonlinear systems and have been successfully used on several challenging benchmark models. All proposed algorithms for computing discrepancy function give soundness and relative completeness of the overall simulation-driven safety verification algorithm. We present a series of experiments to illustrate the accuracy and performance of the approach.

To my parents and husband, for whom my love is beyond verification.

ACKNOWLEDGMENTS

I would first like to express my thanks to my adviser, Professor Sayan Mitra of the ECE Department at UIUC, who is not only my thesis advisor but also a mentor for many aspects of my life. His passion for research and devotion to guiding students have motivated me to move firmly along the road of computer engineering. Without his constant help, this work would never be possible.

I would also like to thank Doctor Zhenqi Huang for providing so much support in my early graduate life. He is like an elder brother, who always helps me out when I am stuck in research. His penetrating insights on control theory and verification were valuable guidance for me to explore the unknown.

It has been a great honor to work with Professor Mahesh Viswanathan, Professor Parasara Sridhar Duggirala, Mr. Bolun Qi, Mr. Matthew Potok and Mr. Suket Karnawat on the verification tool. I appreciate very much the wonderful days I spent with Dr. Xiaoqing Jin, Dr. Jim Kapinski and Dr. Jyotirmoy Deshmukh at the Toyota Technical Center. I am also very grateful to have my labmates and friends Ritwika Ghosh, Nicole Chan, Hussein Sibaie, Yixiao Lin, Debjit Pal and Rui Jiang with whom I have spent the past years together. In addition, my sincere thanks also go to Carol Wisniewski for the generous help she provides.

Many thanks to my parents for their unconditional love and sacrifice. To be someone they are proud of is my most powerful motivator. I would also like to thank my husband, Qiang, for kicking out sadness and sorrows from my life, and making me complete.

Finally, I wish to thank the National Science Foundation of the United States for providing financial support for my research.

Table of Contents

Chapter 1

INTRODUCTION

1.1 Need for Simulation-driven Verification

The 21st century has witnessed phenomenal growth in cyber-physical systems (CPS), which tightly couple physical processes with software, networks, and sensing. Most cars today come with cruise control which helps maintain the speed. In the near future, the vehicles will be equipped with automatic braking and steering systems to avoid collisions. Finally, fully self-driving cars are also being built to improve road safety and efficiency. Unmanned aerial vehicles are starting to share increasingly crowded airspace with commercial and passenger air traffic, autonomous satellites will soon coordinate with one another and service aging satellites, networked medical devices are being implanted for health monitoring and drug delivery. Reliability and security lapses of such cyber-physical systems routinely disrupt communities, and in many occasions have led to catastrophic failures, with major damage to infrastructure and people. For example, a software glitch in the Toyota Prius is the prime cause for a massive recall of 1.9 million cars, which caused significant monetary loss and damage to the reputation of the company.

Recent breakthroughs in verification techniques have shown great potential for improving the design and certification processes for cyber-physical systems [1, 2, 3, 4, 5]. Early stage successful applications of the verification techniques can be found in automotive power-train control systems [6, 7], medical devices [8, 9], aerospace brake systems [10] and power plants [11].

The two predominant approaches for enhancing the reliability and safety of CPS are based on dynamic analysis and static analysis. The former usually generates traces from the models or the real systems to find possible defects. It is computationally inexpensive and therefore is popular in industry. However, it suffers from incompleteness: it is impossible to cover all possible

(uncountably infinite) behaviors of the system in finite time using finite number of traces. Static analysis, on the other hand, analyzes the formal models of the system to infer properties about all possible behaviors. This is done either by computing (or approximating) the reachable states or by deducing properties of the model using specific proof rules. Wherever feasible verification provides powerful system-level reliability guarantees, however, most verification techniques for cyber-physical systems do not scale to large, realistic models because reachability computations suffer from the state-space explosion problem and the deductive approaches require significant human insight in constructing the right proof obligations.

Recently, a third way of enhancing reliability has gained traction [5, 12, 13, 14, 15, 16]. This uses simulation traces for providing system-level reliability guarantees by appropriately generalizing an individual trace (generated from a test or a simulation) to a set of executions and then verifying the property for this generated set. The success of this approach hinges on good generalizations which can lead to coverage of all possible behaviors from a finite sequence of traces. In the context of cyber-physical systems [5, 13, 14], this generalization has been achieved by exploiting the smoothness of the continuous dynamics of a system. The scalability of simulating complex non-linear dynamics has enabled this approach to successfully verify interesting classes of hybrid systems.

1.2 Main Results: Soundness, Relative Completeness, and Precision

Consider a dynamical system $\dot{x} = f(x)$ and its solution $\xi(x_0, t)$ starting from the initial state x_0 (please see Section 2.2 for more details). Following the simulation-driven reachability approach of [5, 17, 18], we over-approximate the reachable states of the system by first computing a numerical simulation of $\xi(x, t)$ from a specific initial state x_0, and then symbolically computing a *reachtube* from this simulation that contains all solutions starting from a neighborhood of x_0 that contains the initial set. Next we will check the intersection of the over-approximation reachtube with the unsafe region, decide if there is no intersection or if there exists a counter-example, or if neither is true, then start over from a smaller neighborhood around the initial set. The

bottleneck for this approach is the algorithms to compute the reachtube, that is, to compute the over-approximation of the set of states that are reachable from a set of initial states for nonlinear systems using simulations.

Consider a function β that provides upper bounds on the distance between the trajectories at any time, that is, $\|\xi(x_1, t) - \xi(x_2, t)\| \leq \beta(\|x_1 - x_2\|, t)$, for any state x_1, x_2 in the state space. If we use the function to *bloat* the simulation to get the reachtube, the reachtube is then guaranteed to contain all the reachable states of the system. We call such a function that bounds on the distance between any two trajectories a *discrepancy function*. There are several concepts that are related to discrepancy function, for example, sensitivity analysis [2], incremental Lyapunov functions [19], contraction metrics [20], matrix measures [21] etc. However, finding the discrepancy function automatically for general nonlinear systems is still a challenging problem.

Inspired by [21, 20], an upper bound c on the matrix measure of the system's Jacobian matrix $J_f(x)$ can be used as an exponential upper bound on the distance between neighboring trajectories. Closed-form expressions for matrix measures are in general difficult to obtain for nonlinear systems. For example, for matrix A, the matrix measure under Euclidean norm is the largest eigenvalue of the symmetric part of the matrix $\lambda_{\max}((A + A^T)/2)$. However, if we can over-approximate all possible values of the system's Jacobian matrix $J_f(x)$, we can obtain an upper bound on the matrix measure of the Jacobian matrix without knowing its closed form. This is achievable because the Jacobian matrix is a function of states and the procedure is: (a) use constant interval matrices to bound the variation of the Jacobian matrix over a small part of the state spaces, (b) compute the upper bound of the matrix measure of the interval matrix. In this thesis, we will introduce two algorithms `LDF2` and `LDFM` to compute the upper bound of the matrix measure of the interval matrix.

The two different algorithms will focus on two different matrix measures. `LDF2` computes the matrix measure under 2-norm, i.e., the largest eigenvalue of the symmetric part of the Jacobian matrix. We will use a matrix perturbation theorem to transform the problem of bounding the largest eigenvalue to bounding the 2-norm of a matrix valued function. The algorithm is more efficient (faster) than the other since no optimization problem is involved.

The second algorithm `LDFM` focuses instead on computing the local optimal bound of the matrix measure under linear coordinate transformation,

which leads to a more accurate discrepancy function. The idea is to search all possible linear coordinate transformation such that the matrix measures under the transformed coordinates are minimized. It involves solving several optimization problems using semidefinite programming and therefore is more computationally expensive than the previous method. We will also provide two techniques for computing the optimal bound on the matrix measure of the interval matrix. The first method uses the vertex matrices of the interval matrix, and the second uses interval matrix norms. The vertex matrices approach provides more accurate results but is more expensive, while the interval matrix norm approach is faster but less accurate. Both approaches are less conservative than the former fast algorithm as they find locally optimal exponential change rates.

In summary, the contributions of this thesis are as follows:

(a) We provide two algorithms, namely `LDF2` and `LDFM`, for over-approximating reachtubes for nonlinear models. Although in this thesis we concentrate on the reachability analysis of nonlinear dynamical systems, with appropriate handling of guards and transitions as shown in [5], these algorithms can be used in hybrid verification tools like C2E2 [5] and Breach [2].

(b) We show that the proposed algorithms for computing discrepancy function preserve the soundness and the relative completeness of the overall verification algorithm.

(c) We establish that algorithm `LDFM` returns locally optimal exponential rate for estimating the distance between neighboring trajectories. Moreover, for contractive models, the error in over-approximation using the algorithm `LDFM` converges to 0—a desirable property that existing simulation-driven verification algorithms do not have.

(d) We compare prototype implementations of the algorithms with Flow* [1] on a suite of linear and nonlinear system examples; the results suggest that this method provides significant advantages for large and complex systems.

1.3 Related Work

Several recent approaches have been proposed to obtain proofs about (bounded time) invariant or safety properties from simulations [2, 17, 18, 22]. One such approach is sensitivity analysis, which is a technique to systematically simulate arbitrary nonlinear systems with inputs [2]. The technique relies on computing the *sensitivity matrix*, a matrix that captures the sensitivity of the system to its initial condition x_0. This is then used to give an upper bound on the distance between two system trajectories. For general nonlinear models, this approach may not be sound, as higher order error terms are ignored when computing this upper bound. In [23], the authors provided sound simulation-driven methods to overapproximate the distance between trajectories, but these methods are mainly limited to affine and polynomial systems.

The idea of computing the reachable sets from trajectories is motivated by notions of incremental stability [19], which provides techniques to measure the distance between trajectories. In this work, we do not require systems to be incrementally stable, and we allow bounded parameter variation in the dynamics. Similar ideas have been considered based on abstraction techniques to synthesize controllers [24].

Other approaches for reachable set estimation for nonlinear systems operate directly on the vector field, leading to computations involving higher-order Taylor expansions [1]. In contrast, our technique performs operations on the symmetric part of the Jacobian matrix of the vector field. By obtaining bounds on this matrix, we conservatively and accurately characterize the reachable set of states over bounded time. The matrix bounds can be obtained for a broad class of nonlinear systems.

The work closest to ours is the reachability analysis using matrix measures [21], where the authors prove that the matrix measure of the Jacobian matrix can bound the distance between neighboring trajectories of the system. The technique in [21] requires the support of user-provided matrix measure functions. However, it is generally difficult to find the closed-form matrix measure. For instance, for the 2-norm case, the matrix value function is equivalent to the closed-form eigenvalue function of the Jacobian matrix. In contrast, our approach automatically computes the bounds on 2-norm or locally optimal matrix measures.

Techniques such as ours that perform reachable set over-approximations for continuous (non-hybrid) systems are crucial components of many frameworks for hybrid systems verification. For example, our approach can be used to provide the annotations for hybrid models used by the C2E2 tool to perform verification, as in [18]. Future work will evaluate the performance of our technique for hybrid examples, but in this work, we focus on reachable set over-approximations for continuous nonlinear dynamical systems.

1.4 Organization

The rest of the thesis is organized as follows. In Chapter 2, we provide notations and background information that will be used throughout the thesis. Then we give an overview of the simulation-driven verification approach in Chapter 3. Chapter 4 contains the main results: different algorithms to compute the discrepancy function and how to use them to do reach set computation. The algorithms provided in Chapter 4 can be used directly as the core function in the verification algorithm in Chapter 4. In Chapter 5, we use benchmark models to illustrate the accuracy and the performance of the proposed algorithms, followed by the conclusion in Chapter 6.

Chapter 2

PRELIMINARIES

In this chapter, we will introduce definitions, background information, and basic results which will be used throughout the thesis.

2.1 Sets, Functions, Vectors and Matrix Norms

2.1.1 Functions

The set of all real numbers is denoted by $\mathbb{R}$. The n-dimensional *Euclidean space* is defined by the set of all n-dimensional vectors $x = [x_1, x_2, \cdots, x_n]^T$, where $x_1, x_2, \cdots, x_n \in \mathbb{R}$, and is denoted by $\mathbb{R}^n$.

A function f mapping a set S_1 to a set S_2 is denoted by $f : S_1 \mapsto S_2$. A function f is *uniformly continuous* if $\forall \epsilon > 0, \exists \delta > 0$ such that $\forall \|x - y\| < \delta \Rightarrow \|f(x) - f(y)\| < \epsilon$. The δ here is independent of x, but only depends on ϵ. For example, the function $f(x) = x^2, x \in \mathbb{R}$ is continuous but not uniformly continuous.

A continuous function $f : \mathbb{R}^n \to \mathbb{R}$ is *smooth* if all its higher derivatives and partial derivatives exist and are also continuous. For example, any polynomial function is smooth. A function is called Lipschitz continuous if it has a Lipschitz constant $L \geq 0$ for which every $x, y \in \mathbb{R}^n$, $||f(x)-f(y)|| \leq L||x-y||$.

A function $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ is a *class $\mathcal{K}$ function* if it is continuous, strictly increasing, and $f(0) = 0$. For example, $f(x) = x, x \geq 0$ is a class $\mathcal{K}$ function.

2.1.2 Vector norms

The norm $\|x\|$ of a vector x is a real valued function with the properties [25]:

1. $\|x\| \geq 0, \forall x \in \mathbb{R}^n$, $\|x\| = 0$ if and only if $x = 0$.

2. $\|x+y\| \leq \|x\| + \|y\|, \forall x, y \in \mathbb{R}^n$.

3. $\|cx\| = |c|\|x\|, \forall c \in \mathbb{R}$ and $x \in \mathbb{R}^n$, where $|\cdot|$ means the absolute value.

The p-norm of a vector is defined as

$$\|x\|_p = \left(|x_1|^p + \cdots |x_n|^p\right)^{\frac{1}{p}}, 1 \leq p < \infty$$

and $\|x\|_\infty = \max_i |x_i|$.

Hereafter, if not specifically stated otherwise, $\|x\|$ refers to the 2-norm, also known as the *Euclidean norm*

$$\|x\|_2 = \left(|x_1|^2 + \cdots |x_n|^2\right)^{\frac{1}{2}} = \left(x^T x\right)^{\frac{1}{2}}.$$

The most frequently used norms are 1, 2 and ∞ norms. They are equivalent in the sense that the following relationship between the different norms is well-known [26]:

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2, \|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty, \|x\|_\infty \leq \|x\|_1 \leq n\|x\|_\infty.$$

For symmetric matrices A and B, the inequality $A \preceq B$ ($A \succeq B$) means that $B - A$ is positive (negative) semi-definite and $A \prec B$ ($A \succ B$) means $B - A$ is positive (negative) definite.

If we perform linear coordinate transformation for the vectors x, the norm of the vectors under the new coordinates would be different. This motivates the *linear transformed* norm of a vector x: $\|x\|_M \triangleq \sqrt{x^T M x}$, where M is a positive definite matrix $M \in \mathbb{R}^{n \times n}$. We call it the M-norm of the vector x. For any $M \succ 0$, there exists a nonsingular matrix $C \in \mathbb{R}^{n \times n}$, such that $M = C^T C$. So $\|x\|_M \equiv \|Cx\|$. That is, $\|x\|_M$ is the 2-norm of the linearly transformed vector Cx. When $M = I$ is the identity matrix, $\|x\|_I$ coincidences with the 2-norm.

2.1.3 Sets

A set S is said to be *bounded* is there exists a constant $c > 0$ such that for all $x \in S$, we have $\|x\| \leq r$. A subset $S \subset \mathbb{R}$ is said to be *open* if for $\forall x \in S$, we can find an arbitrary small neighborhood of x, $N_\epsilon(x) = \{y \in \mathbb{R}^n \mid \|y - x\| < \epsilon\}$

such that $N_\epsilon \subset S$. A set S is said to be *closed* if its complement in $\mathbb{R}^n$ is open. A set S is said to be *compact* if it is closed and bounded.

For $\delta \geq 0$, a δ ball around x is defined as $B_\delta(x) = \{x' \in \mathbb{R}^n \mid ||x' - x|| \leq \delta\}$. We call δ the radius of the ball. For a set $S \subseteq \mathbb{R}^n$, the set of S bloated by δ is defined as $B_\delta(S) = \cup_{x \in S} B_\delta(x)$. The *Minkowski sum* of two sets of position vectors S_1 and S_2 in Euclidean space is defined to be the addition of each vector in S_1 and each vector in S_2; i.e., the Minkowski sum of S_1 and S_2 is defined as $\{x + y \mid x \in S_1, y \in S_2\}$. Let $S \oplus B_\delta(0)$ represents the Minkowski sum of S and $B_\delta(0)$. Therefore, $S \oplus B_\delta(0) = B_\delta(S)$. For sets $S_1, S_2 \subseteq \mathbb{R}^n$, $\texttt{hull}(S_1, S_2)$ is their convex hull. The diameter of a compact set S is defined as $dia(S) = \sup_{x,y \in S} \|x - y\|$. The notation $E_{M,c}(x_c) = \{x \mid \|x - x_c\|_M^2 \leq c\}$ represents an ellipsoid centered at x_c, with *shape* M and *size* c.

2.1.4 Matrix norms

We denote the transpose of a matrix A by A^T. An $n \times m$ matrix A of real elements defines a linear mapping $y = Ax$ from $\mathbb{R}^m$ to $\mathbb{R}^n$. The induced p-norm of the matrix A is defined as

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p = 1} \|Ax\|_p.$$

For $p = 1, 2, \infty$, the p-norm is given by

$$\begin{aligned} \|A\|_1 &= \max_j \textstyle\sum_{i=1}^m |a_{ij}|, \\ \|A\|_2 &= \sqrt{\lambda_{\max}(A^T A)}, \\ \|A\|_\infty &= \max_i \textstyle\sum_{j=1}^n |a_{ij}|, \end{aligned}$$

where $\lambda_{\max}(A^T A)$ is the maximum eigenvalue of $A^T A$. The lower case letters with subscripts denote the corresponding element of a matrix, e.g., a_{ij} denotes the element in the i^{th} row and j^{th} column of A. In the rest of the thesis, if not specifically claimed otherwise, $\|A\|$ also refers to the 2-norm of A. The matrix norm also obeys some inequalities. If matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times l}$ are real valued matrices, then

$$\begin{aligned} &\frac{1}{\sqrt{n}} \|A\|_\infty \leq \|A\|_2 \leq \sqrt{m} \|A\|_\infty, \quad \frac{1}{\sqrt{m}} \|A\|_1 \leq \|A\|_2 \leq \sqrt{n} \|A\|_1, \\ &\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}, \qquad\qquad \|AB\|_p \leq \|A\|_p \|B\|_p. \end{aligned}$$

The Frobenius norm of an $m \times n$ matrix A is a matrix norm which is defined as the square root of the sum of the absolute squares of the elements:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij})^2}.$$

We also have

$$\|A\|_2 \leq \|A\|_F.$$

2.2 Dynamical Systems

The continuous evolution of a cyber-physical system can be mathematically modeled as a dynamical system. Consider an n-dimensional *dynamical system*:

$$\dot{x} = f(x), \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz continuous function describing the continuous evolution of the physical variables of the cyber-physical system.

A *solution or a trajectory* of the system is defined as a function $\xi : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, such that for any initial state $x_0 \in \mathbb{R}^n$ and at any time $t \in \mathbb{R}_{\geq 0}$, $\xi(x_0, t)$ satisfies the differential equation (2.1). The *initial set* of the system (2.1) is defined as the set of initial states, and we will denote it as Θ throughout the thesis.

The existence and uniqueness of the solution can be guaranteed by the Lipschitz continuity of f.

Assume that the function f is also continuously differentiable. The *Jacobian* of f, $J_f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$, is a matrix-valued function of all the first-order partial derivatives of f with respect to x: $[J_f(x)]_{ij} = \frac{\partial f_i(x)}{\partial x_j}$. The following lemma states a relationship between f and its Jacobian J_f which can be proved using the generalized mean value theorem [17].

Lemma 2.1. *For any continuously differentiable vector-valued function* $f : \mathbb{R}^n \to \mathbb{R}^n$, *and* $x, r \in \mathbb{R}^n$,

$$f(x+r) - f(x) = \left(\int_0^1 J_f(x+sr)ds\right) \cdot r, \tag{2.2}$$

where the integral is component-wise.

Proof. In this proof, the i's in subscript correspond the i^{th} components of the vector functions. For any $t \in [0,1], i \in \{1,\ldots,n\}$, we define $g_i(t) := f_i(x+tr)$. Then we have

$$f_i(x+r) - f_i(x) \;=\; g_i(1) - g_i(0) = \int_0^1 \frac{dg_i(t)}{dt} dt. \tag{2.3}$$

Using the chain rule of gradient, we have

$$\begin{aligned} \frac{dg_i(t)}{dt} &= \nabla f_i(u)|_{u=x+tr} \cdot \frac{d(x+tr)}{dt} \\ &= \nabla f_i(u)|_{u=x+tr} \cdot r = \sum_{j=1}^{n} \left.\frac{\partial f_i(u)}{\partial u_j}\right|_{u=x+tr} r_j, \end{aligned} \tag{2.4}$$

where $\nabla f_i(u) = [\frac{\partial f_i(u)}{\partial u_1}, \frac{\partial f_i(u)}{\partial u_2}, \ldots, \frac{\partial f_i(u)}{\partial u_n}]$ is the gradient of function f_i. Substituting (2.4) in (2.3), we have:

$$\begin{aligned} f_i(x+r) - f_i(x) &= \int_0^1 \left(\sum_{j=1}^{n} \left.\frac{\partial f_i(u)}{\partial u_j}\right|_{u=x+sr} r_j \right) ds \\ &= \sum_{j=1}^{n} \left(\int_0^1 \left.\frac{\partial f_i(u)}{\partial u_j}\right|_{u=x+sr} ds \right) r_j. \end{aligned}$$

Since $J_f(x+sr)$ is the matrix consisting of the components of $\left.\frac{\partial f_i(u)}{\partial u_j}\right|_{u=x+sr}$, the lemma holds. □

2.3 Simulation and Reach Set

Although it is generally difficult to get the closed-form solution of dynamic systems, validated simulation libraries, such as VNODE-LP [27] and CAPD [28], use numerical integration to generate a sequence of states with guaranteed error bounds. First we give the following definition of *simulation* as a sequence of time-stamped hyper-rectangles.

Definition 2.1. *(Simulation) For any $x_0 \in \mathbb{R}, \tau > 0, \epsilon > 0, T > 0$, a* (x_0, τ, ϵ, T)-simulation *of the system described in Eq. (2.1) is a sequence of time-stamped sets $\{(R_i, t_i)_{i=0}^n\}$ satisfying the following:*

1. *R_i is a compact set in $\mathbb{R}^n$ with a diameter smaller than ϵ, for $i = 0, 1, \ldots, n$.*

2. *Let $\xi(x_0, t)$ be the system trajectory starting from x_0 along time. Then $\xi(x_0, t_i) \in R_i, i = 0, 1, \ldots, n$, and $\forall t \in (t_{i-1}, t_i)$, $\xi(x_0, t) \in \texttt{hull}(R_{i-1}, R_i)$, for $i = 1, \ldots, n$.*

3. *τ is called the* maximum sampling period*, which means that for each $i = 1, \ldots, n$, $0 < t_i - t_{i-1} \leq \tau$. Note $t_0 = 0$ and $t_n = T$.*

We call a state x *reachable* from the initial set Θ if there exist a state $\theta \in \Theta$ and a time $t \geq 0$ such that $\xi(\theta, t) = x$. We call the set of states that are reachable from the initial set Θ during time $[t_1, t_2]$ the *reach set*, and denote it as $\texttt{Reach}(\Theta, [t_1, t_2])$. Similarly, we denote the set of reachable states at time t from initial set Θ as $\texttt{Reach}(\Theta, t)$.

Next, we introduce the definition of *reachtube*, which is also a sequence of time-stamped hyper-rectangles, but instead contains the all the trajectories starting from the initial set Θ.

Definition 2.2. *(Reachtube) For any $\Theta \subseteq \mathbb{R}^n, T > 0$, a (Θ, T)-reachtube is a sequence of time-stamped compact sets $\{(O_i, t_i)_{i=0}^n\}$, such that each $\texttt{Reach}(\Theta, [t_{i-1}, t_i]) \subseteq O_i$, where $\Theta \subseteq \mathbb{R}^n$ is a set of states, and T is the time bound.*

Given an n-dimensional dynamical system as in equation (2.1), a compact initial set $\Theta \in \mathbb{R}^n$, an unsafe set $\texttt{unsafe} \subseteq \mathbb{R}^n$, and a time bound $T > 0$, the safety verification problem is to check whether $\texttt{Reach}(\Theta, [0, T]) \cap \texttt{unsafe} = \emptyset$.

It has been proven that the safety verification of rectangular hybrid automata is undecidable [29]. The reachability problem for general nonlinear dynamical systems is also undecidable [30, 31]. Recent focus has been on methods to over-approximate the reach set of the system over bounded time. Existing methods like Taylor models [32] use interval arithmetic [33] to bound the integration value. However, this method suffers from complexity that increases exponentially with both the dimension of the system and order of the model. The major contribution of this thesis is the introduction of the methods to compute reachtubes that are less conservative and less time consuming.

2.4 Interval Matrices

The interval matrices will be used in the thesis to linearly over-approximate behaviors of nonlinear models. Interval matrices are matrices where each element is an interval instead of a constant.

We call $[B, C]$ an *matrix pair* if $B, C \in \mathbb{R}^{n\times n}$ and $b_{ij} \leq c_{ij}$ for all $1 \leq i, j \leq n$. For a matrix pair $[B, C]$, we define the *matrix interval*,

$$\texttt{Interval}([B, C]) \triangleq \{A \in \mathbb{R}^{n\times n} | b_{ij} \leq a_{ij} \leq c_{ij}, 1 \leq i, j \leq n\}.$$

We call $\mathcal{A} = \texttt{Interval}([B, C])$ an interval matrix. Two useful notions are the *center matrix* and the *range matrix*, which are defined as $\texttt{CT}([B, C]) = (B+C)/2$, and $\texttt{RG}([B, C]) = (C-B)/2$, respectively. Then $\texttt{Interval}([B, C])$ can also be written as $\texttt{Interval}([A_c - A_r, A_c + A_r])$, where $A_c = \texttt{CT}([B, C])$, $A_r = \texttt{RG}([B, C])$.

Next we introduce the notion of *interval matrix norm*. We start our discussion with an arbitrary norm $\|\cdot\|$ of matrices; it can be $1, 2, \infty$, or the Frobenius norm. Later we will pick specific norms for each case. Given a norm for matrices $\|\cdot\|$, the corresponding norm on an interval matrix is defined as:

$$|||\mathcal{A}||| = \sup_{A\in\mathcal{A}} \|A\| \tag{2.5}$$

and $|||\mathcal{A}|||$ is called the interval matrix norm of $\mathcal{A}$. The following theorem from [34] provides a method to calculate the norm of an interval matrix from the norms of its center and range.

Theorem 2.1 (Theorem 10 from [34])**.** *For any interval matrix $\mathcal{A}$,*

$$\begin{aligned} |||\mathcal{A}|||_1 &= \| \, |\texttt{CT}(\mathcal{A})| + \texttt{RG}(\mathcal{A})\|_1, \\ |||\mathcal{A}|||_\infty &= \| \, |\texttt{CT}(\mathcal{A})| + \texttt{RG}(\mathcal{A})\|_\infty, \\ |||\mathcal{A}|||_F &= \| \, |\texttt{CT}(\mathcal{A})| + \texttt{RG}(\mathcal{A})\|_F, \end{aligned}$$

where $|A|$ is the matrix obtained by taking the element-wise absolute value of matrix A.

For an interval matrix $\texttt{Interval}([B, C])$, we define

$$\mathcal{V} = \texttt{VT}(\texttt{Interval}([B, C])) = \{V \in \mathbb{R}^{n\times n} | v_{ij} = b_{ij}, \text{or, } v_{ij} = c_{ij}, 1 \leq i, j, \leq n\}.$$

The elements of $\mathcal{V}$ are called *vertex matrices*, the entries of which are the boundary values of B or C. The cardinality of $\mathcal{V}$ is 2^{n^2}.

Let $\mathcal{A} = \texttt{Interval}([B, C])$, and $\mathcal{V} = \texttt{VT}(\mathcal{A})$. We use $\texttt{hull}(\mathcal{V})$ to denote the convex hull of $\mathcal{V}$. Assume A_i, $i = 1, 2, \ldots, N$ are all the elements of $\mathcal{V}$, where N is the cardinality of $\mathcal{V}$. Then

$$\begin{aligned}\texttt{hull}(\{A_1, \ldots, A_N\}) \triangleq \{A \in \mathbb{R}^{n\times n} | \exists \alpha_1, \ldots, \alpha_N \geq 0, \text{ and} \\ \sum_{i=1}^{N} \alpha_i = 1, \text{ s.t. } A = \sum_{i=1}^{N} \alpha_i A_i\}.\end{aligned}$$

It can be shown that the convex hull of the vertex matrices for an interval matrix is the interval matrix itself.

Proposition 2.1. *For any interval matrix* $\mathcal{A}$, $\texttt{hull}(\texttt{VT}(\mathcal{A})) = \mathcal{A}$.

Proposition 2.1 can be proved by constructing a bijection that maps an n-dimensional interval matrix to an n^2-dimensional hyper-rectangle. Vectorizing, or *flattening*, the vertex matrices in $\mathcal{A}$, we obtain the vertices of this hyper-rectangle. Then Proposition 2.1 holds, since the convex hull of the vertices of a rectangle is the rectangle itself.

2.5 Discrepancy Function

A discrepancy function bounds the distance between two neighboring trajectories, based on the initial distance between states and the time [17, 18]. That is, given any two trajectories $\xi(x_1, t)$ and $\xi(x_2, t)$ of the system (2.1) starting from states x_1 and x_2 respectively, the discrepancy function β is a function of initial distance between x_1 and x_2, and time t. As shown in Figure 2.1, at any time t, the distance between $\xi(x_1, t)$ and $\xi(x_2, t)$ should be no greater than the value of discrepancy function at t. The distance between any two states can be measured under Euclidean norm or M-norm defined in Section 2.1. We give the formal definition of the discrepancy function as follows:

Definition 2.3. *Given a positive definite matrix* M*, a continuous function* $\beta : \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ *is a discrepancy function of the system in Equation (2.1) if*

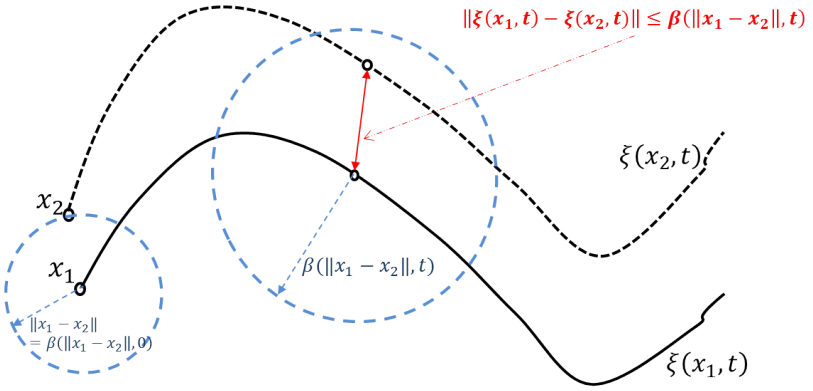

Figure 2.1: An illustration of discrepancy function.

(1) for any pair of states $x_1, x_2 \in \mathbb{R}^n$*, and any time* $t \geq 0$*,*

$$\|\xi(x_1,t)-\xi(x_2,t)\|_M \leq \beta(\|x_1-x_2\|_M, t), \text{ and}$$

(2) for any t*,* $\lim_{\|x_1-x_2\|_M \to 0^+} \beta(\|x_1-x_2\|_M, t) = 0$*.*

According to the definition of discrepancy function, for system (2.1), at any time t, the ball centered at $\xi(x_0,t)$ with radius $\beta(\delta,t)$ contains the reach set of (2.1) starting from $B_\delta(x_0)$. Therefore, by bloating the simulation trajectories using the corresponding discrepancy function, we can obtain the over-approximating reachtube. Similar ideas have been considered based on abstraction techniques to synthesize controllers [24]. Definition 2.3 corresponds to the definition of discrepancy function (Definition 2) in [18], except that we allow an arbitrary M-norm as a metric. Note that here we do not require that trajectories converge to each other. As noted in [18, 21], several techniques (contraction metric [20], incremental stability [19], matrix measure [21], etc.) can be used to find discrepancy functions; however, those techniques either restrict the class of nonlinear systems (e.g., polynomial systems, as in [23]) or require crucial user-supplied inputs (e.g., the closed-form expression of matrix measure function, as in [21]).

2.6 Matrix Measure

The *measure*, also known as the *logarithmic norm*, $\mu(A)$ of a matrix $A \in \mathbb{R}^{n\times n}$ can be seen as the one-sided derivative of $\|\cdot\|$ at $I \in \mathbb{R}^{n\times n}$ in the direction

Table 2.1: Commonly used vector norms and their corresponding matrix norms and measures adapted from [21]

Vector norm	Induced matrix norm	Induced matrix measure
$\lvert x\rvert_1 = \sum_j \lvert x_j\rvert$	$\lVert A\rVert_1 = \max_j \sum_i \lvert a_{ij}\rvert$	$\mu_1(A) = \max_j \left(a_{jj} + \sum_{i\neq j} \lvert a_{ij}\rvert\right)$
$\lvert x\rvert_2 = \sqrt{\sum_j x_j^2}$	$\lVert A\rVert_2 = \sqrt{\max_j \lambda_j(A^T A)}$	$\mu_2(A) = \max_j \frac{1}{2}\left(\lambda_j(A + A^T)\right)$
$\lvert x\rvert_\infty = \max_j \lvert x_j\rvert$	$\lVert A\rVert_\infty = \max_i \sum_j \lvert a_{ij}\rvert$	$\mu_\infty(A) = \max_i \left(a_{ii} + \sum_{j\neq i} \lvert a_{ij}\rvert\right)$

of A:

$$\mu(A) = \lim_{t\to 0^+} \frac{\lVert I + tA\rVert - \lVert I\rVert}{t}. \tag{2.6}$$

Some commonly seen norms and their corresponding matrix measures can be found in Table 2.1 [21].

Matrix measure is well defined and has been used to measure the distance between the trajectories. Most results in the rest of this section are from [35] and [36].

Lemma 2.2. *For any $A \in \mathbb{C}^{n\times n}$, $\mu(A)$ is well defined.*

Lemma 2.3 (Basic properties of matrix measure)**.** *Let $A \in \mathbb{C}^{n\times n}$; then*

1. $-\lVert A\rVert \leq -\mu(-A) \leq \mu(A) \leq \lVert A\rVert$.
2. $\mu(cA) = c\mu(A), \forall c \geq 0$.
3. $-\mu(-A) \leq Re\,(\lambda_i(A)) \leq \mu(A), \forall i \in \{1, 2, \ldots, n\}$.
4. *If $P \in \mathbb{R}^{n\times n}$ is nonsingular, then the measure μ_P of the norm $\lvert x\rvert_P = \lvert Px\rvert$ is given in terms of μ by $\mu_P(A) = \mu(PAP^{-1})$.*

The matrix measure has long been used to provide estimates on solutions of systems of ordinary differential equations. The next proposition is the key that provides a bound on the distance between trajectories in terms of their initial distance and the rate of expansion of the system given by the measure of the Jacobian matrix $J(x)$ with respect to x.

Proposition 2.2. *Let $\mathcal{D} \subseteq \mathbb{R}^n$ and let the Jacobian $J(x) = \frac{\partial f}{\partial x}(x)$ satisfy $\mu(J(x)) \leq c$ for all $x \in \mathcal{D}$. If every trajectory of equation (2.1) with initial conditions in the line segment $\{hx_0 + (1-h)z_0 : \mid \quad h \in [0,1]\}$ remains in $\mathcal{D}$ until time T, then the solutions $\xi(x_0, t)$ and $\xi(z_0, t)$ satisfy*

$$\lvert\xi(x_0, t) - \xi(z_0, t)\rvert \leq \lvert x_0 - z_0\rvert e^{ct} \tag{2.7}$$

for all $t \in [0, T]$.

Proposition 2.2 provides global divergence between trajectories of equation (2.1) using only information about the system's Jacobian at each point. It provides a way to compute discrepancy function. If there exists $c < 0$ such that for all $(t, x) \in [0, \infty) \times \mathcal{D}$ we have $\mu(J(x)) \leq c$, then system (2.1) or the vector field $f(x)$ is said to be contracting with respect to $|\cdot|$. But here we do not assume the sign of c.

In this chapter, we established notations and definitions about functions, matrices and dynamical systems. We have also discussed some useful results that serve as the background for the remaining chapters.

Chapter 3

SIMULATION-DRIVEN VERIFICATION

3.1 Safety Verification and Reach Set Over-approximation

The *safety verification problem* is to decide whether any reachable state of the system violates some safety requirement within bounded time. We formalize the problem of safety verification for dynamical systems as follows:

Given an n-dimensional dynamical system as in equation (2.1), a compact initial set $\Theta \in \mathbb{R}^n$, an unsafe set $\texttt{unsafe} \subseteq \mathbb{R}^n$, and a time bound $T > 0$, we would like to check if $\texttt{Reach}(\Theta, [0, T]) \cap \texttt{unsafe} = \emptyset$. If there exists some $\epsilon > 0$ such that $B_\epsilon(\texttt{Reach}(\Theta, [0, T])) \cap \texttt{unsafe} = \emptyset$, we say the system is robustly safe. That is, the system is robustly safe if all states in some envelope around the system behaviors are safe. If there exists some $\epsilon > 0, x \in \Theta$, such that $B_\epsilon(\xi(x, t)) \subseteq \texttt{unsafe}$ over some interval $[t_1, t_2], 0 \leq t_1 < t_2 \leq T$, we say the system is robustly unsafe.

A verification algorithm for checking the safety of the system is said to be *sound* if the answers of safety/unsafety of the system given by the algorithm are correct. The algorithm is said to be *relatively complete* if the algorithm is guaranteed to terminate when the system is either robustly safe or unsafe

As we have discussed in Section 2.3, computing the reach set exactly is undecidable even for the simplest classes of systems. We can instead use the reachtube, which conservatively estimates all the behaviors of the system. A sequence of papers [2, 17, 18, 37] presented algorithms for solving this problem for a broad class of nonlinear dynamical, switched, and hybrid systems.

The simulation-driven bounded-time safety verification approach consists of the following three steps [17]:

1. Simulate the system from a finite set of states x_i that are chosen from

the compact (close and bounded) initial set Θ. The union of the δ balls centered at x_i should contain the initial set: $\Theta \subseteq \cup_i B_\delta(x_i)$.

2. Bloat the (x_i, τ, ϵ, T)-simulations with discrepancy function such that the bloated sets are over-approximations of the reachable states from initial covers $B_\delta(x_i)$.

3. Check each of these over-approximations, and decide if the system is safe or not. If such a decision cannot be made, then we should start from the beginning with finer sampling grids over the initial set.

This approach if useful not only in these bounded-time safety verification problems, but also in verifying a broader class of system properties (for example, temporal precedence [37]).

The most crucial and difficult step in the above procedure is the second step: how to choose a discrepancy function. On the one hand, the function should be large enough to give a strict over-approximation of the reachable set; on the other hand, it should be small enough so that the over-approximation is not too pessimistic. Moreover, the value of the function should converge to 0 as the initial set converges to a single point. In the rest of the thesis, we first review the simulation-driven verification algorithm in [17], then discuss algorithms which compute discrepancy function and use it to compute reach set over-approximations.

3.2 Simulation-driven Verification Algorithm

The simulation-driven verification algorithm implements the simulate-bloat-check-refine steps discussed in Section 3.1. It has five inputs: 1. initial set $\Theta \subset \mathbb{R}^n$, 2. time bound $T > 0$, 3. unsafe set $\texttt{unsafe} \subset \mathbb{R}^n$, 4. initial simulation precision ϵ_0, 5. initial simulation sampling period τ_0, and two outputs SAFE and UNSAFE. It will terminate and return the correct result if the system is robustly safe or robustly unsafe.

Algorithm 1 shows the structure of the simulation-driven verification algorithm. It returns SAFE if the reach set $\texttt{Reach}(\Theta, [0, T])$ has no intersection with the unsafe set, along with a robustly safe reachtube STB, or returns UNSAFE upon finding a counter-example, the simulation ψ which has some

Algorithm 1: Simulation-driven Verification of Dynamic Systems

input: $\Theta, T, \texttt{unsafe}, \epsilon_0, \tau_0$

$\delta \leftarrow dia(\Theta); \epsilon \leftarrow \epsilon_0; \tau \leftarrow \tau_0; \mathsf{STB} \leftarrow \emptyset;;$
$\mathcal{C} \leftarrow Cover(\Theta, \delta, \epsilon);$
while $\mathcal{C} \neq \emptyset$ **do**
  for $\langle \theta, \delta, \epsilon \rangle \in \mathcal{C}$ **do**
    $\psi = \{(R_k, t_k)_{i=0}^n\} \leftarrow Simulate(\theta, T, \epsilon, \tau);$
    $\mathcal{R} \leftarrow Bloat(\psi, \delta, \epsilon);$
    if $\mathcal{R} \cap \texttt{unsafe} = \emptyset$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \backslash \{\langle \theta, \delta, \epsilon \rangle\}$;
      $\mathsf{STB} \leftarrow \mathsf{STB} \cup \mathcal{R}$;
    else if $\exists k, R_k \subseteq \texttt{unsafe}$ **then**
      return (UNSAFE, ψ)
    else
      $\mathcal{C} \leftarrow \mathcal{C} \cup Cover(B_\delta(\theta), \frac{\delta}{2}, \frac{\epsilon}{2}) \backslash \{\langle \theta, \delta, \epsilon \rangle\};$
      $\tau \leftarrow \frac{\tau}{2}$;
    end
  end
end
return (SAFE, STB);

part fully contained in the unsafe region. An illustration of Algorithm 1 is shown in Figure 3.1.

There are several functions referred to in Algorithm 1. Functions $dia()$ and $Simulate()$ are defined to return the diameter of set and simulation result respectively. The $Bloat()$ function takes as the inputs the simulation ψ starting from θ, the size of the initial cover δ and the simulation precision ϵ, and returns a reachtube that contains all the trajectories starting from the initial cover $B_\delta(\theta)$. It can be done by bloating the simulation using discrepancy function as described in Section 2.5, which is an over-approximation of the distance between any neighboring trajectories starting from $B_\delta(\theta)$. The main contribution of this thesis includes algorithms for implementing this $Bloat()$ function and in Chapter 4 we will present these algorithms in detail. Function $Cover()$ returns a set of triples $\{\langle \theta, \delta, \epsilon \rangle\}$, where θ's are sample states, the union of $B_\delta(\theta)$ covers Θ completely, and ϵ is the precision of simulation.

There are two important data structures used in Algorithm 1: $\mathcal{C}$ is a collection of the triples returned by $Cover()$, which represents the subset of Θ that has not yet been proved safe, and STB stores the updated bounded time reachtube.

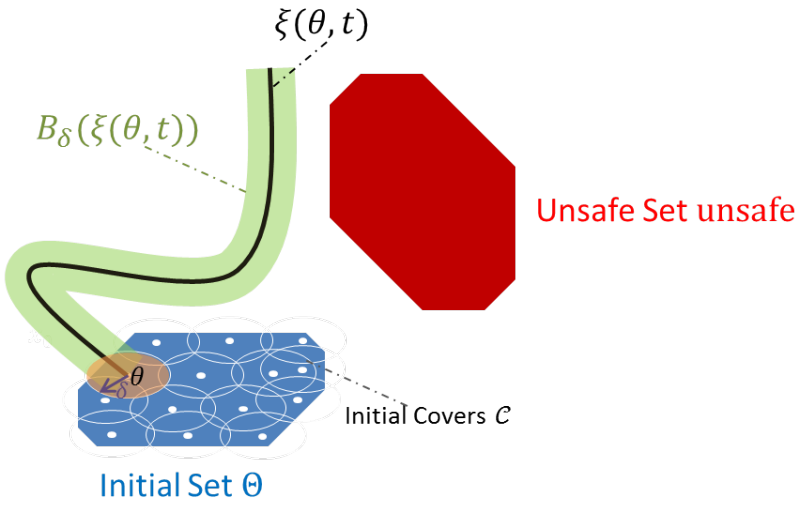

Figure 3.1: An illustration of Algorithm 1. Blue shape: initial set Θ. Red shape: unsafe set `unsafe`. White circles: initial covers. Black trace: simulation from initial state x_0. Green shape: reachtube from initial cover $B_\delta(x_0)$, where δ is the radius of the initial cover.

Initially, $\mathcal{C}$ contains a singleton $\langle \theta_0, \delta_0 = dia(\Theta), \epsilon_0 \rangle$, where $\Theta \subseteq B_{\delta_0}(\theta_0)$ and ϵ_0 is a small positive constant. For each triple $\langle \theta, \delta, \epsilon \rangle \in \mathcal{C}$, the **while**-loop from Line 3 checks the safety of the reachtube from $B_\delta(\theta)$, which is computed in Line 5-6. ψ is a $(\theta, T, \epsilon, \tau)$-simulation, which is a sequence of time-stamped rectangles $\{(R_k, t_k)\}$ and is guaranteed to contain the trajectory $\xi(\theta, T)$ by Definition 2.1. Bloating the simulation result ψ by the discrepancy function to get $\mathcal{R}$, a $(B_\delta(\theta), T)$-reachtube, we have an over-approximation of $\texttt{Reach}(B_\delta(\theta), [0, T])$. The core function $Bloat()$ will be discussed in detail in the next chapter. If $\mathcal{R}$ is disjoint from `unsafe`, then the reachtube from $B_\delta(\theta)$ is safe and the corresponding triple can be safely removed from $\mathcal{C}$. If for some k, R_k (one rectangle of the simulation) is completely contained in the unsafe set, then we can get a counterexample of a trajectory that violates the safety property. Otherwise the safety of $\texttt{Reach}(B_\delta(\theta), [0, T])$ is nondeterministic and a refinement of $B_\delta(\theta)$ needs to be made with smaller δ and smaller ϵ, τ.

Firstly, the algorithm requires that the $Bloat()$ function returns an over-approximation of all the reachable states of the system from the given initial cover $\langle \theta, \delta, \epsilon \rangle$. This guarantees that the union of all the bloated simulations STB is an over-approximation of $\texttt{Reach}(\Theta, [0, T])$, which leads to the soundness of the algorithm. Then, it is required that as δ gets finer (i.e., smaller),

the value of the discrepancy function will become smaller (i.e., the reachtube is arbitrary close to the simulation), which guarantees that the algorithm always terminates. To sum up, if the discrepancy function satisfies all the requirements as stated in Definition 2.3, it gives us two key properties of the algorithm [18]:

Theorem 3.1. *(Soundness and relative completeness) Consider a nonlinear dynamical system as described by equation (2.1) with continuously differentiable $f(\cdot)$. Let $\Theta \subseteq \mathbb{R}^n$ be a compact initial set and* `unsafe` $\subseteq \mathbb{R}^n$ *be an unsafe set.*

1. *Soundness: If Algorithm 1 outputs "SAFE", the system is safe, that is,* $\texttt{Reach}(\Theta, T) \cap \texttt{unsafe} = \emptyset$*; If it outputs "UNSAFE" instead, then there exists at least one unsafe trajectory with* $\xi(x_0, t) \in \texttt{unsafe}$ *with* $x_0 \in \Theta$ *and* $t \leq T$*.*

2. *Relative Completeness: Algorithm 1 always terminates and outputs "SAFE" if the system is robustly safe. Algorithm 1 always terminates and outputs "UNSAFE" if there exists at least a trajectory* $\xi(x_0, \cdot)$ *with* $x_0 \in \Theta$ *such that it is robustly unsafe.*

In the next chapter, we will introduce different algorithms that implement the *Bloat*() function, which compute reachtubes using discrepancy function, and prove that they satisfy the desirable requirements.

Chapter 4

LOCAL DISCREPANCY ALGORITHMS

In this chapter, we will discuss several algorithms that realize the $Bloat()$ function discussed in Section 3.2. They will use the discrepancy function discussed in Section 2.5, which measures the changing of the distance between two neighboring trajectories. For general nonlinear systems, the Jacobian matrix is a function of the state, and we use constant interval matrices to bound the variation of the Jacobian over small parts of the state space. The upper bound on the matrix measure of this interval matrix is used as an upper bound on the matrix measure of the Jacobian matrix over this part of the state space. We use this bound as the exponential change rate of the discrepancy function.

We will discover different discrepancy functions under different coordinates that can fit different uses. We will introduce algorithm `LDF2` [17] that uses discrepancy function under Euclidean norm, which is fast but coarse since it bloats a simulation uniformly in each direction. We will also introduce algorithm `LDFM` [38] that computes the optimal coordinate transformation to make the reachtube tighter but needs more time. We will introduce two different version of algorithm `LDFM` using two techniques for computing the optimal exponential change rate from the interval matrix. The first method uses the vertex matrices of the interval matrix, and the second uses interval matrix norms. The vertex matrices approach provides more accurate results but is more expensive, while the interval matrix norm approach is faster but less accurate. Both approaches are less conservative than the algorithm `LDF2` as they find locally optimal exponential change rates. A positive side effect of the current methods is that it bloats a simulation nonuniformly in different directions. The main results of this chapter are adapted from [17, 38].

4.1 Interval Matrix and Local Discrepancy Function

The main obstacle to finding a (global) discrepancy function for general nonlinear systems is that it is difficult to measure all the possible rates of convergence or divergence between trajectories over the entire state space. We will restrict the domain of the discrepancy function to some compact set $S \subset \mathbb{R}^n$ instead of $\mathbb{R}^n$ in Definition 2.3. It will be shown that such local discrepancy functions are sufficient for computing the reachtubes.

We first show that under the continuous differentiable assumption of the system described using equation (2.1), the Jacobian function of the system over a compact set S can be over-approximated by an interval matrix. Then we establish that the distance between two trajectories in S satisfies a differential equation described using the interval matrix.

Since we assume the system is continuously differentiable, the Jacobian matrix should be continuous. For compact set S, the elements of the Jacobian matrix $J_f(x)$ are bounded for all $x \in S$ because of the continuity assumptions. Assuming we can compute the upper and lower bound of each term of the Jacobian matrix $J_f(x)$ within S, we can over-approximate the integration of the Jacobian matrix on the right-hand side of (4.1) using an interval matrix. The following Lemma shows that an interval matrix that contains the possible values that $J_f(x)$ can take within S exists.

Lemma 4.1. *If the Jacobian matrix $J_f(x)$ is continuous, then there exists an interval matrix* `Interval`$([B, C])$ *over compact sets S such that*

$$\forall x \in S, J_f(x) \in \texttt{Interval}([B, C]).$$

The lemma follows from the fact that each term is a continuous function of x, and over the compact domain S, the function has a maximum and minimum value that define the matrix pair $[B, C]$. The bounds of such values can be obtained for a broad class of nonlinear systems using interval arithmetic or an optimization toolbox.

Then we use the interval matrix which over-approximates the behavior of the Jacobian matrix over the compact set to analyze the rate of convergence or divergence between trajectories:

Lemma 4.2. *For system (2.1), suppose $S \subseteq \mathbb{R}^n$ is a compact convex set, and $[t_1, t_2]$ is a time interval such that for any $x \in S$, $t \in [t_1, t_2]$, $\xi(x, t) \in S$. If*

there exists an interval matrix $\mathcal{A}$ such that $\forall x \in S, J_f(x) \in \mathcal{A}$, then for any $x_1, x_2 \in S$, and for any fixed $t \in [t_1, t_2]$, the distance $y(t) = \xi(x_2, t) - \xi(x_1, t)$ satisfies $\dot{y}(t) = A(t)y(t)$, for some $A(t) \in \mathcal{A}$.

Proof. Given a compact convex set S containing all the trajectories in the time interval $[t_1, t_2]$, using Lemma 2.1 we have the following:

$$\begin{aligned} \dot{y}(t) &= \dot{\xi}(x_2, t) - \dot{\xi}(x_1, t) = f(\xi(x_2, t)) - f(\xi(x_1, t)) \\ &= \left(\int_0^1 J_f(\xi(x_1, t) + sy(t))ds \right) y(t), \end{aligned} \tag{4.1}$$

where $y(t)$ is the distance $\xi(x_2, t) - \xi(x_1, t)$ starting from $x_1, x_2 \in S$. The interval matrix $\mathcal{A} = \texttt{Interval}([B, C])$ satisfies the conditions in Lemma 4.1.

For any fixed t, $\int_0^1 J_f(\xi(x_1, t) + sy(t))ds$ is a constant matrix. Because $\xi(x_1, t), \xi(x_2, t)$ are contained in the convex set S, $\forall s \in [0, 1], \xi(x_1, t) + sy(t)$ should also be contained in S. Then at t, $J_f(\xi(x_1, t) + sy(t)) \in \texttt{Interval}([B, C])$. Since the integration is from 0 to 1, it is straightforward to check that

$$\int_0^1 J_f(\xi(x_1, t) + sy(t))ds \in \mathcal{A}.$$

We rewrite (4.1) as

$$\dot{y}(t) = A(t)y(t), A(t) \in \mathcal{A} \tag{4.2}$$

which means at any fixed time $t \in [t_1, t_2]$, we always have $\dot{y}(t) = A(t)y(t)$, where $A(t)$ is unknown but $A(t) \in \mathcal{A}$. □

How to use the differential equation in Lemma 4.2 to get a discrepancy function? Given any matrix $M \succ 0$, $\|y(t)\|_M^2 = y^T(t)My(t)$, and by differentiating $\|y(t)\|_M^2$, we have that for any fixed $t \in [t_1, t_2]$,

$$\begin{aligned} \frac{d\|y(t)\|_M^2}{dt} &= \dot{y}^T(t)y(t) + y^T(t)\dot{y}(t) \\ &= y^T(t)(A(t)^T M + MA(t))y(t), \end{aligned} \tag{4.3}$$

for some $A(t) \in \mathcal{A}$.

We write $A(t)$ as A in the following for simplicity. If there exists a $\hat{\gamma}$ such that

$$A^T M + MA \preceq \hat{\gamma} M, \forall A \in \mathcal{A},$$

then (4.3) becomes

$$\frac{d\|y(t)\|_M^2}{dt} \leq \hat{\gamma}\|y(t)\|_M^2.$$

After applying Grönwall's inequality, we have

$$\|y(t)\|_M \leq \|y(t_1)\|_M e^{\frac{\hat{\gamma}}{2}(t-t_1)}, \forall t \in [t_1, t_2].$$

$\frac{\hat{\gamma}}{2}$ can also be seen as an upper bound of the matrix measure of the family of matrices $\mathcal{A}$ (see Section 2.6). Since $\mu_M(A) \leq \frac{\hat{\gamma}}{2}, \forall A \in \mathcal{A}$ means $CAC^{-1} + (CAC^{-1})^T \preceq \hat{\gamma} I, \forall A \in \mathcal{A}$, where $M = C^T C$. Pre- and post-multiplying the inequality by C^T and C, we can also get $A^T M + MA \preceq \hat{\gamma} M, \forall A \in \mathcal{A}$.

The above provides a discrepancy function:

$$\beta(\|x_1 - x_2\|_M, t) = \|x_1 - x_2\|_M e^{\frac{\hat{\gamma}}{2}(t-t_1)}.$$

This discrepancy function could result in less conservative reachtubes, depending on the selection of M and $\hat{\gamma}$. Ideally, we would like to identify the optimal M such that we can obtain tightest bound $\hat{\gamma}$. This problem is formulated as follows:

$$\begin{aligned} \min_{\hat{\gamma}\in\mathbb{R}, M\in\mathbb{R}^{n\times n}} \quad & \hat{\gamma} \\ \text{s.t} \quad & A^T M + MA \preceq \hat{\gamma} M, \quad \forall A \in \mathcal{A} \\ & M \succ 0. \end{aligned} \tag{4.4}$$

To compute the reach set of a nonlinear model from a set of initial states over a long time horizon $[0, T]$, in principle, we could compute a single discrepancy function that holds over the entire duration. For unstable systems, this would result in large interval matrices, leading to large over-approximations. To mitigate this problem, we divide the time interval $[0, T]$ into smaller intervals $[0, t_1], [t_1, t_2]$, etc., and compute a piece-wise discrepancy function, where each piece is relevant for a smaller portion of the state space and the time. Assuming we can find an exponential discrepancy function $\beta_i(\|x_1 - x_2\|_{M_i}, t) = \|x_1 - x_2\|_{M_i} e^{\gamma_i(t-t_1)}$ for each time interval $[t_{i-1}, t_i], i = 1, \ldots, k$ and $t_k = T$, we can compute the reachtube recursively by bloating the simulation between $[t_{i-1}, t_i]$ using β_i then using the reach set over-approximation at t_i as the initial set for the time interval $[t_i, t_{i+1}]$.

However, solving (4.4) to get the optimal $\hat{\gamma}$ for each time interval involves solving optimization problems with infinite numbers of constraints (imposed by the infinite set of matrices in $\mathcal{A}$). To overcome the problem, we introduce two different strategies to use local discrepancy functions. The first one only considers discrepancy functions under the Euclidean norm and thus avoids the optimization problem (4.4), while the second tries to transform the problem (4.4) to equivalent or relaxed problems but with finite constraints.

4.2 Fast Discrepancy Function

In this section, we introduce a method to compute the discrepancy function without solving optimization problem (4.4), which leads to an algorithm that can compute the reachtube fast but with possibly larger approximation error compared to the methods discussed in Section 4.3.

4.2.1 Local discrepancy under Euclidean norm

Optimization problem (4.4) tries to find the optimal metric M such that the exponential changing rate $\hat{\gamma}$ of the discrepancy function is minimized. Intuitively, solving (4.4) is relatively complicated as the constraints consider all the possible behaviors of the Jacobian matrix over the local compact set. What if we fix $M = I$? Can it simplify the problem to only finding the minimum γ when M can only be the identity matrix? After revisiting Proposition 2.2 and the definition of the matrix measure under the 2-norm case, we can see that the largest eigenvalue of the symmetric part of the Jacobian matrix can also be used as the exponential change rate of the system.

Lemma 4.3. *For system (2.1), suppose $S \subseteq \mathbb{R}^n$ is a compact convex set, and $[t_1, t_2]$ is a time interval such that for any $x \in S$, $t \in [t_1, t_2]$, $\xi(x, t) \in S$. Suppose $\gamma \in \mathbb{R}$ satisfies: $\forall x \in S$,*

$$eig((J_f^T(x) + J_f(x)))/2 \leq \gamma; \tag{4.5}$$

where $eig()$ means the eigenvalue; then for any $x_1, x_2 \in S$ and for any $t \in [t_1, t_2]$,

$$\|\xi(x_1, t) - \xi(x_2, t)\| \leq \|x_1 - x_2\| e^{\gamma(t - t_1)}.$$

Proof. Let us fix two states $x_1, x_2 \in S$. We have assumed that for any $t \in [t_1, t_2]$, $\xi(x_1, t) \in S, \xi(x_2, t) \in S$. Define $y(t) \equiv \xi(x_2, t) - \xi(x_1, t)$. For a fixed time t, from Lemma 2.1 we have

$$\dot{y}(t) = \left(\int_0^1 J_f(\xi(x_1, t) + sy(t))ds \right) y(t). \tag{4.6}$$

Since S is a convex set, and recalling that $\xi(x_1, t)$, $\xi(x_2, t) \in S$, we can get that for any $s \in [0, 1]$, $\xi(x_1, t) + sy(t) \subseteq S$.

Differentiating $\|y(t)\|^2$, we have

$$\begin{aligned}
\tfrac{d\|y(t)\|^2}{dt} &= \dot{y}^T(t)y(t) + y^T(t)\dot{y}(t) \\
&= y^T(t) \left(\textstyle\int_0^1 J_f^T(\xi(x_1, t) + sy(t))ds \right) y(t) \\
&+ y^T(t) \left(\textstyle\int_0^1 J_f(\xi(x_1, t) + sy(t))ds \right) y(t). \\
&\leq y^T(t) \left(\textstyle\int_0^1 (2\gamma I)ds \right) y(t) \quad \text{[using (4.5)]} \\
&= 2\gamma y^T(t)y(t) \\
&= 2\gamma \|y(t)\|^2.
\end{aligned} \tag{4.7}$$

Integrating both sides over t_1 to any $t \in [t_1, t_2]$, we have

$$\begin{aligned}
&\ln(\|y(t)\|^2) - \ln(\|y(t_1)\|^2) \leq 2\gamma(t - t_1) \\
\Rightarrow &\|y(t)\|^2 \leq \|y(t_1)\|^2 e^{2\gamma(t-t_1)} \\
\Rightarrow &\|\xi(x_1, t) - \xi(x_2, t)\| \leq \|x_1 - x_2\| e^{\gamma(t-t_1)}.
\end{aligned}$$

□

However, computing a bound γ on $eig(J_f^T(x) + J_f(x))/2, x \in S$ is difficult because this bound has to work for the infinite family of matrices. We introduce Algorithm 2 that computes an upper-bound on the eigenvalues of the symmetric part of the Jacobian function.

Algorithm 2: Algorithm `Eig_UB`.

input: $J_f(\cdot)$, $S \subset \mathbb{R}^n$
1 J $\leftarrow J_f(Center(S))$;
2 $\lambda \leftarrow \max(eig(J + J^T)/2)$;
3 Compute $\mathcal{A}$ such that $\forall x \in S, J_f(x) + J_f^T(x)) - (J + J^T) \in \mathcal{A}$;
4 **error** $\leftarrow$ `upperbound of` $\|\mathcal{A}\|_2$;
5 $c \leftarrow \lambda + \frac{\textbf{error}}{2}$;
6 **return** c ;

Algorithm 2 is based on the matrix perturbation theorem (Theorem 4.1). First, the $Center()$ function at Line 1 returns the center point of the compact set S. Then compute the largest eigenvalue λ of the symmetric part of the Jacobian matrix function at the center point at Line 2. At line 3, use interval arithmetic to compute an interval matrix $\mathcal{A}$ such that $\mathcal{A}$ over-approximates the possible values of the error matrix $J_f(x) + J_f^T(x)) - (J + J^T)$, which is the difference between the symmetric part of the Jacobian at any other state and the center state of the compact set S. Compute an upper-bound **error** of the 2-norm of the interval matrix $\mathcal{A}$ at Line 4. Finally, the addition of λ and **error**/2 is returned as an upper-bound of the eigenvalues of all the symmetric parts of the Jacobian matrices over S.

Next we will show that the algorithm indeed computes an upper-bound of the eigenvalues of the symmetric part of the Jacobian matrix over compact set S (Lemma 4.4).

Theorem 4.1 (Matrix Perturbation Theorem [39])**.** *If A and E are $n \times n$ symmetric matrices, then*

$$\lambda_n(E) \leq \lambda_k(A + E) - \lambda_k(A) \leq \lambda_1(E),$$

where $\lambda_i(\cdot)$ is the i^{th} largest eigenvalue of a matrix.

Corollary 4.1. *If A and E are $n \times n$ symmetric matrices, then*

$$|\lambda_k(A + E) - \lambda_k(A)| \leq \|E\|. \tag{4.8}$$

Since A is symmetric, $\|A\| = \sqrt{\lambda_{\max}(A^T A)} = \max(|\lambda(A)|)$. From Theorem 4.1, we have $|\lambda_k(A + E) - \lambda_k(A)| \leq \max\{|\lambda_n(E)|, |\lambda_1(E)|\} = \|E\|$.

Using Corollary 4.1, we will next show that the value returned by Algorithm 2 is an upper bound on the eigenvalues of the symmetric part of $J_f(x)$, where $x \in S$.

Lemma 4.4. *If c is the value returned by Algorithm 2, then for $\forall x \in S$: $J_f^T(x) + J_f(x) \preceq 2cI$.*

Proof. Consider any point $x \in S$. We define the perturbation matrix $E(x) \equiv J_f^T(x) + J_f(x) - (J^T + J)$, and it is straightforward from Line 3 $\forall x \in S, E(x) \in \mathcal{A}$. Since $J_f^T(x) + J_f(x)$ and $J^T + J$ are symmetric matrices, Corollary 4.1 implies that $\lambda_{max}(J_f^T(x) + J_f(x)) - \lambda_{max}(J^T + J) \leq ||E(x)||$.

The *error* term computed in Line 4 is the upper bound on $||E(x)||$. Therefore, $\lambda_{max}(J_f^T(x) + J_f(x)) \leq \lambda_{max}(J^T + J) + error$. In Line 5 set c equals to $\lambda_{max}((J^T + J)/2)+\mathbf{error}/2$. Thus, $\lambda_{max}(J_f^T(x) + J_f(x)) \leq 2c$, which immediately indicates that $\forall x \in S : J_f^T(x) + J_f(x) \preceq 2cI$. □

Let $S \subset \mathbb{R}^n$ be a compact closed set and $E : S \to \mathbb{R}^{n\times n}$ be a function that maps a state $x \in S$ to an n-by-n matrix. If every entry of $E(x)$, denoted by $e_{ij}(x), i, j = 1, \ldots, n$, is a continuous function over S, then we have $\|E(x)\| \leq \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{e}_{ij}^2}$, where $\tilde{e}_{ij} = \sup_{x\in S}|e_{ij}(x)|$, $\forall i, j = 1, \ldots, n$.[1]

The third equation of Theorem 2.1 gives us an effective way to compute the Frobenius norm of the interval matrix $\mathcal{A}$, which can be used as an upper bound on the 2-norm of $\mathcal{A}$.

According to Lemma 4.1, the upper bound of the symmetric part of the Jacobian matrix always exists. In Algorithm 2, because J_f is a real matrix, the maximum eigenvalue λ of $(J^T + J)/2$ is bounded. Assuming that each component of $E(x) = J_f^T(x)+J_f(x)-J^T-J$ is continuous over the closed set S, then we can find the upper bound of $\|E(x)\|$, so the "error" term is also bounded. Therefore, the value returned by Algorithm 2 for continuous-term Jacobian function over compact set is always bounded.

Since $J_f : S \to \mathbb{R}^{n\times n}$ is bounded as stated earlier, there always exists an upper bound γ for the eigenvalues of $(J_f^T(x)+J_f(x))/2$ that can be computed using Algorithm 2. Also, the set S above can be chosen to be a coarse over-approximation of the reach set, obtained using the Lipschitz constant [17]. Using the computed S and γ, Lemma 4.3 provides a bound on the 2-norm distance between trajectories.

Given the simulation result of $\xi(x_1, t)$, for any other initial state x_2 such that $\|x_1 - x_2\| \leq c$, we will have that $\forall t \in [t_1, t_2], \|\xi(x_1, t) - \xi(x_2, t)\| \leq ce^{\gamma(t-t_1)}$. That means that at any time $t \in [t_1, t_2]$, $\xi(x_2, t)$ is contained in the hyber-ball centered at $\xi(x_1, t)$ with radius $ce^{\gamma(t-t_1)}$. Thus, a discrepancy function for system (2.1) over S is given by $\beta(\|x_1-x_2\|, t) = \|x_1-x_2\|e^{\gamma(t-t_1)}$.

Example 4.1. *Consider a 2-dimensional nonlinear system over the set* $S = \{x = [v, w]^T \mid v \in [-2, -1], w \in [2, 3]\}$

$$\dot{v} = \tfrac{1}{2}(v^2 + w^2); \quad \dot{w} = -v. \tag{4.9}$$

[1]The same conclusion can also be obtained by using the fact that $\|A\|_2 \leq \|A\|_F$ for any matrix A.

The Jacobian matrix of the system is

$$\begin{bmatrix} v & w \\ -1 & 0 \end{bmatrix}. \tag{4.10}$$

Using Algorithm 2 we obtain $\gamma = 1.0178$ *as an upper bound on the eigenvalues of the symmetric part of the Jacobian matrix over* S*. Using Lemma 4.3, we obtain the following discrepancy function for this system:*

$$\|\xi(x_1, t) - \xi(x_2, t)\| \leq \|x_1 - x_2\| e^{1.0178t},$$

for as long as the trajectories remain inside S*.*

4.2.2 Algorithm to compute reach set using fast discrepancy function

In this section, we introduce Algorithm `LDF2` which uses Lemma 4.3 and Algorithm 2 to compute a $(B_\delta(R_0), T)$-reachtube of system (2.1), where R_0 is the initial rectangle of the simulation ψ.

Algorithm 3: Algorithm `LDF2`.

input: $\psi = \{(R_k, t_k)_{i=0}^n\}$,$J_f(\cdot)$,$L_f$,$\delta, \epsilon$

$\Delta \leftarrow \delta$,$b \leftarrow$ zeros(k) ;
for *i = 1:n* **do**
  $\tau \leftarrow t_i - t_{i-1}$;
  $d \leftarrow (\Delta + \epsilon)e^{L_f \tau}$;
  S $\leftarrow$ `hull`$(R_{i-1}, R_i) \oplus B_d(0)$;
  $\gamma[i] \leftarrow$ `Eig_UB`$(J_f(\cdot), S)$;
  $O_i \leftarrow B_{\delta'}$ (`hull`(R_{i-1}, R_i)) where $\delta' = \max\{(\Delta + \epsilon), (\Delta + \epsilon)e^{\gamma[i]\tau}\}$;
  $\Delta \leftarrow (\Delta + \epsilon)e^{\gamma[i]\tau}$;
  $\mathcal{R} \leftarrow \mathcal{R} \cup [O_i, t_i]$;
end
return $\mathcal{R}$;

Algorithm 3 shows the pseudocode for `LDF2` used as the *Bloat*() function in the verification algorithm. `LDF2` takes as input a parameter δ, an error bound for simulation ϵ, the Lipschitz constant L_f, the Jacobian matrix $J_f(\cdot)$ of function f, and a $(\theta, \tau, \epsilon, T)$-simulation $\psi = \{(R_i, t_i)\}, i = 0, 1, \ldots, k$. It returns an $(B_\delta(R_0), T)$-reachtube.

The algorithm starts with the initial set $B_\delta(R_0)$ and with $\Delta = \delta$. In each iteration of the **for**-loop it computes a rectangle O_i of the reachtube corresponding to the time interval $[t_{i-1}, t_i]$. In the i^{th} iteration, Δ is updated so that $B_\Delta(R_{i-1})$ is an over-approximation of the reachable states from $B_\delta(R_0)$ at t_{i-1} (Lemma 4.6). In Lines 4 and 5, a set S is computed by bloating the convex hull $\texttt{hull}(R_{i-1}, R_i)$ by a factor of $d = (\Delta + \epsilon)e^{L_f(t_i - t_{i-1})}$. The set S will later be proved to be a (coarse) over-approximation of the reachtube from $B_\Delta(R_{i-1})$ over the time interval $[t_{i-1}, t_i]$ (Lemma 4.5). At Line 6 an upper bound on the maximum eigenvalue of the symmetric part of the Jacobian over the set S is computed using Algorithm 2 (Lemma 4.4). At Line 7 the rectangle O_i is computed as an over-approximation of the reach set during the time interval $[t_{i-1}, t_i]$. Then Δ is updated as $(\Delta + \epsilon)e^{\gamma[i](t_i - t_{i-1})}$ for the next iteration.

Next, we are going to prove that Algorithm 3 returns an $(B_\delta(R_0), T)$-reachtube. Firstly, we need to prove that the set S computed at Line 5 satisfies the assumption in Lemma 4.3. That is, in the i^{th} iteration of the loop, the computed S is an over-approximation of the set of states that can be reached by the system from $B_\Delta(R_{i-1})$ over the time interval $[t_{i-1}, t_i]$.

Lemma 4.5. *In the i^{th} iteration of the loop of Algorithm 3,*

$$\texttt{Reach}(B_\Delta(R_{i-1}), [t_{i-1}, t_i]) \subseteq S.$$

Proof. Let $\xi(\theta, t)$ denote the actual trajectory from θ, where θ is the initial state of ψ. By Definition 2.1 for ψ, it is known that $\theta \in R_0$ and $\forall i = 1, \ldots, k, \xi(\theta, t_i) \in R_i$.

For a fixed iteration number i, consider state $x = \xi(\theta, t_{i-1}) \in R_{i-1}$ from Definition 2.1. We know from the definition of a simulation (Definition 2.1) that for any $t \in [t_{i-1}, t_i]$, $\xi(x, t) \in hull(R_{i-1}, R_i)$. Now consider any state $x' \in B_\Delta(R_{i-1})$. Since L_f is the Lipschitz constant of f, using Gronwall's inequality we have that

$$\|\xi(x, t) - \xi(x', t)\| \leq \|x - x'\| e^{L_f(t - t_{i-1})}.$$

Since $\|x - x'\| \leq \Delta + \epsilon$, we have

$$\|\xi(x, t) - \xi(x', t)\| \leq (\Delta + \epsilon) e^{L_f(t - t_{i-1})}.$$

Therefore, $\xi(x', t) \in hull(R_{i-1}, R_i) \oplus B_{(\Delta+\epsilon)e^{L_f(t_i - t_{i-1})}}(0) = S$. □

Next, we inductively prove that O_i computed at Line 7 is an over-approximation of the reach set during the time interval $[t_{i-1}, t_i]$.

Lemma 4.6. *For any $i = 1, \ldots, k$, $\texttt{Reach}(B_\delta(R_0), t_i) \subseteq B_{\Delta_i}(R_i)$, and $\texttt{Reach}(B_\delta(R_0), [t_{i-1}, t_i]) \subseteq O_i$, where Δ_i is Δ after Line 8 is executed in the i^{th} iteration.*

Proof. In this proof, let $\xi(\theta, \cdot)$ denote the trajectory from θ. From the Definition 2.1 for ψ, we know that $\theta \in R_0$ and $\forall i = 1, \ldots, k, \xi(\theta, t_i) \in R_i$. Let S_i denote S after Line 5 is executed in the i^{th} iteration. The lemma is proved by induction on i. Note that the initial set is $B_\delta(R_0)$, and before the **for**-loop, Δ_0 is set as δ.

When $i = 1$, we already have $\texttt{Reach}(B_\delta(R_0), t_0) = B_\delta(R_0) = B_{\Delta_0}(R_0)$.

Lemma 4.5 indicates that $\forall t \in [t_0, t_1], \texttt{Reach}(B_{\Delta_0}(R_0), [t_0, t_1]) \subseteq S$. And consider state $x = \theta \in R_0$, we also know $\xi(x, t) \in \texttt{hull}(R_0, R_1)$ and $\xi(x, t_1) \in R_1$. From Lemma 4.3, it follows that for $\forall x' \in B_{\Delta_0}(R_0), \forall t \in [t_0, t_1]$,

$$\|\xi(x, t) - \xi(x', t)\| \leq \|x - x'\| e^{\gamma[1](t - t_0)}.$$

At Line 8, $\Delta_1 \leftarrow (\Delta_0 + \epsilon)e^{\gamma[1](t_1 - t_0)}$. Since $\gamma[1]$ could be positive or negative, $\max_{t \in [t_0, t_1]} \|x - x'\| e^{\gamma[1](t - t_0)} = \max\{\Delta_1, \Delta_0 + \epsilon\}$. Therefore,

$$\texttt{Reach}(B_\delta(R_0), [t_0, t_1]) \subseteq \texttt{hull}(R_0, R_1) \oplus B_{\max\{\Delta_1, \Delta_0 + \epsilon\}}(0) = O_1,$$

and at time t_1, $\xi(x', t_1)$ is at most Δ_1 distance to $\xi(x, t_1) \in R_1$, so

$$\texttt{Reach}(B_\delta(R_0), t_1) = \texttt{Reach}(B_{\Delta_0}(R_0), t_1) \subseteq B_{\Delta_1}(R_1).$$

Assume that the lemma holds for $i = m - 1$, which means we have $\texttt{Reach}(B_\delta(R_0), t_{m-1}) \subseteq B_{\Delta_{m-1}}(R_{m-1})$. Next we prove the lemma holds for $i = m$ as well. Consider state $x = \xi(\theta, t_{m-1}) \in R_{m-1}$, $\forall t \in [t_{m-1}, t_m]$; by definition it follows that $\xi(x, t) \in \texttt{hull}(R_{m-1}, R_m)$ and $\xi(x, t_m) \in R_m$. $\forall x' \in B_{\Delta_{m-1}}(R_{m-1}), \forall t \in [t_{m-1}, t_m]$, from Lemma 4.3

$$\|\xi(x, t) - \xi(x', t)\| \leq \|x - x'\| e^{\gamma[m](t - t_{m-1})}.$$

Note at Line 8, $\Delta_m \leftarrow (\Delta_{m-1} + \epsilon)e^{\gamma[m](t_m - t_{m-1})}$. Therefore,

$$\texttt{Reach}(B_{\Delta_{m-1}}(R_{m-1}), [t_{m-1}, t_m]) \subseteq hull(R_{m-1}, R_m) \oplus B_{\max\{\Delta_m, \Delta_{m-1}+\epsilon\}}(0) = O_i.$$

Also we have $\texttt{Reach}(B_\delta(R_0), t_{m-1}) \subseteq B_{\Delta_{m-1}}(R_{m-1})$, so

$$\texttt{Reach}(B_\delta(R_0), [t_{m-1}, t_m]) \subseteq O_i.$$

And at time t_m, $\xi(x', t_m)$ is at most Δ_m distance to $\xi(x, t_m) \in R_m$. Hence, $\texttt{Reach}(B_{\Delta_{m-1}}(R_{m-1}),\ t_m) \subseteq B_{\Delta_m}(R_m)$. Recall that $\texttt{Reach}(B_\delta(R_0), t_{m-1}) \subseteq B_{\Delta_{m-1}}(R_{m-1})$, thus $\texttt{Reach}(B_\delta(R_0), t_m) \subseteq B_{\Delta_m}(R_m)$. □

From Lemma 4.6, the following main theorem of this section follows. It states that the Algorithm LDF2 soundly over-approximates the reachable states from $B_\delta(R_0)$.

Theorem 4.2. *For any (x, τ, ϵ, T)-simulation $\psi = (R_0, t_0) \dots (R_k, t_k)$ and any constant $\delta \geq 0$, a call to $\texttt{LDF2}(\psi, \delta, \epsilon)$ returns a $(B_\delta(R_0), T)$-reachtube.*

4.3 Local Optimal Discrepancy Function

Algorithm LDF2 has fundamental drawbacks that prevent it from working for a large class of systems in practice. One drawback is that the bloating factor (or discrepancy function) computed by Lemma 4.3 grows (or shrinks) with time exactly at the same rate along all the dimensions of the system, and this rate is computed by bounding the eigenvalues of the symmetric part of the Jacobian matrix.

For example, the simple linear system

$$\dot{x} = \begin{bmatrix} 0 & 3 \\ -1 & 0 \end{bmatrix} x$$

has eigenvalues $\pm\sqrt{3}i$, and therefore has oscillating trajectories. The actual distance between neighboring trajectories is at most a constant times their initial distance; however, the discrepancy function computed by Lemma 4.3 will bound this distance between trajectories, in all dimensions, as an exponentially growing function $Ce^{\lambda t}$, where $\lambda = 1$ is the largest eigenvalue of the

symmetric part of the Jacobian matrix.[2]

Furthermore, Algorithm 2 uses a coarse method for bounding the largest eigenvalue of the Jacobian matrix, which leads to an undesirable level of conservatism to the point that even for certain contractive systems, the computed reach set over-approximation may not converge over time.

To overcome the shortcomings of Algorithm `LDF2`, we will try to solve the optimization problem (4.4):

$$\begin{aligned} \min_{\hat{\gamma}\in\mathbb{R}, M\in\mathbb{R}^{n\times n}} \quad & \hat{\gamma} \\ \text{s.t} \quad & A^T M + MA \preceq \hat{\gamma} M, \quad \forall A \in \mathcal{A} \\ & M \succ 0 \end{aligned}$$

by replacing the constraints $A^T M + MA \preceq \hat{\gamma} M, \quad \forall A \in \mathcal{A}$ with infinite number of matrices with certain representative matrices and analyzing the consequences of such replacement.

To simplify the presentation, we assume that the solutions (i.e., trajectories) for (2.1) can be obtained exactly. Later at the end of Section 4.3.4, we will discuss how the algorithms work with validated simulations with guaranteed error bounds (see also for a detailed treatment of this [18]).

4.3.1 Vertex matrix constraints method

Proposition 2.1 establishes that an interval matrix is equivalent to the convex hull of its vertex matrices. That means each constant matrix A in the interval matrix $\mathcal{A}$ will have a representation based on elements of $\texttt{VT}(\mathcal{A})$. This allows us to simplify the optimization problem in Equation (4.4) to one with a finite number of constraints, based on the vertex matrices.

The next lemma provides a method for computing discrepancy functions from the vertex matrices of an interval matrix.

Lemma 4.7. *Let $S \subseteq \mathbb{R}^n$ be the set of states and $[t_1, t_2]$ be a time interval such that for any state $x \in S$, we have $\xi(x,t) \in S$ for $t \in [t_1, t_2]$. Let M be a positive definite $n \times n$ matrix. If there exists an interval matrix $\mathcal{A}$ such that*

[2] In [17], a simple coordinate transformation method is introduced to address this problem, but that requires user intervention and adds an approximation error that is of the order of the matrix condition number.

(a) $\forall\ x \in S,\ J_f(x) \in \mathcal{A}$, *and*

(b) $\exists\ \hat{\gamma} \in \mathbb{R},\ \forall\ A_i \in \texttt{VT}(\mathcal{A}),\ \ A_i^T M + M A_i \preceq \hat{\gamma} M$.

Then for any $x_1, x_2 \in S$ *and* $t \in [t_1, t_2]$*:*

$$\|\xi(x_1, t) - \xi(x_2, t)\|_M \leq e^{\frac{\hat{\gamma}}{2}(t-t_1)} \|x_1 - x_2\|_M.$$

Proof. From Proposition 2.1, we know that $J_f(x) \in \mathcal{A} = \texttt{hull}(\texttt{VT}(\mathcal{A}))$. It follows from (a) and Lemma 4.2 that for any $t \in [t_1, t_2]$, there exists a matrix $A \in \mathcal{A}$ such that $\dot{y}(t) = Ay(t)$, and $A \in \texttt{hull}\{A_1, A_2, \ldots, A_N\}$. Using this, at any time $t \in [t_1, t_2]$, the derivative of $\|y(t)\|_M^2$ can be written as:

$$\begin{aligned}
\frac{d\|y(t)\|_M^2}{dt} &= y^T(t) A^T M y(t) + y^T(t) M A y(t) \\
&= y^T(t) \left(\left(\sum_{i=1}^N \alpha_i A_i^T \right) M + M \left(\sum_{i=1}^N \alpha_i A_i \right) \right) y(t) \\
&= y^T(t) \left(\sum_{i=1}^N \alpha_i \left(A_i^T M + M A_i \right) \right) y(t) \\
&\leq y^T(t) \left(\sum_{i=1}^N \alpha_i \hat{\gamma} M \right) y(t) \quad \text{[using (b)]} \\
&= \hat{\gamma} y^T(y) M y(t) = \hat{\gamma} \|y(t)\|_M^2.
\end{aligned}$$

By applying Grönwall's inequality, we obtain $\|y(t)\|_M \leq e^{\frac{\hat{\gamma}}{2}(t-t_1)} \|y(t_1)\|_M$. □

Lemma 4.7 suggests the following bilinear optimization problem for finding discrepancy over compact subsets of the state space:

$$\begin{aligned}
\min_{\hat{\gamma} \in \mathbb{R}, M \in \mathbb{R}^{n \times n}} \quad & \hat{\gamma} \\
\text{s.t.} \quad & A_i^T M + M A_i \preceq \hat{\gamma} M, \text{for each } A_i \in \texttt{VT}(\mathcal{A}) \\
& M \succ 0.
\end{aligned} \tag{4.11}$$

Letting $\hat{\gamma}_{\max}$ be the maximum of the eigenvalues of $A_i^T + A_i$ for all i, then $A_i^T + A_i \preceq \hat{\gamma}_{\max} I$ (i.e., $M = I$) holds for every A_i, so a feasible solution exists for (4.11). To obtain a minimal feasible solution for $\hat{\gamma}$, we choose a range of $\gamma \in [\gamma_{\min}, \gamma_{\max}]$, where $\gamma_{\min} < \gamma_{\max}$ and perform a line search of $\hat{\gamma}$ over $[\gamma_{\min}, \gamma_{\max}]$. Note that if $\hat{\gamma}$ is fixed, then (4.11) is a semidefinite program

(SDP), and a feasible solution can be obtained by an SDP solver. As a result, we can solve (4.11) using a line search strategy, where an SDP is solved at each step. The solution we obtain using this technique may not be optimal, but we note that any feasible $\hat{\gamma}$ and M conservatively capture the behaviors of the difference between trajectories. Further, in practice, we can always choose a negative enough lower bound $\hat{\gamma}_{\min}$, such that if $\hat{\gamma} < \hat{\gamma}_{\min}$, then we can use $\hat{\gamma}_{\min}$ as a sufficient relaxation (upper bound) for $\hat{\gamma}$.

The above process for identifying a feasible (optimal) $\hat{\gamma}$ and a corresponding M can be used to compute reach set over-approximations, based on the discrepancy function $\beta(\|x_1 - x_2\|_M, t) = e^{\frac{\hat{\gamma}}{2}(t-t_1)}\|x_1 - x_2\|_M$.

Example 4.2. *Consider system (4.9) over the given compact set S as in Example 4.1. By solving optimization problem (4.11), we can get $\hat{\gamma} = -0.6$ and $M = \begin{bmatrix} 2.7263 & -1.3668 \\ -1.3668 & 6.7996 \end{bmatrix}$. Then by invoking Lemma 4.7, we obtain the discrepancy function*

$$\|\xi(x_1, t) - \xi(x_2, t)\|_M \leq \|x_1 - x_2\|_M e^{-0.3t},$$

for as long as the trajectories remain inside S.

Compared with Example 4.1, the interval matrix constraints method provides a tighter discrepancy function than that provided by Algorithm `LDF2` since it computes a local optimal exponential change rate between trajectories over S.

The over-approximation computed using this method is less conservative than the method based on the 2-norm (Section 2.3), because the optimal metric is searched for the minimum possible exponential change rate, which is achieved by allowing the amount of bloating in each direction to be different, instead of the uniform rate for all directions as in Algorithm `LDF2`.

This approach is computationally more intensive than the `LDF2` method due to the potentially $O(2^{n^2})$ matrices in $\texttt{VT}(\mathcal{A})$ that appear in the SDP (4.11). In the next section, we present a second method that avoids the exponential increase in the number of constraints in (4.11).

4.3.2 Interval matrix norm method

We present a second method for computing discrepancy functions based on interval matrix norms, which uses the center and range matrices to characterize the norm of the interval matrix $\mathcal{A}$. The next lemma provides a method to compute a discrepancy function using the matrix norm of an interval matrix.

Lemma 4.8. *Let $S \subseteq \mathbb{R}^n$ be sets of states and $[t_1, t_2]$ be a time interval such that for any $x \in S$, $\xi(x,t) \in S$, for $t \in [t_1, t_2]$, where $0 \leq t_1 < t_2 \leq T$. Let M be a positive definite $n \times n$ matrix. If there exists an interval matrix $\mathcal{A}$ such that*

(a) $\forall\ x \in S$, $J_f(x) \in \mathcal{A}$, and

(b) $\exists\ \hat{\gamma} \in \mathbb{R}$, such that $\mathtt{CT}(\mathcal{A})^T M + M\mathtt{CT}(\mathcal{A}) \preceq \hat{\gamma} M$,

then for any $x_1, x_2 \in S$ and $t \in [t_1, t_2]$:

$$\|\xi(x_1,t) - \xi(x_2,t)\|_M \leq e^{\left(\frac{\hat{\gamma}}{2} + \frac{\delta}{2\lambda_{min}(M)}\right)(t-t_1)} \|x_1 - x_2\|_M, \tag{4.12}$$

where $\delta = \sqrt{\|\|\mathcal{D}\|\|_1 \|\|\mathcal{D}\|\|_\infty}$, and

$$\mathcal{D} = \{D \mid \exists A \in \mathcal{A} \text{ such that } D = (A - \mathtt{CT}(\mathcal{A}))^T M + M(A - \mathtt{CT}(\mathcal{A}))\}$$

is also an interval matrix.

Proof. Fix any $x_1, x_2 \in S$. Let $A_c = \mathtt{CT}(\mathcal{A})$ and $A_r = \mathtt{RG}(\mathcal{A})$, so $\mathcal{A} = \mathtt{Interval}([A_c - A_r, A_c + A_r])$. We can express $\mathcal{A}$ as the Minkowski sum of A_c and $\mathcal{G}$, which we denote as $A_c \oplus \mathcal{G}$, where $\mathcal{G} = \mathtt{Interval}([-A_r, A_r]) = \{G \mid \exists A \in \mathcal{A} \text{ such that } G = A - \mathtt{CT}(\mathcal{A})\}$. We use a standard property of norms to bound the 2-norm as follows (see [26], page 57):

$$\begin{aligned} \|G^T M + MG\|_2 &\leq \sqrt{\|G^T M + MG\|_1 \|G^T M + MG\|_\infty} \\ &\leq \sqrt{\sup\{\|D\|_1 | D \in \mathcal{D}\} \sup\{\|D\|_\infty | D \in \mathcal{D}\}} \\ &\leq \sqrt{\|\|\mathcal{D}\|\|_1 \|\|\mathcal{D}\|\|_\infty} = \delta, \end{aligned} \tag{4.13}$$

which uses the fact that $G^T M + MG \in \mathcal{D}$. As $\lambda_{\min}(M)$ is the minimum eigenvalue of the positive definite matrix M, then $\forall y \neq 0$,

$$0 < \lambda_{\min}(M)\|y\|^2 \leq y^T M y. \tag{4.14}$$

Moreover, $\forall G \in \mathcal{G}$, and any vector $y \in \mathbb{R}^n$

$$y^T(G^TM + MG)y \leq \|G^TM + MG\|_2\|y\|^2. \tag{4.15}$$

Combining the above with (4.13) yields

$$y^T(G^TM + MG)y \quad \leq \quad \delta\|y\|^2. \tag{4.16}$$

Since $x_1, x_2 \in S$ and $t \in [t_1, t_2]$, it follows from Lemma 4.2 that $\exists G \in \mathcal{G}$, such that the distance between trajectories $y(t) = \xi(x_1, t) - \xi(x_2, t)$ satisfies $\dot{y}(t) = (A_c + G)y(t)$. Considering the above inequalities, we have that

$$\begin{aligned}
\frac{d\|y(t)\|_M^2}{dt} &= y^T(t)\left((A_c^T + G^T)M + M(A_c + G)\right)y(t) \\
&= y^T(t)(A_c^TM + MA_c + G^TM + MG)y(t) \\
&\leq \hat{\gamma}y^T(t)My(t) + y^T(t)(G^TM + MG)y(t) \\
&\leq \hat{\gamma}\|y(t)\|_M^2 + \delta\|y\|^2(t) \quad \text{[using (4.16)]} \\
&\leq \hat{\gamma}\|y(t)\|_M^2 + \delta\frac{\|y(t)\|_M^2}{\lambda_{\min}(M)}. \quad \text{[using (4.14)]}
\end{aligned}$$

The lemma follows by applying Grönwall's inequality. □

In general, Lemma 4.8 provides the discrepancy function

$$\beta(\|x_1 - x_2\|_M, t) = e^{\left(\frac{\hat{\gamma}}{2} + \frac{\delta}{2\lambda_{\min}(M)}\right)(t-t_1)}\|x_1 - x_2\|_M,$$

where an M and $\hat{\gamma}$ need to be selected. This suggests solving the following alternative optimization problem to compute a discrepancy function over compact subsets of the state space.

$$\begin{aligned}
\min_{\hat{\gamma}\in\mathbb{R}, M\in\mathbb{R}^{n\times n}} \quad & \hat{\gamma} \\
\text{s.t} \quad & A_c^TM + MA_c \preceq \hat{\gamma}M, A_c = \mathtt{CT}(\mathcal{A}) \\
& M \succ 0.
\end{aligned} \tag{4.17}$$

Remark 4.1. *In Lemma 4.8, δ is computed as $\sqrt{|||\mathcal{D}|||_1|||\mathcal{D}|||_\infty}$, where $\mathcal{D}$ is an interval matrix. To compute the 1-norm or the infinity-norm of the interval matrix, Theorem 2.1 provides an efficient way which only needs to compute*

the 1 or infinite norm of the absolute value of the interval matrix's center matrix $|\mathtt{CT}(\mathcal{D})|$ *and the range matrix* $\mathtt{RG}(\mathcal{D})$.

Example 4.3. *For system* (4.9) *over the given compact set* S *as in Example 4.1, we can obtain* $\hat{\gamma} = -0.8$ *and* $M = \begin{bmatrix} 2.4431 & -1.0511 \\ -1.0511 & 4.5487 \end{bmatrix}$ *by solving optimization problem* (4.17)*, and* $\delta = 1.4162$*. Applying Lemma 4.8, a discrepancy function for* (4.9) *is given by*

$$\|\xi(x_1, t) - \xi(x_2, t)\|_M \leq \|x_1 - x_2\|_M e^{0.3081t},$$

for as long as the trajectories remain inside S.

Compared with Example 4.1 and 4.2, the interval matrix norm method produces a discrepancy that is tighter than Algorithm `LDF2` but more conservative than the vertex matrix constraints method.

The computations required to produce the discrepancy using the interval matrix norm method are significantly less intensive than for the vertex matrix constraints method. But this comes at the price of decreasing the accuracy (i.e., increasing the conservativeness), due to the positive error term $\frac{\delta}{2\lambda_{\min}(M)}$ that is added to $\frac{\hat{\gamma}}{2}$ in (4.12). In practice, we want to make the compact sets S small so that δ (and by extension the exponential term in (4.12)) remains small.

Lemmas 4.7 and 4.8 provide bounds on the M-norm distance between trajectories. Given the simulation result of $\xi(x_1, t)$, for any other initial state x_2 such that $\|x_1 - x_2\|_M \leq c$, we will have that $\forall t \in [t_1, t_2], \|\xi(x_1, t) - \xi(x_2, t)\|_M \leq ce^{\frac{\gamma'}{2}(t-t_1)}$ ($\gamma' = \hat{\gamma}$ for Lemma 4.7 and $\gamma' = \hat{\gamma} + \frac{\delta}{\lambda_{min}(M)}$ for Lemma 4.8). This means that at any time $t \in [t_1, t_2]$, $\xi(x_2, t)$ is contained in the ellipsoid centered at $\xi(x_1, t)$ defined by the set of points x that satisfy

$$\|(\xi(x_1, t) - x)\|_M^2 \leq ce^{\gamma'(t-t_1)}.$$

That is, $\xi(x_2, t)$ is contained within ellipsoid $E_{M, ce^{\gamma'(t-t_1)}}(\xi(x_1, t))$ (see the definition of ellipsoid in Section 2.1.3).

4.3.3 Algorithm to compute local optimal reach set

Given an initial set $B_\delta(x)$ and time bound T, Lemmas 4.7 and 4.8 provide discrepancy functions over compact sets in the state space and over a bounded time horizon. To compute the reach set of a nonlinear model from a set of initial states over a long time horizon $[0, T]$, we will divide the time interval $[0, T]$ into smaller intervals $[0, t_1], \ldots, [t_{k-1}, t_k = T]$, and compute a piecewise discrepancy function, where each piece is relevant for a smaller portion of the state space and the time.

Consider two adjacent subintervals of $[0, T]$, $a = [t_1, t_2]$ and $b = [t_2, t_3]$. Let β_a, β_b be the discrepancy functions for the intervals a and b. β_a defines an ellipsoid $E_{M_a, c_a(t_2)}(\xi(x_0, t_2))$ that contains $\texttt{Reach}(B_\delta(x), t_2)$ and β_b provides M_b and $c_b(t)$ such that $\texttt{Reach}(B_\delta(x), t_2) \subseteq E_{M_b, c_b(t_2)}(\xi(x_0, t_2))$. To over-approximate the reach set for the interval b, we require that $c_b(t_2)$ is chosen so that at the transition time t_2:

$$E_{M_a, c_a(t_2)}(\xi(x_0, t_2)) \subseteq E_{M_b, c_b(t_2)}(\xi(x_0, t_2)). \tag{4.18}$$

It is a standard SDP problem to compute the minimum value for $c_b(t_2)$ that ensures (4.18) (see, for example [40]). This minimum value is used as $c_b(t_2)$ for computing the reachtube for time interval b.

Let Ea denote the ellipsoid $E_{M_a, c_a(t_2)}(\xi(x_0, t_2))$ and Eb denote the ellipsoid $E_{M_b, c}(\xi(x_0, t_2))$. The problem of minimizing $c_b(t_2)$, given $M_a, M_b, c_a(t_2)$, such that Equation (4.18) holds can be expressed using the following optimization problem:

$$\begin{aligned} &\min \quad c \\ &s.t. \quad Eb \supseteq Ea. \end{aligned} \tag{4.19}$$

Then let $c_b(t_2)$ be equal to the solution.

We can transfer problem (4.19) to the flowing *sum-of-squares* problem using the "S procedure" [41] to make it solvable by SDP solvers:

$$\begin{aligned} &\min \quad c \\ s.t. \quad &c - \|x - \xi(x_0, t_2)\|^2_{M_b} - \lambda \left(c_a(t_2) - \|x - \xi(x_0, t_2)\|^2_{M_a}\right) \geq 0, \\ &\lambda \geq 0. \end{aligned} \tag{4.20}$$

4.3.4 Reachtube over-approximation algorithm

We present an algorithm to compute a $(B_\delta(x), T)$-reachtube for system (2.1) using the results from Section 4.3.1 and 4.3.2. Given an initial set $B_\delta(x)$, which is a ball centered at x, and time bound T, Algorithm `LDFM` computes a sequence of time-stamped sets (R_1, t_1), $(R_2, t_2), \ldots, (R_k, t_k)$, such that the reach set from $B_\delta(x_0)$ is contained in the union of the sets.

In Algorithm `LDFM` we assume that the exact simulation of the solution $\xi(x, t)$ exists and can be represented as a sequence of points and hyper-rectangles for ease of exposition. At the end of this section (Remark 4.3), we will introduce how to modify Algorithm `LDFM` to adopt validated simulation.

The inputs to Algorithm `LDFM` are as follows: (1) A simulation ψ of the trajectory $\xi(x, t)$, where $x = \xi(x, t_0)$ and $t_0 = 0$, represented as a sequence of points $\xi(x, t_0), \ldots, \xi(x, t_k)$ and a sequence of hyper-rectangles $Rec(t_{i-1}, t_i) \subseteq \mathbb{R}^n$. That is, for any $t \in [t_{i-1}, t_i]$, $\xi(x, t) \in Rec(t_{i-1}, t_i)$. (2) The Jacobian matrix $J_f(\cdot)$. (3) A Lipschitz constant L for the vector field (this can be replaced by a local Lipschitz constant for each time interval). (4) A matrix M_0 and constant c_0 such that $B_\delta(x) \subseteq E_{M_0,c_0}(x)$. The output is a $(B_\delta(x), T)$-Reachtube.

Algorithm `LDFM` uses Lemma 4.7 to update the coordinate transformation matrix M_i to ensure an optimal exponential rate γ_i of the discrepancy function in each time interval $[t_{i-1}, t_i]$. It will solve the optimization problem (4.11) in each time interval to get the local optimal rate, and solve the optimization problem (4.18) when it moves forward to the next time interval.

The algorithm proceeds as follows. The diameter of the ellipsoid containing the initial set $B_\delta(x)$ is computed as the initial set size (Line 1). At Line 4, $Rec(t_{i-1}, t_i)$, which contains the trajectory between $[t_{i-1}, t_i]$ is bloated by the factor $\delta_{i-1}e^{L\Delta t}$ which gives the set S that is guaranteed to contain $\texttt{Reach}(B_\delta(x), t)$ for every $t \in [t_{i-1}, t_i]$ (see Lemma 4.5). Next, at Line 5, an interval matrix $\mathcal{A}$ containing $J_f(x)$, for each $x \in S$ is computed. The matrix is guaranteed to exist by Lemma 4.1. The "if" condition in Line 6 determines whether the M_{i-1}, γ_{i-1} used in the previous iteration satisfy the conditions of Lemma 4.7 (γ_0 when $i = 1$, where γ_0 is an initial guess). This condition will avoid performing updates of the discrepancy function if it is unnecessary. If the condition is satisfied, then M_{i-1} is used again for the current iteration i (Lines 7, 8, and 9) and γ_i will be computed as the smallest possible value such

Algorithm 4: Algorithm `LDFM`

input : ψ,$J_f(\cdot)$,L,M_0, c_0
initially: $\mathcal{R} \leftarrow \emptyset, \gamma_0 \leftarrow -100$

1 $\delta_0 = dia\left(E_{M_0,c_0}(x)\right)$;
2 **for** i = *1:k* **do**
3 $\quad \Delta t \leftarrow t_i - t_{i-1}$;
4 $\quad S \leftarrow B_{\delta_{i-1}e^{L\Delta t}}(Rec(t_{i-1}, t_i))$;
5 $\quad \mathcal{A} \leftarrow \texttt{Interval}[B, C]$ such that $J_f(x) \in \texttt{Interval}[B, C], \forall x \in S$;
6 $\quad$ **if** $\forall V \in \texttt{VT}(\mathcal{A}) : V^T M_{i-1} + M_{i-1} V \leq \gamma_{i-1} M_{i-1}$ **then**
7 $\quad\quad M_i \leftarrow M_{i-1}$;
8 $\quad\quad \gamma_i \leftarrow \arg\min_{\gamma \in \mathbb{R}} \ \forall V \in \texttt{VT}(\mathcal{A}) : V^T M_i + M_i V \leq \gamma M_i$;
9 $\quad\quad c_{tmp} \leftarrow c_{i-1}$
10 $\quad$ **else**
11 $\quad\quad$ compute M_i, γ_i from Eq. (4.11) ;
12 $\quad\quad$ compute minimum c_{tmp} such that $E_{M_{i-1},c_{i-1}}(\xi(x, t_{i-1})) \subseteq E_{M_i,c_{tmp}}(\xi(x, t_{i-1}))$;
13 $\quad$ **end**
14 $\quad c_i \leftarrow c_{tmp} e^{\gamma_i \Delta t}$;
15 $\quad \delta_i \leftarrow dia(E_{M_i,c_i}(\xi(x, t_i)))$;
16 $\quad R_i \leftarrow B_{\delta'/2}(Rec(t_{i-1}, t_i))$ where $\delta' = \max\{dia\left(E_{M_i,c_{tmp}}\left(\xi(x, t_{i-1})\right)\right), \delta_i\}$;
17 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup [R_i, t_i]$;
18 **end**
19 **return** $\mathcal{R}$;

that Lemma 4.7 holds (Line 8) without updating the shape of the ellipsoid (i.e., $M_i = M_{i-1}$). In this case, the γ_i computed using M_{i-1} in the previous iteration $(i-1)$ may not be ideal (minimum) for the current iteration (i), but we assume it is acceptable. If M_{i-1} and γ_{i-1} do not satisfy the conditions of Lemma 4.7, that means the previous coordinate transformation can no longer ensure an accurate exponential converging or diverging rate between trajectories. Then M_i and γ_i are recomputed at Line 11. For the vertex matrix constraints case, (4.11) is solved to update M_i and γ_i. At Line 12, an SDP is solved to identify the smallest constant c_{tmp} for discrepancy function updating such that

$$E_{M_{i-1},c_{i-1}}(\xi(x, t_{i-1})) \subseteq E_{M_i,c_{tmp}}(\xi(x, t_{i-1})).$$

At Line 14, we compute the updated ellipsoid size c_i such that $E_{M_i,c_i}(\xi(x, t_i))$

contains $\texttt{Reach}(B_\delta(x), t_i)$. An Line 15, the diameter of $E_{M_i,c_i}(\xi(x,t_i))$ is assigned to δ_i for next iteration. At Line 16 the set R_i is computed such that it contains the reach set during time interval $[t_{i-1}, t_i]$. Finally, at Line 17 $\mathcal{R}$ is returned as an over-approximation of the reach set.

Next, we analyze the properties of Algorithm `LDFM`. We first establish that the γ produced by Line 11 is a local optimal exponential converging or diverging rate between trajectories. Then we prove that Algorithm `LDFM` soundly over-approximates the reachable states from $B_\delta(x)$.

The next lemma states that Line 11 computes the locally optimal exponential rate γ for a given interval matrix approximation.

Lemma 4.9. *In the i^{th} iteration of Algorithm* `LDFM`*, suppose $\mathcal{A}$ is the approximation of the Jacobian over $[t_{i-1}, t_i]$ computed in Line 5. If E_{i-1} is the reach set at t_{i-1}, then for all M' and γ' such that $\texttt{Reach}(E_{i-1}, t_i) \subseteq E_{M',c'}(\xi(x,t_i))$ where c' is computed from γ' (Line 14), we have that the γ produced by Line 11 satisfies $\gamma \leq \gamma'$.*

Proof. (sketch) The lemma follows from the fact that any M', γ' that satisfies $A^T M' + M' A \preceq \gamma' M', \forall A \in \mathcal{A}$ results in an ellipsoidal approximation at t_i that over-approximates the reach set; however, at Line 11 we are computing the minimum exponential change rate γ by searching all possible matrices M for the given interval matrix. Thus, the γ value computed at Line 11 is the optimal exponential change rate over local convex set S for the given interval matrix $\mathcal{A}$. □

In other words, the computed γ is the optimal exponential growth rate for any ellipsoidal reach set approximation, based on a given interval matrix approximation for the Jacobian.

Next, we show that R_i computed at Line 16 is an over-approximation of the reach set during time interval $[t_{i-1}, t_i]$.

Lemma 4.10. *For Algorithm* `LDFM`*, at i^{th} iteration, if $\texttt{Reach}(B_\delta(x), t_{i-1}) \subseteq E_{M_{i-1},c_{i-1}}(\xi(x,t_{i-1}))$, then we have at time t_i,*

$$\texttt{Reach}(B_\delta(x), t_i) \subseteq E_{M_i,c_i}(\xi(x,t_i)),$$

and

$$\texttt{Reach}(B_\delta(x), [t_{i-1}, t_i]) \subseteq R_i.$$

Proof. Note that by Lemma 4.7, at any time $t \in [t_{i-1}, t_i]$, any other trajectory $\xi(x', t)$ starting from $x' \in E_{M_{i-1},c_{i-1}}(\xi(x, t_{i-1}))$ is guaranteed to satisfy

$$\|\xi(x,t) - \xi(x',t)\|_{M_i} \leq \|\xi(x,t_{i-1}) - x'\|_{M_i} e^{\frac{\gamma_i}{2}(t-t_{i-1})}. \tag{4.21}$$

Then, at time t_i, the reach set is guaranteed to be contained in the ellipsoid $E_{M_i,c_i}(\xi(x,t_i))$.

At Line 16 we want to compute the set R_i such that it contains the reach set during time interval $[t_{i-1}, t_i]$. According to equation (4.21), at any time $t \in [t_{i-1}, t_i]$, the reach set is guaranteed to be contained in the ellipsoid $E_{M_i,c(t)}(\xi(x,t))$, where $c(t) = c_{tmp}e^{\gamma_i(t-t_{i-1})}$. R_i should contain all the ellipsoids during time $[t_{i-1}, t_i]$. Therefore, it can be obtained by bloating the rectangle $Rec(t_{i-1}, t_i)$ using the largest ellipsoid's radius (half of the diameter). Since $e^{\gamma_i(t-t_{i-1})}$ is monotonic (increasing when $\gamma_i > 0$ or decreasing when $\gamma_i < 0$) with time, the largest ellipsoid during $[t_{i-1}, t_i]$ is either at t_{i-1} or at t_i. So the largest diameter of the ellipsoids is $\max\{dia\left(E_{M_i,c_{tmp}}\left(\xi(x,t_{i-1})\right)\right), \delta_i\}$. Thus, at Line 16

$$\texttt{Reach}(B_\delta(x), [t_{i-1}, t_i]) \subseteq R_i.$$

□

Next, we show that $\mathcal{R}$ returned at Line 17 is an over-approximation of the reach set.

Theorem 4.3. *For any (x,T)-simulation $\psi = \xi(x,t_0), \ldots, \xi(x,t_k)$ and any constant $\delta \geq 0$, a call to* `LDFM`(ψ, δ) *returns a $(B_\delta(x), T)$-reachtube.*

Proof. Using Lemma 4.10, when $i = 1$, because the initial ellipsoid $E_{M_0,c_0}(x)$ contains the initial set $B_\delta(x)$, we have that $E_{M_1,c_1}(\xi(x,t_1))$ defined at Line 15 contains $\texttt{Reach}(B_\delta(x), t_1)$. Also at Line 16, R_1 contains $\texttt{Reach}(B_\delta(x), [t_0, t_1])$. Repeating this analysis for subsequent iterations, we have that $E_{M_i,c_i}(\xi(x,t_i))$ contains $\texttt{Reach}(B_\delta(x), t_i)$, and R_i contains $\texttt{Reach}(B_\delta(x), [t_{i-1}, t_i])$. Therefore, $\mathcal{R}$ returned at Line 17 is a $(B_\delta(x), T)$-Reachtube.

□

Remark 4.2. *Algorithm 4 uses the vertex matrix constraints method in Section 4.3.1. To apply the interval matrix norm method in Section 4.3.2, just*

modify Lines 6, 8, 11, and 14 according to Lemma 4.8 and optimization problem (4.17). For the interval matrix norm method, the γ computed at Line 11 is the local optimal exponential rate only for the center matrix of the interval matrix; we add an error to this γ to upper bound the exponential rate for the entire interval matrix using Lemma 4.8. Such an error term may introduce conservativeness, but this relaxation decreases the computational cost (see Section 4.3.6).

Remark 4.3. *It is straightforward to modify Algorithm 4 to accept validated simulations and the error bounds introduced. At Line 4 and Line 16, instead of bloating $Rec(t_{i-1}, t_i)$, we need to bloat $\texttt{hull}(\{R_{i-1}, R_i\})$, which is guaranteed to contain the solution $\xi(x,t), \forall t \in [t_{i-1}, t_i]$. Also, at Line 12 and Line 15, when using the ellipsoid $E_{M_i,c_i}(\xi(x,t_i))$, we use $E_{M_i,c_i}(0) \oplus D_i$.*

4.3.5 Accuracy of algorithm `LDFM`

Theorem 4.3 ensures that Algorithm `LDFM` over-approximates the reachable sets from initial set $B_\delta(x)$ for time $[0, T]$. In this section, we give results that formalize the accuracy of Algorithm `LDFM`. In the following, we assume that

$$\mathcal{R} = (R_1, t_1), \ldots, (R_k, t_k = T)$$

is a $(B_\delta(x), T)$-Reachtube returned by Algorithm `LDFM`.

The first Proposition 4.1 establishes that the bloating factor δ_i in Line 15 for constructing reachtubes goes to 0 as the size of the initial set $B_\delta(x)$ goes to zero. This implies that the over-approximation error from bloating can be made arbitrarily small by making the uncertainty in the initial cover $\langle x, \delta, \epsilon \rangle$ small.

Proposition 4.1. *In Algorithm `LDFM`, for any i, if M_0 and c_0 are optimal, in the sense that no M', c' exists such that $c' < c_0$ and $B_\delta(x) \subseteq E_{M',c'}(x)$, then as $dia(B_\delta(x)) \to 0$ the size of the bloating factor $\delta_i \to 0$ (Line 15).*

Proof. At Line 14, the algorithm updates c_i with some bounded number $c_{tmp}e^{\gamma_i \Delta t}$, and c_{tmp} is either inherited from c_{i-1} (Line 9) or computed by discrepancy function updating (Line 12) of M_{i-1}, c_{i-1}. In either case c_i goes to 0 as c_{i-1} goes to 0. In the discrepancy function updating case (Line 12) this is because we select the smallest ellipsoid $E_{M_i,c_{tmp}}(\xi(x,t_{i-1}))$ that contains

$E_{M_{i-1},c_{i-1}}(\xi(x,t_{i-1}))$, where if $c_{i-1} \to 0$, then $c_{tmp} \to 0$, and thus $c_i \to 0$. If $dia(B_\delta(x)) \to 0$, we will have $c_0 \to 0$ since M_0 and c_0 are optimal, and consequently $c_i \to 0$, for each $i = 1, \ldots, k$. From Line 15, it follows that $\delta_i = 2\sqrt{\lambda_{max}(c_i M_i^{-1})}$ (see [42] page 103), and therefore, as $c_i \to 0$, $\delta_i \to 0$, for each $i = 1, \ldots, k$. □

The contractive system's Jacobian matrix has negative matrix measure under certain coordinate transformation. Next, Corollary 4.2 establishes that for contractive systems the reachtube computed by Algorithm `LDFM` converges to the rectangles that represent the simulation.

Corollary 4.2. *Consider a contractive system for which there exists a matrix M such at $\forall x \in \mathbb{R}^n, J_f(x)^T M + M J_f(x) \preceq \gamma M$, and $\gamma < 0$. Compute the reachtube of the system using Algorithm* `LDFM`*, we have as $k, T \to \infty$,*

$$|dia(R_k) - dia\,(Rec(t_{k-1}, T))\,| \to 0.$$

Proof. From the contractive condition, we have a uniform matrix M such that any evaluation of the Jacobian matrix satisfies $J_f(x)^T M + M J_f(x) \leq \gamma M$. The "If" condition at Line 6 will always hold for $M_i = M$ and $\gamma_i = \gamma$, and at Line that $c_i = c_{i-1} e^{\gamma \Delta t}$. Inductively, we obtain $c_k = c_0 e^{\gamma t_k}$ and $\gamma < 0$. So $c_k \to 0$ as $t_k = T \to \infty$. The bloating factor δ_k, which is the diameter of $E_{M_k,c_k}(\xi(x,t_k))$, also goes to 0. From the definition of R_k, we have $R_k \supseteq Rec(t_{k-1}, T)$. The bloating factor for $Rec(t_{k-1}, T)$ goes 0, so $R_k \to Rec(t_{k-1}, T)$, and the result follows. □

Corollary 4.3. *Consider a linear system $\dot{x} = Ax$ with a Hurwitz matrix A. Compute the reachtube of the system using Algorithm* `LDFM`*, we have as $k, T \to \infty$,*

$$|dia(R_k) - dia\,(Rec(t_{k-1}, T))\,| \to 0.$$

A linear system is contractive if A is Hurwitz as the real part of its eigenvalues are bounded by some constant $\gamma < 0$. Pick matrix P such that PAP^{-1} is the Jordan form of A, then there exists some $\epsilon < 0$ such that $(P^{-1})^T A^T P^T + PAP^{-1} \preceq \epsilon I$. Pre- and post-multiplying by P^T and P, we get: $A^T P^T P + P^T P A \preceq \epsilon P^T P$. Setting $M = P^T P$ we see that the contractive condition is satisfied.

For (even unstable) linear invariant systems, since the Jacobian matrix A does not change over time, the discrepancy function can be computed

globally for any time t and $x_1, x_2 \in \mathbb{R}^n$. Therefore, there is no wrap-over (accumulated) error introduced using Algorithm `LDFM`. We have also proved the convergence of the algorithm for contractive nonlinear systems. For non-contractive nonlinear systems, the over-approximation error might be accumulated. Such wrap-over error introduced by on-the-fly algorithms may not be avoidable. Therefore, for non-contractive or unstable nonlinear systems, it is especially important to reduce the over-approximation error in each time interval, which is what Algorithm `LDFM` aims to achieve.

4.3.6 Computational considerations of algorithm `LDFM`

We discuss the computational aspects of Algorithm`LDFM`. For an n-dimensional system model, assume that there are $n_{\mathcal{I}}$ entries of the Jacobian matrix that are not a constant number. At any iteration, at Line 5, the algorithm solves $2n_{\mathcal{I}}$ optimization problems or uses interval arithmetic to get lower and upper bounds of each component of the Jacobian. For linear time invariant systems, this step is eliminated. At Line 6 the vertex matrix constraints method will compute $2^{n_{\mathcal{I}}}$ matrix inequalities; however, the interval matrix norm method will compute 1 matrix inequality. At Line 8 or Line 11, the vertex matrix constraints method will solve 1 convex optimization problem with $2^{n_{\mathcal{I}}} + 1$ constraints, but the matrix interval method solves 1 convex optimization problem with 2 constraints. The discrepancy function updating at Line 12 solves 1 SDP problem. The rest of the algorithm from Line 14 to Line 16 consists of algebraic operations.

From the above analysis, we can conclude that the interval matrix norm method improves the efficiency of the algorithm as compared to the vertex matrix constraints method, especially when the number of non-constant terms in the Jacobian matrix is large; however, the interval matrix norm method introduces the error term $\delta/\lambda_{\min}(M_i)$ at each iteration, resulting in a more conservative result. We can consider the vertex matrix constraints method accurate but with a greater computational burden, and the interval matrix method simple but coarse.

The effective efficiency of the algorithm depends on whether the system is contractive or not. For contractive systems, it is possible that the "if" condition often holds at Line 6, allowing the algorithm to often reuse the

previous norm and contraction rate. For non-contractive systems this may not be the case. Also, the efficiency of the algorithm applied to linear systems is low, since the interval matrix to which the Jacobian matrix belongs is time invariant.

As a comparison, Algorithm `LDF2` does not need to solve any optimization problem, except that at Line 6 we need to solve $2n_{\mathcal{I}}$ optimization problems or use interval arithmetic to get lower and upper bounds of each component of the error matrix in Algorithm 2. The remaining lines are all algebraic operations. However, Algorithm `LDF2` is a special case of Algorithm `LDFM` since it fixes $M = I$ for all the time intervals instead of trying to compute the optimal M_i to achieve the best exponential converging/diverging rate of the trajectories.

Chapter 5

EXPERIMENTAL EVALUATION

We evaluate the performance of the methods proposed in this thesis. We first use a small linear example to show the level of conservativeness that Algorithm `LDFM` adds to the exact reach set. Then we use a series of benchmark examples with different complexities to compare the efficiency and accuracy of our proposed algorithms with the reach set computation tool Flow*.

5.1 Accuracy of Algorithm `LDFM`

In this section, we illustrate the accuracy of Algorithm `LDFM` using an example for which the exact reach set (at sample time) is known. We use the 2-dimensional linear system with nilponent term model:

$$\dot{x} = (-\epsilon I + N)x, \tag{5.1}$$

where $\epsilon = 1/10$, I is the identity matrix and $N = (0, 1; 0, 0)$.

We choose two results to be plotted in Figure 5.1 and Figure 5.2, where the initial set Θ in both cases is a ball with radius 0.2 centered at $[1, 1]^T$ and $[0, 0]^T$ respectively. The sampled reachtubes computed using Algorithm `LDFM` are shown in red ellipsoids and the sampled exact reach sets are shown in green ellipsoids. To compute the exact ellipsoid at 1 second snapshots, we use the forward image of an ellipsoid in discrete-time according to the linear transformation defined by the above linear system (page 99 of [42]).

Table 5.1 also shows the degree of conservativeness added by the proposed method. We compare the volume of the reach set as computed from [42] and by `LDFM`, both normalized by the volume of the initial set. We choose the average volume of the reach set, which is the sum of the volumes over the sampling time divided by the number of samples, and the final volume of the reach set. Because we are using a linear system, the results should

be independent from the initial set since the converging or diverging rate between trajectories for linear systems remains unchanged for the entire state space. The discrepancy function computed for system (5.1) is

$$\beta_M(\|x_1 - x_2\|, t) = \|x_1 - x_2\|_M e^{-0.1t},$$

where $M = \begin{bmatrix} 1.2106 & -1.5138 \\ -1.5138 & 136.1004 \end{bmatrix}$.

From the numerical results from Table 5.1 we can see that the volume of the exact reach set sampled per 1 second is around 5% of the over-approximation reach set computed using Algorithm `LDFM` at the same sample time. The conservativeness added at the final time increases with the increase of the time horizon.

Table 5.1: Conservativeness of Algorithm `LDFM` on a 2-dimensional linear system. Initial set: a circle centered at $[1, 1]$ with radius 0.2. **TH**: Time Horizon. **A/I VR**: the ratio of the average volume of the over-approximation reach set over sampled time points to the initial set volume. **F/I VR**: the ratio of the volume of the over-approximation reach set at the final time T to the initial set volume. **Exact/ Algorithm `LDFM` ratio**: the ratio of the normalized exact reach set with the normalized reach set over-approximation computed by Algorithm `LDFM`.

TH(s)	Algorithm `LDFM`		Exact Ellipsoid		Exact/ Algorithm `LDFM` ratio	
	A/I VR	F/I VR	A/I VR	F/I VR	A/I VR	F/I VR
10	170.15	98.22	11.15	3.38	6.55%	3.44%
20	117.23	36.13	6.47	0.46	5.52%	1.27%
30	86.42	13.29	4.44	0.06	5.14%	0.47%
40	67.30	4.89	3.36	8.40e-3	5.00%	0.17%
50	54.67	1.80	2.70	1.12e-3	4.95%	0.06%

5.2 Comparison of the Algorithms with Flow*

We implemented a prototype tool in MATLAB based on Algorithm `LDF2` and Algorithm `LDFM` and tested it on several benchmark verification problems. Simulations are generated using the validated simulation engine CAPD [28], which returns a sequence of time-stamped rectangles as required by our algorithm. The optimization problems (4.11), (4.17), and the SDP problems are solved using SDP3 [43] and Yalmip [44].

Table 5.2: **Dim**: Dimension of the system. **ID**: Initial Diameter. **TH**: Time Horizon. **RT**: Running Time (includes simulation time in CAPD). **A/I VR**: Average/Initial Volume Ratio, the ratio of the average volume of the over-approximation reach set at sampled time points with the initial set volume. **F/I VR**: Final/Initial Volume Ratio, the ratio of the volume of the over-approximation reach set at the final time T with the initial set volume. The experiments were conducted on an Intel Xeon V2 desktop computer.

	System	Dim	ID	TH(s)	Algorithm `LDFM`			Flow*			Algorithm `LDF2`		
					RT(s)	A/I VR	F/I VR	RT(s)	A/I VR	F/I VR	RT(s)	A/I VR	F/I VR
1	Van der Pol	2	0.20	10	0.69	0.28	4.60e-4	1.66	0.26	1.61e-4	0.23	0.44	2.30e-3
2	Moore-Greitzer	2	0.20	10	4.76	0.25	3.79e-4	4.67	0.34	2.16e-3	0.35	1.07	6.50e-3
3	Brusselator	2	0.20	10	5.39	0.69	3.78e-2	6.66	0.33	2.41e-2	0.32	1.87	0.39
4	Diode Oscillator	2	0.01	9	39.98	0.65	9.08e-6	214.40	1.22	0.65	1.39	67.78	106.35
5	Robot Arm	4	0.01	10	9.26	1.83	0.31	354.80	3.35	2.97	1.88	70.22	29.02
6	Powertrain	4	4e-4	5	3.97	10.14	0.77	267.00	3457	1886	1.88	467.98	192.45
7	Saturation	6	0.04	10	1.99	2.19	1.83	5245	1.72	1.16	0.55	4.97e9	8.13e9
8	Laub-Loomis	7	0.05	10	5.97	11.11	6.44	355.10	3.72	0.88	1.78	1.61e43	6.64e43
9	Biology Model	7	5e-3	2	9.25	171.30	344.10	603.60	68.03	343.40	4.51	1.30e9	2.31e9
10	AS Polynomial	12	5e-3	2	3.63	45.79	45.74	5525	3.06e10	2.87e10	5.02	2.11e5	1.39e5
11	Helicopter (L)	28	0.10	30	2.96	0.75	0.55	288.20	4.24e49	6.02e39	0.17	2.46e63	1.27e63

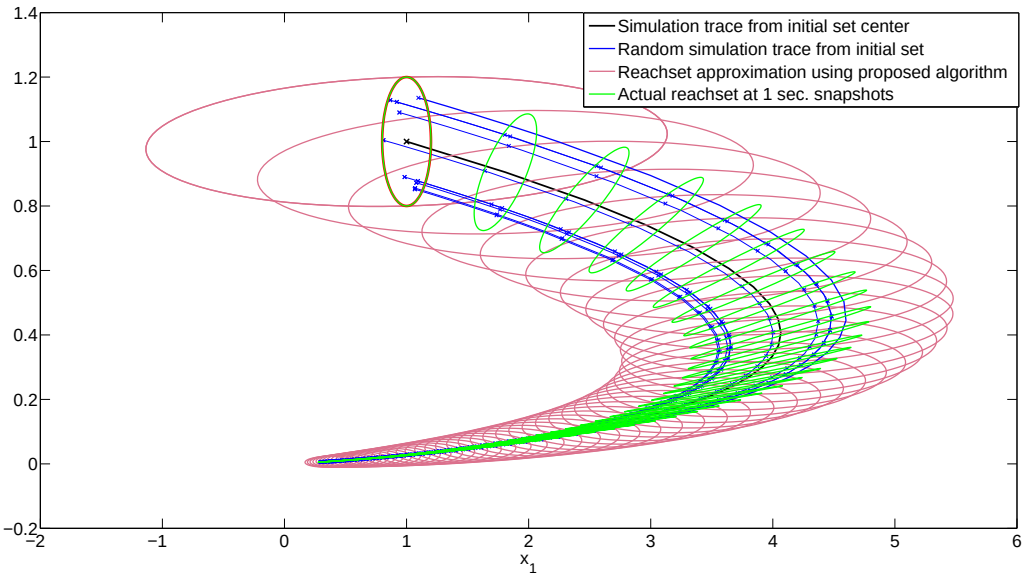

Figure 5.1: Reach set over-approximation using Algorithm `LDFM` (red ellipsoid) and exact reach set (green ellipsoid) of linear system with nilpotent term. Initial set: a circle centered at $[1, 1]^T$ with radius 0.2.

We evaluated the algorithm on several nonlinear benchmark problems. Van der Pol, Moore-Greitzer and Brusselator are standard low-dimensional examples. The diode oscillator from [21] is low dimensional but has complex dynamics described by degree 5 polynomials. Robot Arm is a four-dimensional model from [45]. Powertrain is the powertrain control system proposed in [46] as part of a verification challenge problem [6]. The Powertrain system is highly nonlinear; the dynamic equations contain polynomials, rational functions, and square roots. Saturation is a system analyzed in [47] that exhibits saturation behavior. Laub-Loomis is a molecular network that produces spontaneous oscillations, and is used as a case study for NLTOOLBOX [48]. AS Polynomial is a twelve-dimensional polynomial system [49] that is asymptotically stable around the origin. We also study one 28-dimensional linear model of a helicopter [3].[1] For systems with fewer than three dimensions, we use the vertex matrix constrains method, and for systems with more than three dimensions, we use the interval matrix norm method.

As mentioned earlier, Algorithm `LDF2` is a special case of Algorithm `LDFM`;

[1]For the initial condition set of the helicopter model, we used 0.1 as the diameter for the first eight dimensions and 0.001 for the remaining ones, because the reach set estimations of Flow* became unbounded when using 0.1 as the diameter for all dimensions.

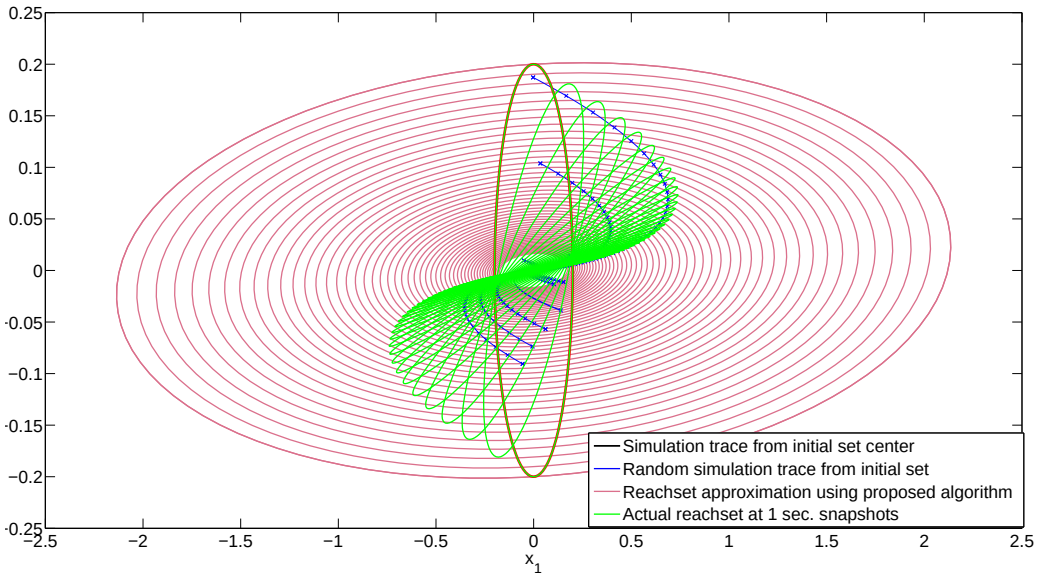

Figure 5.2: Reach set over-approximation using Algorithm `LDFM` (red ellipsoid) and exact reach set (green ellipsoid) of linear system with nilpotent term. Initial set: a circle centered at $[0, 0]^T$ with radius 0.2.

therefore, the reachtubes produced by Algorithm `LDFM` are always less conservative than those produced by Algorithm `LDF2`. For example, the reachtubes computed by Algorithm `LDF2` for Saturation and Laub-Loomis expand to fill the entire user-defined search space, but for the same time horizon the current algorithm proves safety. In this experiment, we incorporate the simple coordination transformation method introduced in [17] to reduce the conservativeness of Algorithm `LDF2`.

We compare the running time and accuracy of Algorithm `LDFM` against a leading nonlinear verification tool, Flow* [1], and also against the Algorithm `LDF2`. Analyses of several of these examples have been reported on Flow*'s website and in those cases we use the given configurations. In other cases, we set the order of Taylor models to be adaptive and we try different remainder values in an attempt to get the best result.

As a measure of precision, we compare the ratio of the reach set volume to the initial set volume. This is a reasonable measure of accuracy because the tools use different set representations (Flow* uses hypercubes[2] and Algorithm `LDFM` and the Algorithm `LDF2` method use ellipsoids). We calculate two volume ratios: (a) average volume of the reach set divided by the initial

[2]Flow* supports other shapes but we chose hypercube to simplify computation.

volume (sampled at the time steps used in Flow*), (b) the reach set at the final time point T divided by the initial volume.

The results are shown in Table 5.2. Consider the performance of Algorithm `LDFM` as compared to Flow*. From Line 1-3 in Table 5.2, we see that for simple low dimensional nonlinear systems, the performance of Flow* is comparable to our algorithm. Lines 4-5 and 7-9 show that for more complicated nonlinear systems (with higher order polynomials or higher order dimensionality), our Algorithm 4 performs much better in terms of running time without sacrificing accuracy. Moreover, from Line 6 and Lines 10-11, Algorithm `LDFM` not only finishes reachtube computation much faster, but also provide less conservative results for even more complicated systems (with complicated nonlinear dynamic or even higher dimensions). For linear systems, Algorithm `LDFM` can provide one global discrepancy function that is valid for the entire space to do reach set over-approximation, as compared to Flow*, where even for linear systems, the complexity for each time interval is exponential in both the dimensionality and the order of the Taylor models. Algorithm `LDFM` is more efficient because it is based on the Jacobian, which has n dimensions, so the complexity of Algorithm `LDFM` using interval matrix norm method increases polynomially with the dimension, if the interval matrix norm method is used.

Consider next the performance of Algorithm `LDFM` as compared to Algorithm `LDF2`. Algorithm `LDF2` requires slightly less computation time in all but one case; however, as expected, Algorithm `LDF2` is more conservative in every case and in some cases is many orders of magnitude more conservative. The result confirms that the complexity of Algorithm `LDFM` is higher than that of Algorithm `LDF2` as discussed in Section 4.3.6, while Algorithm `LDFM` is more accurate because it considers more general cases.

To summarize, Algorithm `LDF2` computes reachtubes within relatively short time but with possibly larger approximation error. Algorithm `LDF2` produces more accurate reachtubes at the cost of more time consuming. Results produced by Algorithm `LDF2` compare favorably with the verification tool Flow* on the examples with higher dimensions or with complex dynamics.

Chapter 6

Conclusion

We discussed several techniques to compute over-approximation of the reachable states from simulation traces, which could be used as the core function of the simulation-driven verification approaches. The techniques are based on the new methods to compute the upper bounds on the matrix measures of the interval matrices, where the interval matrices contain the behaviors of the Jacobian matrix of the nonlinear system over a compact subset of the state space. We used the upper bounds on matrix measures as the exponential change rate of the discrepancy functions, which are used to bloat the simulation traces to get reachtubes.

We provided two different version of the algorithms which use different matrix measures. The first algorithm is based on the 2-norm matrix measure and provides a relatively coarse reachtube but the computation is comparatively fast. The second class of algorithms instead computes the local optimal coordinate transformation such that the local exponential change rate of the discrepancy is minimized, which leads to reachtubes that are less conservative, but take more time.

We evaluated the accuracy of the local optimal algorithm by computing the level of conservativeness it adds to the exact reach set. We also demonstrated the effectiveness of our proposed algorithms by comparing the performance of the prototype implementations with the Flow* tool. Results show that our approaches compare favorably with the verification tool Flow* on the examples with higher dimensions or with complex dynamics.

Future work will include implementing the proposed algorithms in verification tools like C2E2 for performing bounded time verification of hybrid systems. We will also extend the methods to handle the variations of parameters as well as the variations of initial states.

References

[1] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *CAV*. Springer, 2013, pp. 258–263.

[2] A. Donzé and O. Maler, "Systematic simulation using sensitivity analysis," in *HSCC*. Springer, 2007, pp. 174–189.

[3] G. Frehse, C. Le 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*, ser. LNCS, S. Q. Ganesh Gopalakrishnan, Ed. Springer, 2011.

[4] S. Kong, S. Gao, W. Chen, and E. Clarke, "dReach: δ-reachability analysis for hybrid systems," in *TACAS*. Springer, 2015, pp. 200–205.

[5] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: A verification tool for stateflow models," in *TACAS*. Springer, 2015, pp. 68–82.

[6] P. S. Duggirala, C. Fan, S. Mitra, and M. Viswanathan, "Meeting a powertrain verification challenge," in *Computer Aided Verification*. Springer, 2015, pp. 536–543.

[7] N. Aréchiga, J. Kapinski, J. V. Deshmukh, A. Platzer, and B. Krogh, "Numerically-aided deductive safety proof for a powertrain control system," *Electronic Notes in Theoretical Computer Science*, vol. 317, pp. 19–25, 2015.

[8] Z. Huang, C. Fan, A. Mereacre, S. Mitra, and M. Z. Kwiatkowska, "Invariant verification of nonlinear hybrid automata networks of cardiac cells," in *CAV*. Springer, 2014, pp. 373–390.

[9] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam, "Modeling and verification of a dual chamber implantable pacemaker," in *TACAS*. Springer, 2012, pp. 188–203.

[10] M. Bozzano, A. Cimatti, A. F. Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, and S. Tonetta, "Formal design and safety analysis of air6110 wheel brake system," in *International Conference on Computer Aided Verification*. Springer, 2015, pp. 518–535.

[11] A. El-Guindy, D. Han, and M. Althoff, "Formal analysis of drum-boiler units to maximize the load-following capabilities of power plants," *IEEE Transactions on Power Systems*, vol. PP, no. 99, pp. 1–12, 2016.

[12] N. E. Beckman, A. V. Nori, S. K. Rajamani, R. J. Simmons, S. D. Tetali, and A. V. Thakur, "Proofs from tests," *IEEE Transactions on Software Engineering*, vol. 36, no. 4, pp. 495–508, 2010.

[13] Y. Deng, A. Rajhans, and A. A. Julius, "Strong: A trajectory-based verification toolbox for hybrid systems," in *Quantitative Evaluation of Systems*. Springer, 2013, pp. 165–168.

[14] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 167–170.

[15] A. Gupta, R. Majumdar, and A. Rybalchenko, "From tests to proofs," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2009, pp. 262–276.

[16] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas, "Robust test generation and coverage for hybrid systems," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2007, pp. 329–342.

[17] C. Fan and S. Mitra, "Bounded verification with on-the-fly discrepancy computation," in *ATVA*. Springer, 2015.

[18] P. S. Duggirala, S. Mitra, and M. Viswanathan, "Verification of annotated models from executions," in *EMSOFT*. IEEE Press, 2013, p. 26.

[19] D. Angeli, "A Lyapunov approach to incremental stability properties," *Automatic Control, IEEE Transactions on*, vol. 47, no. 3, pp. 410–421, 2002.

[20] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for nonlinear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[21] J. Maidens and M. Arcak, "Reachability analysis of nonlinear systems using matrix measures," *Automatic Control, IEEE Transactions on*, vol. 60, no. 1, pp. 265–270, 2015.

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

[23] A. A. Julius and G. J. Pappas, "Trajectory based verification using local finite-time invariance," in *HSCC*. Springer, 2009, pp. 223–236.

[24] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *Automatic Control, IEEE Transactions on*, vol. 57, no. 7, pp. 1804–1809, 2012.

[25] H. K. Khalil and J. Grizzle, *Nonlinear Systems. Vol. 3.* New Jersey: Prentice Hall, 1996.

[26] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[27] N. Nedialkov, "VNODE-LP: Validated solutions for initial value problem for ODEs," McMaster University, Tech. Rep., 2006.

[28] CAPD, "Computer assisted proofs in dynamics," 2002. [Online]. Available: http://www.capd.ii.uj.edu.pl/

[29] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*. ACM, 1995, pp. 373–382.

[30] E. Hainry, "Reachability in linear dynamical systems," in *Logic and Theory of Algorithms*. Springer, 2008, pp. 241–250.

[31] 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, 2000.

[32] K. Makino and M. Berz, "Taylor models and other validated functional inclusion methods," *International Journal of Pure and Applied Mathematics*, vol. 4, no. 4, pp. 379–456, 2003.

[33] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*. SIAM, 2009.

[34] R. Farhadsefat, J. Rohn, and T. Lotfi, "Norms of interval matrices," Institute of Computer Science, Academy of Sciences of the Czech Republic, Tech. Rep. No. V-1122, August 2011.

[35] C. A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*. SIAM, 1975, vol. 55.

[36] E. D. Sontag, "Contractive systems with inputs," in *Perspectives in Mathematical System Theory, Control, and Signal Processing*. Springer, 2010, pp. 217–228.

[37] P. S. Duggirala, L. Wang, S. Mitra, M. Viswanathan, and C. Muñoz, "Temporal precedence checking for switched models and its application to a parallel landing protocol," in *Formal Methods*. Springer, 2014, pp. 215–229.

[38] C. Fan, J. Kapinski, X. Jin, and S. Mitra, "Locally optimal reach set over-approximation for nonlinear systems," Coordinated Science Laboratory technical report no. UILU-ENG-16-2202, University of Illinois at Urbana-Champaign, Tech. Rep., 2016, http://hdl.handle.net/2142/90424.

[39] G. H. Golub and C. F. Van Loan, *Matrix Computations*. JHU Press, 2012, vol. 3.

[40] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, ser. Studies in Applied Mathematics. Philadelphia, PA: SIAM, 1994, vol. 15.

[41] D. Liberzon, *Switching in Systems and Control*. Springer Science & Business Media, 2012.

[42] A. Kurzhanskii and I. Vályi, *Ellipsoidal Calculus for Estimation and Control*. Nelson Thornes, 1997.

[43] R. H. Tütüncü, K. C. Toh, and M. J. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3," *Mathematical Programming*, vol. 95, no. 2, pp. 189–217, 2003.

[44] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *CACSD*, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Main.HomePage

[45] D. Angeli, E. D. Sontag, and Y. Wang, "A characterization of integral input-to-state stability," *Automatic Control, IEEE Transactions on*, vol. 45, no. 6, pp. 1082–1097, 2000.

[46] J. Kapinski, J. V. Deshmukh, S. Sankaranarayanan, and N. Aréchiga, "Simulation-guided Lyapunov analysis for hybrid dynamical systems," in *HSCC*. ACM, 2014, pp. 133–142.

[47] A. Papachristodoulou and S. Prajna, "Analysis of non-polynomial systems using the sum of squares decomposition," in *Positive Polynomials in Control*. Springer, 2005, pp. 23–43.

[48] R. Testylier and T. Dang, "NLTOOLBOX: A library for reachability computation of nonlinear dynamical systems," in *ATVA*. Springer, 2013, pp. 469–473.

[49] J. Anderson and A. Papachristodoulou, "Dynamical system decomposition for efficient, sparse analysis," in *CDC*. IEEE, 2010, pp. 6565–6570.