Simulation-Driven Reachability Using Matrix Measures

CHUCHU FAN, University of Illinois at Urbana-Champaign
JAMES KAPINSKI and XIAOQING JIN, Toyota Motor North America R&D
SAYAN MITRA, University of Illinois at Urbana-Champaign

Simulation-driven verification can provide formal safety guarantees for otherwise intractable nonlinear and hybrid system models. A key step in simulation-driven algorithms is to compute the reach set overapproximations from a set of initial states through numerical simulations and sensitivity analysis. This article addresses this problem by providing algorithms for computing *discrepancy functions* as the upper bound on the sensitivity, that is, the rate at which trajectories starting from neighboring states converge or diverge. The algorithms rely on computing local bounds on matrix measures as the exponential change rate of the discrepancy function. We present two 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, and therefore, the conservativeness of the overapproximation, is locally minimized. The proposed algorithms 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 functions give soundness and relative completeness of the overall simulation-driven safety-bounded verification algorithm. We present a series of experiments to illustrate the accuracy and performance of the algorithms.

CCS Concepts: • **Computer systems organization → Embedded software**;

Additional Key Words and Phrases: Nonlinear System, Reachability, Matrix measures, Discrepancy function, Embedded System

ACM Reference format:
Chuchu Fan, James Kapinski, Xiaoqing Jin, and Sayan Mitra. 2017. Simulation-Driven Reachability Using Matrix Measures. *ACM Trans. Embed. Comput. Syst.* 17, 1, Article 21 (December 2017), 28 pages.
https://doi.org/10.1145/3126685

1 INTRODUCTION

Cyber-physical systems (CPSs), which tightly couple physical processes with software, networks, and sensing, have become ubiquitous; however, reliability and security lapses of such systems can disrupt communities and lead to catastrophic failures. Recent advances in formal verification techniques [10, 14, 17, 23, 33] show promise in aiding the design and verification of realistic CPS, such as automotive systems [4, 15], medical devices [27, 29], aerospace systems [8], and energy systems [19].

This work is supported by National Science Foundations research grants NSF CAREER 1054247 and NSF CSR 1422798.
Authors' addresses: C. Fan, Coordinated Science Laboratory, Room 247. 1308 W. Main Street Urbana, IL 61801, USA; email: cfan10@illinois.edu; J. Kapinski, 1555 Woodridge Ave. Ann Arbor, MI 48105, USA; email: jim.kapinski@toyota.com; X. Jin, 8797 Barnwood Lane Riverside, CA 92508, USA; email: sunqxj@gmail.com; S. Mitra, Coordinated Science Laboratory, Room 266. 1308 W. Main Street Urbana, IL 61801, USA; email: mitras@illinois.edu.

https://doi.org/10.1145/3126685

There are two predominant approaches for verifying CPS models: dynamic analysis generates traces from models or from executing the actual systems to find possible defects. The approach can be fast but suffers from incompleteness; that is, it is impossible to cover all possible system behaviors. In contrast, static approaches analyze the system model to infer properties covering all possible behaviors. This is done either by computing (or approximating) the reachable states or by deducing properties using proof rules. Unfortunately, this approach typically does not scale. Reachability computations suffer from the state-space explosion problem and the deductive methods require significant human intervention to discharge the necessary proof obligations.

Recently, a third verification approach has gained traction [5, 11, 13, 17, 26, 30]. It uses simulation traces to verify invariant properties by appropriately generalizing an individual simulation (or test) trace to a set of behaviors and then verifies the property on this generalized set. The success of this approach hinges on good generalizations that can lead to coverage of all possible behaviors from only finitely many simulations. For CPS, this generalization can be computed by exploiting the smoothness of the continuous dynamics [11, 13, 17]. The scalability of this approach has been demonstrated by verifying several challenging benchmarks [15, 18, 27].

Consider a dynamical system described by the differential equation $\dot{x} = f(x)$. The initial set $\Theta \subset \mathbb{R}^n$ is a compact set in $\mathbb{R}^n$. A trajectory or a solution of this system is a function that gives the state of the system at time t starting from a given initial state $x_0 \in \Theta$. The reach set is the set of states that are reachable from an initial set. Our approach follows the simulation-driven verification approach of [16, 17]. It overapproximates the reach set using only finitely many (possibly error-prone) numerical simulations and by bloating each individual simulation by an appropriate factor derived from the sensitivity of the trajectories to the initial conditions. This notion of sensitivity is formalized as *discrepancy functions*; the general properties needed for soundness and completeness of verification based on discrepancy functions were identified in [16], though the key step of computing discrepancy functions automatically for general nonlinear systems remained an open problem.

As noted in [16, 38], several techniques (contraction metric [37], incremental stability [2], matrix measure [6], etc.) can be used to find discrepancy functions; however, those techniques either restrict the class of nonlinear systems (e.g., polynomial systems, as in [31]) or require nontrivial user inputs (e.g., the closed-form expression of a matrix measure function, as in [38]). In this work, we address this problem by providing algorithms that compute discrepancy functions automatically for general nonlinear dynamical systems. The proposed algorithm can provide locally optimal reach set overapproximations.

Our approach for computing discrepancy is based on the well-known result that an upper bound 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 [6, 41]. Closed-form expressions for matrix measures are in general difficult to obtain for nonlinear systems. For example, for matrix $J_f(x)$, the matrix measure under Euclidean norm is the largest eigenvalue of the symmetric part of the matrix $\lambda_{\max}((J_f(x) + J_f^T(x))/2)$. However, if we can overapproximate all possible values of the system's Jacobian matrix $J_f(x)$ over some compact set $S \subset \mathbb{R}^n$, we can obtain an upper bound on the matrix measure of the Jacobian matrix over S without knowing its closed form. This two-step computation proceeds as follows: (1) it uses interval matrices to bound the variation of the Jacobian matrix over S, and (2) it computes the upper bound of the matrix measure of the interval matrix. In this work, we will introduce two algorithms `LDF2` (using 2-norm) and `LDFM` (using coordinate transformed 2-norm under transformation M) to compute the upper bound of the matrix measure of the interval matrix.

The two algorithms use two different matrix measures. `LDF2` computes the matrix measure under the 2-norm, then applies a matrix perturbation theorem to transform the problem of bounding the largest eigenvalue to bounding the 2-norm of a matrix valued function [21]. This algorithm

does not require solving optimization problems numerically. The second algorithm LDFM generates a more accurate discrepancy function by computing the local optimal bound of the matrix measure under a linear coordinate transformation. The idea is to search all possible linear coordinate transformations to minimize the matrix measures, which involves solving several optimization problems using semidefinite programming [20]. LDFM achieves greater accuracy at the expense of higher computational effort, as compared to LDF2. We provide two techniques for computing the optimal bound on the matrix measure of the interval matrix for the LDFM method. The first method uses the vertex matrices of the interval matrix, and the second uses the norm of the interval matrix. Both approaches are less conservative than the LDF2 algorithm as they find locally optimal exponential rates for the discrepancy function. We also include some comparison and discussion of the tradeoff between accuracy and speed for these two techniques.

For dynamical systems with inputs $\dot{x} = f(x, u)$, the simulation traces are decided by both the initial states and inputs u. We extend the methods to compute *input-to-state discrepancy functions*, which can be used to construct reach sets for systems with bounded nondeterministic input signals. Experimental results show that due to an error of $e^{\frac{1}{2}t}$ that is added to such input-to-state discrepancy function, the computed reach set overapproximations for systems with inputs will have dramatically increasing error.

In summary, the contributions of this article are as follows: (1) We provide two algorithms, namely, LDF2 and LDFM, for overapproximating reach sets for nonlinear models using local discrepancy functions. Although in this work we concentrate on the reachability analysis of nonlinear dynamical systems, with appropriate handling of guards and transitions as shown in [17], these algorithms can be used in hybrid verification tools like C2E2 [17] and Breach [14]. (2) We show that the proposed algorithms for computing discrepancy functions preserve the soundness and the relative completeness of overall verification. (3) We establish that algorithm LDFM returns a locally optimal exponential rate for estimating the sensitivity of the system. To our knowledge, this is the first result that gives a guarantee about the quality of the overapproximations, which in turn implies the efficiency of the verification algorithm. (4) We conduct a sequence of experiments on the prototype implementations of the algorithms, including the comparison with Flow* [10] on a suite of linear and nonlinear system examples; the results suggest that the proposed methods provide significant advantages for verifying models with large dimensions (greater than four) and with complicated differential equations.

1.1 Related Work

Several approaches have been proposed to obtain proofs about (bounded time) invariant or safety properties from simulations [14, 16, 21, 24]. Sensitivity analysis is a technique to systematically simulate arbitrary nonlinear systems with inputs [14]. 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. In [31], the authors provided sound simulation-driven methods to overapproximate the distance between trajectories, but these methods are mainly limited to affine and polynomial systems. For general nonlinear models, this approach may not be sound, as higher-order error terms are ignored when computing this upper bound.

The idea of computing the reach sets from trajectories is similar to the notions of incremental Lyapunov function [2]. In this work, we do not require systems to be incrementally stable. Similar ideas have also been considered for control synthesis in [44]. Other approaches for reach set estimation for nonlinear systems operate directly on the vector field involving higher-order Taylor expansions [10, 33]. However, this method suffers from complexity that increases exponentially with both the dimension of the system and the order of the model.

The work closest to ours involves reachability analysis using matrix measures [38], where the authors use the fact that the matrix measure of the Jacobian matrix can bound the distance between neighboring trajectories [6, 41]. The technique relies on user-provided closed-form matrix measure functions, which are in general difficult to compute. In contrast, our approach automatically computes the bounds on matrix measures under 2-norm or coordinated transformed 2-norm.

Techniques such as ours that perform reach set overapproximations for continuous systems are crucial building blocks for hybrid system verification. For example, our approach can be used by the C2E2 tool to perform verification, as in [17]. Future work will evaluate the performance of our technique for hybrid examples; here we focus on continuous nonlinear dynamical systems.

The rest of the article is organized as follows. Section 2 sets up the mathematical preliminaries. Section 3 gives an overview of the simulation-driven verification approach as developed in [16]. Section 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 Section 4 can be used directly as the core function in the verification algorithm in Section 3. Section 5 illustrates the accuracy and the performance of the proposed algorithms, followed by the conclusion in Section 6.

2 PRELIMINARIES

Throughout the article, we use mathematical font to denote sets and typewriter font to denote operations and algorithm names. We use $\mathbb{R}$ to denote the set of real numbers. $\mathbb{R}^n$ denotes the n-dimensional *Euclidean space*. For a vector $x \in \mathbb{R}^n$, $\|x\| = \sqrt{x^T x}$ will denote its 2-norm (Euclidean norm). Given a positive definite $n \times n$ matrix M, the *M-norm* of the vector x, $\|x\|_M = \sqrt{x^T M x}$ is the norm of x under the transformation M. For any $M > 0$, there exists a nonsingular matrix $C \in \mathbb{R}^{n \times n}$, such that $M = C^T C$. So, $\|x\|_M = \sqrt{x^T C^T C x} = \|Cx\|$. 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.

For sets $S_1, S_2 \subseteq \mathbb{R}^n$, $\texttt{hull}(S_1, S_2)$ is their convex hull. The hull of a set of $n \times n$ matrices is defined in the usual way, by considering each matrix as a vector in $\mathbb{R}^{n^2}$. The diameter of a compact set S is defined as $\texttt{Dia}(S) = \sup_{x,y \in S} \|x - y\|$. $E_{M,\delta}(x_0) = \{x \mid \|x - x_0\|_M \leq \delta\}$ represents an ellipsoid centered at $x_0 \in \mathbb{R}^n$, with *shape* M and *size* δ. The δ ball around x_0: $B_\delta(x) = \{x \mid ||x - x_0|| \leq \delta\}$ is a special case of $E_{M,\delta}(x_0)$, where M is the identity matrix I.

For any matrix $A \in \mathbb{R}^{n \times n}$, A^T is its transpose. $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$) are the maximum and minimum eigenvalues of A, respectively. A lowercase letter with subscript a_{ij} denotes the element in the i^{th} row and j^{th} column of a matrix A. $\|A\|_1, \|A\|_2, \|A\|_\infty, \|A\|_F$ denote respectively the 1,2, infinity, and Frobenius norms of A. Hereafter, we will drop the subscript for $\|A\|_2$ and write it as $\|A\|$. $|A|$ is the matrix obtained by taking the element-wise absolute value of matrix A.

2.1 Dynamical Systems

The continuous evolution of an n-dimensional *dynamical system* is given by the differential equation

$$\dot{x} = f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a locally Lipschitz and continuously differentiable function. A *solution* or a *trajectory* of the system is a function $\xi : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, such that for any state $x_0 \in \mathbb{R}^n$ and at any time $t \in \mathbb{R}_{\geq 0}$, $\xi(x_0, t)$ satisfies Equation (1). The existence and uniqueness of solutions are guaranteed by the Lipschitz continuity of f. 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, that is, $[J_f(x)]_{ij} = \frac{\partial f_i(x)}{\partial x_j}$.

The following lemma states a relationship between f and its Jacobian J_f, which can be proved using the generalized mean value theorem [21].

Lemma 1. *For any continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}^n$, and $x, r \in \mathbb{R}^n$,*

$$f(x+r) - f(x) = \left(\int_0^1 J_f(x+sr)ds \right) \cdot r, \tag{2}$$

where the integral is evaluated component-wise.

Simulations and Reachable States. Although it is generally difficult to obtain closed-form expressions for the solution ξ, validated simulation libraries, such as VNODE-LP [39] and CAPD [9], use numerical integration to generate a sequence evaluation of ξ with guaranteed error bounds. We define *simulation* as a sequence of timestamped hyperrectangles that contains the solutions of the system.

Definition 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 Equation (1) is a sequence of timestamped sets $\{(R_i, t_i)_{i=0}^k\}$ satisfying the following:

(1) τ is called the *maximum sampling period*, which means that for each $i = 1, \ldots, k$, $0 < t_i - t_{i-1} \leq \tau$. Note $t_0 = 0$ and $t_k = T$.
(2) Each R_i is a hyperrectangle in $\mathbb{R}^n$ with a diameter smaller than ϵ.
(3) For each $i = 0, 1, \ldots, k$, $\xi(x_0, t_i) \in R_i$, , and $\forall t \in (t_{i-1}, t_i)$, $\xi(x_0, t) \in \texttt{hull}(R_{i-1}, R_i)$, for $i = 1, \ldots, k$.

For a given initial set $\Theta \subseteq \mathcal{R}^n$, a state $x \in \mathbb{R}^n$ is said to be *reachable* if there exist a state $\theta \in \Theta$ and a time $t \geq 0$ such that $\xi(\theta, t) = x$. We denote by $\xi(\Theta, [t_1, t_2])$ the set of states that are reachable from Θ at any time $t \in [t_1, t_2]$. The set of reachable states at time t from initial set Θ is denoted by $\xi(\Theta, t)$. Given an n-dimensional dynamical system as in Equation (1), a compact initial set $\Theta \subset \mathbb{R}^n$, an unsafe set $\overline{\mathcal{S}} \subseteq \mathbb{R}^n$, and a time bound $T > 0$, the *safety verification* problem (also called the bounded invariant verification) is to decide whether $\xi(\Theta, [0, T]) \cap \overline{\mathcal{S}} = \emptyset$.

Next, we introduce the definition of *reachtube*, which is also a sequence of timestamped hyperrectangles, but instead contains $\xi(\Theta, [0, T])$.

Definition 2 (Reachtube). For any $\Theta \subset \mathbb{R}^n, T > 0$, a (Θ, T)-*reachtube* is a sequence of timestamped compact sets $\{(O_i, t_i)_{i=0}^k\}$, such that for each i in the sequence, $\xi(\Theta, [t_{i-1}, t_i]) \subseteq O_i$.

Discrepancy Function. Sensitivity of the solutions to changes in the initial states is formalized by discrepancy functions. Specifically, a discrepancy function bounds the distance between two neighboring trajectories as a function of the distance between their initial states and time [16, 21].

Definition 3 (Discrepancy Function). 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 (1) with initial set Θ if

(1) for any pair of states $x_1, x_2 \in \Theta$, and any time $t \geq 0$, $\|\xi(x_1, t) - \xi(x_2, t)\|_M \leq \beta(\|x_1 - x_2\|_M, t)$, 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 Equation (1), at any time t, the ball centered at $\xi(x_0, t)$ with radius $\beta(\delta, t)$ contains every solution of Equation (1) starting from $B_\delta(x_0)$. Therefore, by bloating the simulation trajectories using the corresponding discrepancy function, we can obtain an overapproximation of the reachtube. Definition 3 corresponds to the definition of

discrepancy function given in [16], except that it generalizes the distance measure to arbitrary M-norm as a metric. We remark that this definition of discrepancy function is similar to the incremental Lyapunov functions; however, here we do not require that trajectories converge to each other.

2.2 Interval Matrices and Matrix Norms

For a pair of matrices $B, C \in \mathbb{R}^{n\times n}$ with the property that $b_{ij} \le c_{ij}$ for all $1 \le i, j \le n$, we define the set of matrices $\texttt{Interval}([B, C]) \triangleq \{A \in \mathbb{R}^{n\times n} | b_{qij} \le a_{ij} \le c_{ij}, 1 \le i, j \le n\}$. Any such set of matrices is called an *interval matrix*. Interval matrices will be used to linearly overapproximate behaviors of nonlinear models. Two useful notions are the *center matrix* and the *range matrix*, defined respectively as $\texttt{CT}([B, C]) = (B + C)/2$ and $\texttt{RG}([B, C]) = (C - B)/2$. Then $\texttt{Interval}([B, C])$ can also be written as $\texttt{Interval}([A_c - A_r, A_c + A_r])$, where $A_c = \texttt{CT}([B, C])$, $A_r = \texttt{RG}([B, C])$.

Given any norm for matrices $\|\cdot\|_p$, where $p \in \{1, 2, \infty, F\}$, the corresponding *norm of an interval matrix* is defined as

$$|||\mathcal{A}|||_p = \sup_{A\in\mathcal{A}} \|A\|_p. \tag{3}$$

The following theorem from [22] provides a method for calculating the norms of interval matrices from the corresponding norms of center and range matrices.

Theorem 4 (Theorem 10 from [22]). *For any interval matrix $\mathcal{A}$,*

$$|||\mathcal{A}|||_1 = \| \, |\texttt{CT}(\mathcal{A})| + \texttt{RG}(\mathcal{A})\|_1, \; |||\mathcal{A}|||_\infty = \| \, |\texttt{CT}(\mathcal{A})| + \texttt{RG}(\mathcal{A})\|_\infty, \; |||\mathcal{A}|||_F = \| \, |\texttt{CT}(\mathcal{A})| + \texttt{RG}(\mathcal{A})\|_F.$$

A *vertex matrix* of an interval matrix $\texttt{Interval}([B, C])$ is a matrix V whose every element is either b_{ij} or c_{ij}. Let $\texttt{VT}(\texttt{Interval}([B, C]))$ be the set of all the vertex matrices of the interval matrix $\texttt{Interval}([B, C])$. The cardinality of $\texttt{VT}(\texttt{Interval}([B, C]))$ with $B, C \in \mathbb{R}^{n\times n}$ is 2^{n^2}. It can be shown that the convex hull of the set of vertex matrices for an interval matrix $\mathcal{A}$ is the matrix itself [20].

Proposition 3. *For any interval matrix $\mathcal{A}$,* $\texttt{hull}(\texttt{VT}(\mathcal{A})) = \mathcal{A}$.

Matrix Measure. The *matrix measure*, also known as the *logarithmic norm* [12], of a matrix $A \in \mathbb{R}^{n\times n}$ can be seen as the one-sided derivative of $\|\cdot\|_p$ at $I \in \mathbb{R}^{n\times n}$ in the direction of A:

$$\mu_p(A) = \lim_{t\to 0^+} (\|I + tA\|_p - \|I\|_p)/t, \tag{4}$$

where $\|\cdot\|_p$ can be any norm. If $p = 2$, the matrix measure becomes $\mu_2(A) = \lambda_{max}((A + A^T)/2)$. If matrix $C \in \mathbb{R}^{n\times n}$ is nonsingular, then the measure μ_M of the norm $\|x\|_M = \|Cx\|$ $(M = C^T C)$ is given in terms of μ by $\mu_M(A) = \mu(CAC^{-1})$. The matrix measure has long been used to provide estimates on solutions of systems of ordinary differential equations. The next proposition from [41] uses the matrix measure to provide 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 4. *Let $\mathcal{D} \subseteq \mathbb{R}^n$ and let the Jacobian $J_f(x)$ satisfy $\mu_p(J_f(x)) \le c$ for all $x \in \mathcal{D}$. If every trajectory of Equation (1) with initial conditions in the line segment $\{hx_0 + (1 - h)z_0 : | \; h \in [0, 1]\}$ remains in $\mathcal{D}$ until time T, then the solutions $\xi(x_0, t)$ and $\xi(z_0, t)$, for all $t \in [0, T]$, satisfy $\|\xi(x_0, t) - \xi(z_0, t)\|_p \le \|x_0 - z_0\|_p e^{ct}$.*

This provides global divergence between trajectories of Equation (1) using only information about the system's Jacobian at each point, and it provides a way to compute a discrepancy function. If there exists $c < 0$ such that for all $(t, x) \in [0, \infty) \times \mathcal{D}$ we have $\mu_p(J_f(x)) \le c$, then Equation (1) or the vector field $f(x)$ is said to be contracting with respect to $\|\cdot\|_p$, but in the following, we do not assume anything about the sign of c.

ALGORITHM 1: Simulation-Driven Verification Algorithm

input: $\Theta, T, \overline{\mathcal{S}}, \epsilon_0, \tau_0$
$\delta \leftarrow \texttt{Dia}(\Theta); \epsilon \leftarrow \epsilon_0; \tau \leftarrow \tau_0; STB \leftarrow \emptyset$
$\mathcal{C} \leftarrow \texttt{Cover}(\Theta, \delta, \epsilon)$
while $\mathcal{C} \neq \emptyset$ **do**
  for $\langle \theta, \delta, \epsilon \rangle \in \mathcal{C}$ **do**
    $\psi = \{(R_i, t_i)_{i=0}^{k}\} \leftarrow \texttt{Simulate}(\theta, T, \epsilon, \tau)$
    $\mathcal{R} \leftarrow \texttt{Bloat}(\psi, \delta, \epsilon)$
    if $\mathcal{R} \cap \overline{\mathcal{S}} = \emptyset$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \backslash \{\langle \theta, \delta, \epsilon \rangle\}$; $STB \leftarrow STB \cup \mathcal{R}$
    else if $\exists j, R_j \subseteq \overline{\mathcal{S}}$ **then**
      return $(\overline{\mathcal{S}}, \psi)$
    else
      $\mathcal{C} \leftarrow \mathcal{C} \cup \texttt{Cover}(B_\delta(\theta), \frac{\delta}{2}, \frac{\epsilon}{2}) \backslash \{\langle \theta, \delta, \epsilon \rangle\}$
      $\tau \leftarrow \frac{\tau}{2}$
return (SAFE, STB)

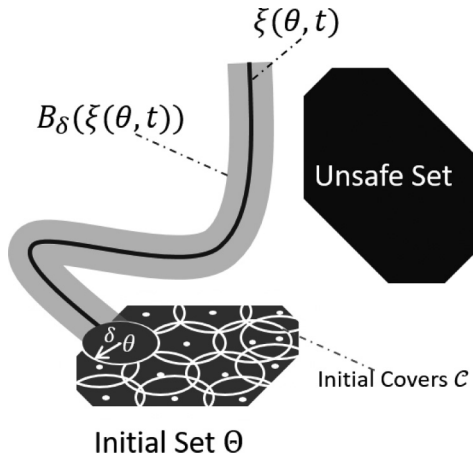

Fig 1. illustration of Algorithm 1.

3 SIMULATION-DRIVEN VERIFICATION

In this section, we give an overview of the simulation-driven verification approach (Algorithm 1) introduced in [13, 16]. Our goal is to decide bounded safety properties, that is, given Equation (1), a compact initial set $\Theta \subset \mathbb{R}^n$, an unsafe set $\overline{\mathcal{S}} \subseteq \mathbb{R}^n$, and a time bound $T > 0$, we would like to check whether $\xi(\Theta, [0, T]) \cap \overline{\mathcal{S}} = \emptyset$. Instead, in this section, we present an algorithm for deciding a relaxation of this problem, which is called *robust safety verification.*

If there exists some $\epsilon > 0$ such that $B_\epsilon(\xi(\Theta, [0, T])) \cap \overline{\mathcal{S}} = \emptyset$, we say the system is *robustly safe.* That is, all states in some envelope around the system behaviors are safe. If there exists some $\epsilon, x \in \Theta$ such that $B_\epsilon(\xi(x, t)) \subseteq \overline{\mathcal{S}}$ 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 returned 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 robustly unsafe.

Algorithm 1 shows the structure of the simulation-driven verification approach. It has been shown that given a discrepancy function for the system, Algorithm 1 is a semidecision procedure for robust safety verification. The algorithm has five inputs: (1) a compact initial set $\Theta \subset \mathbb{R}^n$, (2) time bound $T > 0$, (3) unsafe set $\overline{\mathcal{S}} \subset \mathbb{R}^n$, (4) initial simulation precision ϵ_0, (5) and initial simulation sampling period τ_0, as well asl two outputs SAFE and UNSAFE. It terminates if the system is robustly safe or robustly unsafe.

The simulation-driven approach in Algorithm 1 consists of the following three main steps: Simulate the system from a finite set of sta tes (θ) that are chosen from the compact initial set Θ. The union of a set of balls of diameter δ centered at each of the states should contain Θ. (2) Bloat the $\{(R_i, t_i)_{i=0}^{k}\}$ simulations using a discrepancy function such that the bloated sets are reachtubes from the initial covers. (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 balls with smaller diameter δ.

There are several functions referred to in Algorithm 1. Functions `Dia()` and `Simulate()` are defined to return the diameter of a set and a 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 $B_\delta(\theta)$. This can be done by bloating the simulation using a discrepancy function as described in Section 2.1, which is an overapproximation of the distance between any neighboring trajectories starting from $B_\delta(\theta)$. The main contribution of this article includes algorithms for implementing this `Bloat()` function. Function `Cover()` returns a set of triples $\{\langle\theta, \delta, \epsilon\rangle\}$, where θs are sample states, the union of $B_\delta(\theta)$ covers Θ, and ϵ is the precision of simulation.

Initially, C contains a singleton $\langle\theta_0, \delta_0 = \texttt{Dia}(\Theta), \epsilon_0\rangle$, where $\Theta \subseteq B_{\delta_0}(\theta_0)$ and ϵ_0 is a small positive constant. For each triple $\langle\theta, \delta, \epsilon\rangle \in C$, the **while**-loop from Line 3 checks the safety of the reachtube from $B_\delta(\theta)$, which is computed in Lines 5 and 6. ψ is a $(\theta, T, \epsilon, \tau)$-simulation, which is a sequence of timestamped rectangles $\{(R_i, t_i)\}$ and is guaranteed to contain the trajectory $\xi(\theta, T)$ by Definition 1. Bloating the simulation result ψ by the discrepancy function to get $\mathcal{R}$, a $(B_\delta(\theta), T)$-reachtube, we have an overapproximation of $\xi(B_\delta(\theta), [0, T])$. The core function `Bloat()` will be discussed in detail next. If $\mathcal{R}$ is disjoint from $\overline{\mathcal{S}}$, then the reachtube from $B_\delta(\theta)$ is safe and the corresponding triple can be safely removed from C. If for some j, R_j (one rectangle of the simulation) is completely contained in the unsafe set, then we can obtain a counterexample in the form of a trajectory that violates the safety property. Otherwise, the safety of $\xi(B_\delta(\theta), [0, T])$ is not determined, and a refinement of $B_\delta(\theta)$ needs to be made with smaller δ and smaller ϵ, τ.

Algorithm 1 returns SAFE if the reach set $\xi(\Theta, [0, T])$ has no intersection with the unsafe set, along with a robustly safe reachtube *STB*, or returns UNSAFE upon finding a counterexample, the simulation ψ, which enters the unsafe region. Algorithm 1 is sound because it will only return SAFE if $\xi(\Theta, T) \cap \overline{\mathcal{S}} = \emptyset$. Also, Algorithm 1 is relatively complete, because for a robustly safe or unsafe system, the procedure will continue to perform refinements until either SAFE or UNSAFE is returned.

A crucial and challenging aspect of Algorithm 1 is choosing an appropriate discrepancy function with which to implement the `Bloat()` function. In the next section, we introduce algorithms that implement this function.

4 LOCAL DISCREPANCY ALGORITHMS

The `Bloat()` function is implemented using the notion of discrepancy discussed in Section 2.1, which measures the rate of change of the distance between two neighboring trajectories. We can define different types of discrepancy functions to fit different uses. We will introduce algorithm `LDF2` [21], which uses discrepancy functions defined with the Euclidean norm and is fast but coarse, as it bloats a simulation uniformly in each direction. We will also introduce algorithm `LDFM` [20], which uses a generalized norm and computes an optimal coordinate transformation and results in a tighter reachtube but requires more computation time. We will introduce two different versions 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 can bloat a simulation nonuniformly in different directions and are able to find locally optimal exponential change rates.

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 bound the convergence (or divergence) rates across all trajectories. By restricting the definition of discrepancy functions over carefully computed parts of the state space, we will gain two benefits. First, such local discrepancy functions will still be adequate to compute the relevant reachtubes for safety verification, and second, it will become possible to compute a *local* discrepancy function automatically from simulation traces.

We begin by showing that, over a compact set $S \subseteq \mathbb{R}^n$, the Jacobian J_f of the system described by Equation (1) can be overapproximated by an interval matrix. Then we establish that the distance between two trajectories in S satisfies a differential equation from a set of differential equations described using the interval matrix. By bounding the matrix measure of the interval matrix, we can get a discrepancy function.

Since we assume the system is continuously differentiable, the Jacobian matrix is continuous, and therefore, over a compact set S, the elements of $J_f(x)$ are bounded.

Lemma 5. *If the Jacobian matrix $J_f(x)$ is continuous, then for any compact set S, there exists an interval matrix $\mathcal{A}$ such that $\forall x \in S, J_f(x) \in \mathcal{A}$.*

For interval matrix $\mathcal{A} = \texttt{Interval}(B, C)$, the bounds B and C can be obtained using interval arithmetic or an optimization toolbox by maximizing and minimizing the terms of J_f over S. Such set S can be chosen to be a coarse overapproximation of the reach set, obtained using the Lipschitz constant (see Lemma 9). Once the bounds are obtained, we use the interval matrix that overapproximates the behavior of $J_f(x)$ over S to analyze the rate of convergence or divergence between trajectories:

Lemma 6. *For Equation (1) with initial set Θ starting from time t_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 $x \in \Theta$, $t \in [t_1, t_2]$, $\xi(x, t) \in S$. If there exists an interval matrix $\mathcal{A}$ such that $\forall x \in S, J_f(x) \in \mathcal{A}$, then for any $x_1, x_2 \in \Theta$, 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 from Θ in the time interval $[t_1, t_2]$, using Lemma 1, we have the following:

$$\dot{y}(t) = \dot{\xi}(x_2, t) - \dot{\xi}(x_1, t) = f(\xi(x_2, t)) - f(\xi(x_1, t)) = \left(\int_0^1 J_f(\xi(x_1, t) + sy(t))ds \right) y(t), \quad (5)$$

where $y(t)$ is the distance $\xi(x_2, t) - \xi(x_1, t)$ starting from $x_1, x_2 \in \Theta$. We construct $\mathcal{A}$ as the interval matrix to bound the Jacobian matrix using Lemma 5. 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, $J_f(\xi(x_1, t) + sy(t)) \in \texttt{Interval}([B, C])$ and it is straightforward to check that $\int_0^1 J_f(\xi(x_1, t) + sy(t))ds \in \mathcal{A}$. We rewrite Equation (5) as

$$\dot{y}(t) = A(t)y(t), A(t) \in \mathcal{A}, \quad (6)$$

which means at any fixed time $t \in [t_1, t_2]$, $\dot{y}(t) = A(t)y(t)$ always holds, where $A(t)$ is an element of $\mathcal{A}$. □

Equation (6) used in Lemma 6 can be used to define 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 $t \in [t_1, t_2]$,

$$\frac{d\|y(t)\|_M^2}{dt} = \dot{y}^T(t)y(t) + y^T(t)\dot{y}(t) = y^T(t)(A(t)^T M + MA(t))y(t), \quad (7)$$

for some $A(t) \in \mathcal{A}$. We write $A(t)$ as A in the following for brevity. If there exists a $\hat{\gamma}$ such that $A^T M + MA \preceq \hat{\gamma} M, \forall A \in \mathcal{A}$, then Equation (7) 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 2.2). 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 postmultiplying the inequality by C^T and C, we arrive at $A^T M + MA \preceq \hat{\gamma} M, \forall A \in \mathcal{A}$.

The aforementioned provides a discrepancy function:

$$\beta(\|x_1 - x_2\|_M, t) = \|x_1 - x_2\|_M e^{\frac{\hat{\gamma}}{2}(t-t_1)}. \tag{8}$$

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 the tightest bound $\hat{\gamma}$. This problem is formulated as follows:

$$\begin{aligned} \min_{\hat{\gamma} \in \mathbb{R}, M > 0} \quad & \hat{\gamma} \\ \text{s.t} \quad & A^T M + MA \preceq \hat{\gamma} M, \quad \forall A \in \mathcal{A}. \end{aligned} \tag{9}$$

To compute the reach set 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 overapproximations. To mitigate this problem, we divide the time interval $[0, T]$ into smaller intervals $[0, t_1], [t_1, t_2]$, and so forth, 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 and then using the reach set overapproximation at t_i as the initial set for the time interval $[t_i, t_{i+1}]$.

Solving Equation (9) to obtain 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 this problem, we introduce two different strategies. The first one only considers discrepancy functions under the Euclidean norm, and thus, avoids solving the optimization problem in Equation (9), while the second tries to transform the problem in Equation (9) to equivalent or relaxed problems but with finitely many constraints.

4.2 Fast Discrepancy Function

In this section, we introduce a method to compute the discrepancy function without solving the optimization problem in Equation (9), which leads to an algorithm that can compute the reachtube fast but with possibly larger approximation errors compared to the methods discussed in Section 4.3.

4.2.1 Local Discrepancy Under Euclidean Norm. The optimization problem in Equation (9) attempts to find the optimal metric M such that the exponential changing rate $\hat{\gamma}$ of the discrepancy function is minimized. Informally, solving Equation (9) is complicated as the constraints consider all the possible behaviors of $J_f(x)$ over the compact sets. A relaxation of this problem is to fix $M = I$, and then the largest eigenvalue of the symmetric part of the Jacobian matrix can be used as the exponential change rate of the system.

Lemma 7. *For Equation (1) with initial set Θ starting from time t_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 $x \in \Theta$, $t \in [t_1, t_2]$, $\xi(x, t) \in S$. Suppose $\gamma \in \mathbb{R}$*

satisfies $\forall x \in S$, $\lambda_{max}((J_f^T(x) + J_f(x)))/2 \leq \gamma$; *then, for any* $x_1, x_2 \in \Theta$, *and for any* $t \in [t_1, t_2]$,

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

Lemma 7 follows from Equation (8) and the fact that when $M = I$, $\mu_2(J_f(x)) = \lambda_{max}((J_f^T(x) + J_f(x)))/2$. Computing a bound γ on $\lambda_{max}(J_f^T(x) + J_f(x))/2, x \in S$ is difficult because this bound must work for the infinite family of matrices. We introduce Algorithm 2 to compute an upper bound on the eigenvalues of the symmetric part of the Jacobian function.

ALGORITHM 2: Algorithm `Eig_UB`.

input: $J_f(\cdot), S \subseteq \mathbb{R}^n$
1 J $\leftarrow J_f$(`Center`(S))
2 $\lambda \leftarrow \lambda_{max}(J + J^T)/2$
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
7
8
9

ALGORITHM 3: Algorithm `LDF2`.

input: $\psi = \{(R_i, t_i)_{i=0}^k\}, J_f(\cdot), L_f, \delta, \epsilon$
1 $\Delta \leftarrow \delta, b \leftarrow$ zeros(k)
2 **for** i = 1:k **do**
3 $\quad \tau \leftarrow t_i - t_{i-1}$
4 $\quad d \leftarrow (\Delta + \epsilon)e^{L_f \tau}$
5 $\quad$ S $\leftarrow$ `hull`$(R_{i-1}, R_i) \oplus B_d(0)$
6 $\quad \gamma[i] \leftarrow$ `Eig_UB`$(J_f(\cdot), S)$
7 $\quad O_i \leftarrow B_{\delta'}$ (`hull`(R_{i-1}, R_i)) where
$\quad\quad \delta' = \max\{(\Delta + \epsilon), (\Delta + \epsilon)e^{\gamma[i]\tau}\}$
8 $\quad \Delta \leftarrow (\Delta + \epsilon)e^{\gamma[i]\tau}$
9 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup [O_i, t_i]$
10 **return** $\mathcal{R}$

Algorithm 2 is based on a matrix perturbation theorem (Theorem 2 in [21]). 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 $\mathcal{A}$ overapproximates 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. This step can be accomplished using Theorem 4 and the fact that $\forall A \in \mathbb{R}^{n \times n}$, $\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$. 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.

The following lemma shows that the value returned by Algorithm 2 is an upper bound on $\lambda_{max}(J_f(x) + J_f(x)^T)/2$, where $x \in S$. A detailed proof can be found in [21].

Lemma 8. *If c is the value returned by Algorithm 2, then* $\forall x \in S : J_f^T(x) + J_f(x) \leq 2cI$.

According to Lemma 5, 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, we can find the upper bound of $\|E(x)\|$, so the error term is also bounded. Therefore, the value returned by Algorithm 2 for a continuous-term Jacobian function over a compact set is always bounded.

Since $J_f : S \to \mathbb{R}^{n \times n}$ is bounded, 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. Using the computed S and γ, Lemma 7 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 hyberball centered at $\xi(x_1,t)$ with radius $ce^{\gamma(t-t_1)}$. Thus, a discrepancy function for Equation (1) over S is given by $\beta(\|x_1-x_2\|,t)=\|x_1-x_2\|e^{\gamma(t-t_1)}$.

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

$$\dot{v}=\tfrac{1}{2}(v^2+w^2);\quad \dot{w}=-v. \tag{10}$$

The Jacobian matrix of the system is $\left[\begin{smallmatrix} v & w \\ -1 & 0\end{smallmatrix}\right]$. 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 7, we obtain the following discrepancy function for this system: $\|\xi(x_1,t)-\xi(x_2,t)\|\le\|x_1-x_2\|e^{1.0178t}$, 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 7 and Algorithm 2 to compute a $(B_\delta(R_0),T)$-reachtube of Equation (1), where R_0 is the initial rectangle of the simulation ψ.

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 (θ,τ,ϵ,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 overapproximation of the reachable states from $B_\delta(R_0)$ at t_{i-1} (Lemma 10). 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) overapproximation of the reachtube from $B_\Delta(R_{i-1})$ over the time interval $[t_{i-1},t_i]$ (Lemma 9). 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 8). At Line 7, the rectangle O_i is computed as an overapproximation 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 prove that Algorithm 3 returns a $(B_\delta(R_0),T)$-reachtube. First, we show that the set S computed at Line 5 satisfies the assumption in Lemma 7. That is, in the i^{th} iteration of the loop, the computed S is an overapproximation 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 9. *In the i^{th} iteration of the loop of Algorithm 3, $\xi(B_\Delta(R_{i-1}),[t_{i-1},t_i])\subseteq S$.*

Lemma 9 follows from the fact that the Lipschitz constant provides a very conservative discrepancy function $\beta(\|x-x'\|_M,t)=\|x-x'\|_M e^{L_f(t-t_{i-1})}$. Note that since the Lipschitz constant is nonnegative, such discrepancy function usually blows up within a short time horizon.

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

Lemma 10. *For any $i=0,\ldots,k$, $\xi(B_\delta(R_0),t_i)\subseteq B_{\Delta_i}(R_i)$, and for any $i=1,\ldots,k$, $\xi(B_\delta(R_0),[t_{i-1},t_i])\subseteq O_i$, 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 Definition 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 $i=0$, we already have $\xi(B_\delta(R_0),t_0)=B_\delta(R_0)=B_{\Delta_0}(R_0)$.

Assume that the first half of the lemma holds for $i=m-1$, which means we have $\xi(B_\delta(R_0),t_{m-1})\subseteq B_{\Delta_{m-1}}(R_{m-1})$. Next we prove that the lemma holds for the second half and for

$i = m$ as well. Lemma 9 indicates that $\forall t \in [t_{m-1}, t_m], \xi(B_{\Delta_0}(R_0), [t_{m-1}, t_m]) \subseteq S_m$. 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 $\xi(x, t) \in \mathtt{hull}(R_{m-1}, R_m)$ 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 7:

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

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

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

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

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

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

From Lemma 10, the following main theorem of this section follows. It states that the Algorithm LDF2 soundly overapproximates the reachable states from $B_\delta(R_0)$.

Theorem 11. *For any (x, τ, ϵ, T)-simulation $\psi = (R_0, t_0) \ldots (R_k, t_k)$ and any constant $\delta \geq 0$, a call to $\mathtt{LDF2}(\psi, \delta, \epsilon)$ 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 discrepancy function computed by Lemma 7 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 7 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.[1]

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 overapproximation may not converge over time.

To overcome these shortcomings of Algorithm LDF2, we will try to solve the optimization problem in Equation (9) by replacing the constraints $A^T M + MA \preceq \hat{\gamma} M, \ \forall A \in \mathcal{A}$ with a finite number of constraints containing certain representative matrices and analyzing the consequences of such replacement.

To simplify the presentation, we assume that the solutions (i.e., trajectories) for Equation (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 [16]).

4.3.1 Vertex Matrix Constraints Method. Proposition 3 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 $\mathtt{VT}(\mathcal{A})$. This allows us to simplify the optimization problem in Equation (9) to one with a finite number of constraints, based on the

[1] In [21], 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.

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

Lemma 12. *For Equation (1) with initial set Θ starting from time t_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 $x \in \Theta$, $t \in [t_1, t_2]$, $\xi(x,t) \in S$. 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) $\exists\, \hat{\gamma} \in \mathbb{R}$, $\forall\, A_i \in \mathsf{VT}(\mathcal{A})$, $A_i^T M + M A_i \preceq \hat{\gamma} M$,

then for any $x_1, x_2 \in \Theta$ 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 3, we know that $J_f(x) \in \mathcal{A} = \mathtt{hull}(\mathsf{VT}(\mathcal{A}))$. It follows from (a) and Lemma 6 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 $A \in \mathtt{hull}\{A_1, A_2, \ldots, A_N\}$. Using this, at any time $t \in [t_1, t_2]$, the derivative of $\|y(t)\|_M^2$ can be written as

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

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

Lemma 12 suggests the following bilinear optimization problem for finding discrepancy over compact subsets of the state space:

$$\begin{aligned}\min_{\hat{\gamma} \in \mathbb{R}, M > 0} \quad & \hat{\gamma} \\ \text{s.t.} \quad & \text{for each } A_i \in \mathsf{VT}(\mathcal{A}), A_i^T M + M A_i \preceq \hat{\gamma} M.\end{aligned} \tag{11}$$

Letting $\hat{\gamma}_{\max}$ be the maximum of the eigenvalues of $A_i^T + A_i$ for all i, $A_i^T + A_i \preceq \hat{\gamma}_{\max} I$ (i.e., $M = I$) holds for every A_i, so a feasible solution exists for Equation (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 Equation (11) is a semidefinite program (SDP), and a feasible solution can be obtained by an SDP solver. As a result, we can solve Equation (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 aforementioned process for identifying a feasible (optimal) $\hat{\gamma}$ and a corresponding M can be used to compute reach set overapproximations, 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 2. Consider Equation (10) over the given compact set S as in Example 1. By solving the optimization problem in Equation (11), we can obtain $\hat{\gamma} = -0.6$ and $M = \begin{bmatrix} 2.7263 & -1.3668 \\ -1.3668 & 6.7996 \end{bmatrix}$. Then, by invoking Lemma 12, 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 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 overapproximation computed using this method is less conservative than the method based on the 2-norm (Section 4.2), 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 $\mathtt{VT}(\mathcal{A})$ that appear in the SDP (Equation (11)). In the next section, we present a second method that avoids the exponential increase in the number of constraints in Equation (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 13. *For Equation (1) with initial set Θ starting from time t_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 $x \in \Theta$, $t \in [t_1, t_2]$, $\xi(x,t) \in S$. 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 $x_1, x_2 \in \Theta$ and $t \in [t_1, t_2]$:

$$\|\xi(x_1,t) - \xi(x_2,t)\|_M \leq e^{\left(\frac{\hat{\gamma}}{2} + \frac{\delta}{2\lambda_{min}(M)}\right)(t-t_1)} \|x_1 - x_2\|_M, \tag{12}$$

where $\delta = \sqrt{|||\mathcal{D}|||_1 |||\mathcal{D}|||_\infty}$, and $\mathcal{D} = \{D \mid \exists A \in \mathcal{A}$ such that $D = (A - \mathtt{CT}(\mathcal{A}))^T M + M(A - \mathtt{CT}(\mathcal{A}))\}$ is also an interval matrix.

Proof. Fix any $x_1, x_2 \in \Theta$. 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 $\mathcal{G} = \mathtt{Interval}([-A_r, A_r]) = \{G \mid \exists A \in \mathcal{A}$ such that $G = A - \mathtt{CT}(\mathcal{A})\}$. We use a standard property of norms to bound the 2-norm as follows (see [25], page 57):

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

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 $\forall y \neq 0$, and

$$0 < \lambda_{\min}(M)\|y\|^2 \leq y^T M y. \tag{13}$$

Moreover, $\forall G \in \mathcal{G}$, and any vector $y \in \mathbb{R}^n y^T (G^T M + MG) y \leq \|G^T M + MG\|_2 \|y\|^2$. Combining with Equation (13) yields

$$y^T (G^T M + MG) y \leq \delta \|y\|^2. \tag{14}$$

It follows from Lemma 6 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 previous inequalities, we have

$$\begin{aligned}\frac{d\|y(t)\|_M^2}{dt} &= y^T(t)\left((A_c^T + G^T)M + M(A_c + G)\right)y(t) = y^T(t)(A_c^T M + MA_c + G^T M + MG)y(t)\\ &\le \hat{\gamma} y^T(t)My(t) + y^T(t)(G^T M + MG)y(t) \le \hat{\gamma}\|y(t)\|_M^2 + \delta\|y\|^2(t) \quad \text{[using Equation (14)]}\\ &\le \hat{\gamma}\|y(t)\|_M^2 + \delta\frac{\|y(t)\|_M^2}{\lambda_{\min}(M)}. \quad \text{[using Equation (13)]}\end{aligned}$$

The lemma follows by applying Grönwall's inequality. □

In general, Lemma 13 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:

$$\begin{aligned}\min_{\hat{\gamma}\in\mathbb{R}, M>0} \quad & \hat{\gamma} \\ \text{s.t} \quad & A_c^T M + MA_c \preceq \hat{\gamma} M, A_c = \mathtt{CT}(\mathcal{A}).\end{aligned} \tag{15}$$

Remark 1. In Lemma 13, 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 4 provides an efficient way that 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 3. For Equation (10) over the given compact set S as in Example 1, we can obtain $\hat{\gamma} = -0.8$ and $M = \begin{bmatrix} 2.4431 & -1.0511 \\ -1.0511 & 4.5487 \end{bmatrix}$ by solving the optimization problem in Equation (15), and $\delta = 1.4162$. Applying Lemma 13, a discrepancy function for Equation (10) is given by $\|\xi(x_1, t) - \xi(x_2, t)\|_M \le \|x_1 - x_2\|_M e^{0.3081t}$ for as long as the trajectories remain inside S.

Compared with Examples 1 and 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 Equation (12). In practice, we want to make the compact sets S small so that δ (and by extension the exponential term in Equation (12)) remains small.

Lemmas 12 and 13 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 $\|x_1 - x_2\|_M \le c$, we will have that $\forall t \in [t_1, t_2]$, $\|\xi(x_1, t) - \xi(x_2, t)\|_M \le ce^{\frac{\gamma'}{2}(t - t_1)}$ ($\gamma' = \hat{\gamma}$ for Lemma 12 and $\gamma' = \hat{\gamma} + \frac{\delta}{\lambda_{min}(M)}$ for Lemma 13). 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 $\|(\xi(x_1, t) - x)\|_M^2 \le ce^{\gamma'(t - t_1)}$. 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).

4.3.3 Algorithm to Compute Local Optimal Reach Set. Given an initial set $B_\delta(x)$ and time bound T, Lemmas 12 and 13 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 piece-wise discrepancy function, where each piece is relevant for a smaller portion of the state space and time.

Consider two adjacent subintervals of $[0, T]$, $a = [t_1, t_2]$ and $b = [t_2, t_3]$. Let $E_{M_a, c_a(t_2)}(\xi(x_0, t_2))$ be an ellipsoid that contains $\xi(B_\delta(x), t_2)$, and suppose we are given an M_b and we want to select a $c_b(t)$ such that $\xi(B_\delta(x), t_2) \subseteq E_{M_b, c_b(t_2)}(\xi(x_0, t_2))$. To overapproximate the reach set for the interval b, we require that $c_b(t_2)$ is chosen so that at the transition time t_2,

$$E_{M_a, c_a(t_2)}(\xi(x_0, t_2)) \subseteq E_{M_b, c_b(t_2)}(\xi(x_0, t_2)). \tag{16}$$

This is a standard SDP problem to compute the minimum value for $c_b(t_2)$ that ensures Equation (16) (see, e.g., [7]). 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 (16) holds can be expressed using the following optimization problem:

$$\begin{aligned} &\min \quad c \\ &s.t. \quad Eb \supseteq Ea. \end{aligned} \tag{17}$$

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

We can transfer Equation (17) to the following *sum-of-squares* problem as the "S procedure" [35] to make it solvable by SDP solvers:

$$\begin{aligned} &\min \quad c \\ &s.t. \quad c - \|x - \xi(x_0, t_2)\|^2_{M_b} - \lambda \left(c_a(t_2) - \|x - \xi(x_0, t_2)\|^2_{M_a} \right) \geq 0, \lambda \geq 0. \end{aligned} \tag{18}$$

4.3.4 Reachtube Overapproximation Algorithm. We present an algorithm to compute a $(B_\delta(x), T)$-reachtube for Equation (1) using the results from Sections 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 timestamped sets (O_1, t_1), $(O_2, t_2), \ldots, (O_k, t_k)$ 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 hyperrectangles for ease of exposition. At the end of this section (Remark 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 hyperrectangles $Rec(t_{i-1}, t_i) \subseteq \mathbb{R}^n$—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); and (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 12 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 in Equation (11) in each time interval to get the local optimal rate, and solve the optimization problem in Equation (16) 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 $\xi(B_\delta(x), t)$ for every $t \in [t_{i-1}, t_i]$ (see Lemma 9). 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 5. 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 12 (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,

ALGORITHM 4: Algorithm LDFM

input: ψ,$J_f(\cdot)$,L,M_0, c_0
initially:~$\mathcal{R} \leftarrow \emptyset, \gamma_0 \leftarrow -100$
1 $\delta_0 = \texttt{Dia}\left(E_{M_0,c_0}(x)\right)$
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 Equation (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 $\quad c_i \leftarrow c_{tmp} e^{\gamma_i \Delta t}$
14 $\quad \delta_i \leftarrow \texttt{Dia}(E_{M_i,c_i}(\xi(x, t_i)))$
15 $\quad O_i \leftarrow B_{\delta'/2}(Rec(t_{i-1}, t_i))$ where $\delta' = \max\{dia\left(E_{M_i,c_{tmp}}(\xi(x, t_{i-1}))\right), \delta_i\}$
16 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup [O_i, t_i]$
17 **return** $\mathcal{R}$

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 that Lemma 12 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 12, 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, Equation (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 13, we compute the updated ellipsoid size c_i such that $E_{M_i,c_i}(\xi(x, t_i))$ contains $\xi(B_\delta(x), t_i)$. At Line 14, the diameter of $E_{M_i,c_i}(\xi(x, t_i))$ is assigned to δ_i for the next iteration. At Line 15, the set O_i is computed such that it contains the reach set during time interval $[t_{i-1}, t_i]$. Finally, at Line 15, $\mathcal{R}$ is returned as an overapproximation 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 overapproximates 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 14. *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 $\xi(E_{i-1}, t_i) \subseteq E_{M',c'}(\xi(x, t_i))$ where c' is computed from γ' (Line 13), 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 overapproximates 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 O_i computed at Line 15 is an overapproximation of the reach set during time interval $[t_{i-1}, t_i]$.

Lemma 15. *For Algorithm* LDFM*, at the* i^{th} *iteration, if* $\xi(B_\delta(x), t_{i-1}) \subseteq E_{M_{i-1},c_{i-1}}(\xi(x, t_{i-1}))$*, then we have at time* t_i*,* $\xi(B_\delta(x), t_i) \subseteq E_{M_i,c_i}(\xi(x, t_i))$*, and* $\xi(B_\delta(x), [t_{i-1}, t_i]) \subseteq O_i$*.*

Proof. Note that by Lemma 12, 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

$$\|\xi(x,t) - \xi(x',t)\|_{M_i} \leq \|\xi(x,t_{i-1}) - x'\|_{M_i} e^{\frac{\gamma_i}{2}(t-t_{i-1})}. \tag{19}$$

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 15, we want to compute the set O_i such that it contains the reach set during time interval $[t_{i-1}, t_i]$. According to Equation (19), 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 $c(t) = c_{tmp}e^{\gamma_i(t-t_{i-1})}$. O_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 $\max\{dia(E_{M_i,c_{tmp}}(\xi(x, t_{i-1}))), \delta_i\}$. Thus, at Line 15, $\xi(B_\delta(x), [t_{i-1}, t_i]) \subseteq O_i$. □

Next, we show that $\mathcal{R}$ returned at Line 15 is an overapproximation of the reach set.

Theorem 16. *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 15, 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 14 contains $\xi(B_\delta(x), t_1)$. Also at Line 15, O_1 contains $\xi(B_\delta(x), [t_0, t_1])$. Repeating this reasoning for subsequent iterations, we have that $E_{M_i,c_i}(\xi(x, t_i))$ contains $\xi(B_\delta(x), t_i)$, and O_i contains $\xi(B_\delta(x), [t_{i-1}, t_i])$. Therefore, $\mathcal{R}$ returned at Line 15 is a $(B_\delta(x), T)$-reachtube. □

Remark 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, one has to simply modify Lines 6, 8, 11, and 13 using Lemma 13 and the optimization problem in Equation (15). 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 13. Such an error term may introduce conservativeness, but this relaxation decreases the computational cost (see Section 4.3.6).

Remark 3. It is straightforward to modify Algorithm 4 to accept validated simulations and the error bounds introduced. At Line 4 and Line 15, 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 14, when using the ellipsoid $E_{M_i,c_i}(\xi(x, t_i))$, we use $E_{M_i,c_i}(0) \oplus R_i$.

4.3.5 Accuracy of Algorithm LDFM*.* Theorem 16 ensures that Algorithm LDFM overapproximates the reach 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 $\mathcal{R} = (O_1, t_1), \ldots, (O_k, t_k = T)$ is a $(B_\delta(x), T)$-reachtube returned by Algorithm LDFM. The first Proposition 17 establishes that the

bloating factor δ_i in Line 14 for constructing reachtubes goes to 0 as the size of the initial set $B_\delta(x)$ goes to 0. This implies that the overapproximation error from bloating can be made arbitrarily small by making the uncertainty in the initial cover $\langle x, \delta, \epsilon \rangle$ small.

Proposition 17. *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* $\mathtt{Dia}(B_\delta(x)) \to 0$ *the size of the bloating factor $\delta_i \to 0$ (Line 14).*

Proof. At Line 13, 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 $\mathtt{Dia}(B_\delta(x)) \to 0$, 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 14, it follows that $\delta_i = 2\sqrt{\lambda_{max}(c_i M_i^{-1})}$ (see [34], 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 a negative matrix measure under certain coordinate transformation. Next, Corollary 18 establishes that for contractive systems the reachtube computed by Algorithm LDFM converges to the rectangles that represent the simulation.

Corollary 18. *Consider a contractive system for which there exists a matrix M such that $\forall x \in \mathbb{R}^n, J_f(x)^T M + M J_f(x) \preceq \gamma M$, and $\gamma < 0$. Computing the reachtube of the system using Algorithm* LDFM*, we have as $k, T \to \infty$,*

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

Proof. From the contractive condition, we have a uniform matrix M such that any evaluation of the Jacobian matrix satisfies $J_f(x)^T M + M J_f(x) \preceq \gamma M$. The "if" condition at Line 6 will always hold for $M_i = M$ and $\gamma_i = \gamma$, and at Line 13 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 O_k, we have $O_k \supseteq Rec(t_{k-1}, T)$. The bloating factor for $Rec(t_{k-1}, T)$ goes to 0, so $O_k \to Rec(t_{k-1}, T)$, and the result follows. □

Corollary 19. *Consider a linear system $\dot{x} = Ax$ with a Hurwitz matrix A. Compute the reachtube of the system using Algorithm* LDFM*; we have as $k, T \to \infty$,*

$$|\mathtt{Dia}(O_k) - \mathtt{Dia}\,(Rec(t_{k-1}, T))\,| \to 0.$$

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 postmultiplying by P^T and P, we get $A^T P^T P + P^T PA \preceq \epsilon P^T P$. 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 accumulated error introduced using Algorithm LDFM. We have also proved the convergence of the algorithm for contractive nonlinear systems in Corollary 18. For noncontractive nonlinear systems, the overapproximation error might be accumulated. Such error caused by a wrapping effect in the on-the-fly algorithms may not be avoidable. Therefore, for noncontractive or unstable nonlinear systems, it is especially important to reduce the overapproximation error in each time interval, which is what Algorithm LDFM aims to achieve.

4.3.6 Computational Considerations 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 one matrix inequality. At Line 8 or Line 11, the vertex matrix constraints method will solve one convex optimization problem with $2^{n_{\mathcal{I}}}+1$ constraints, but the matrix interval method solves one convex optimization problem with two constraints. The discrepancy function updating at Line 12 solves one SDP problem. The rest of the algorithm from Line 13 to Line 15 consists of algebraic operations.

From the previous 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 nonconstant 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 noncontractive systems, this may not be the case. Also, the algorithm only computes the discrepancy function once for the linear system, 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.

4.4 Input-to-State Discrepancy

We consider an extension of the aforementioned reachset estimation methods to systems with inputs. The definition for input-to-state discrepancy appeared in [28]. Lemma 20 gives a concrete method for computing input-to-state discrepancy functions in a slightly different format. Consider the dynamical system with input $\dot{x} = f(x,u)$, where $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is Lipschitz continuous. The input u could be control inputs, disturbances, or parameters. For a given input signal that is an integrable function $v : [0,\infty) \to \mathbb{R}^p$ and an initial state $x_0 \in \mathbb{R}^n$, a solution (or trajectory) of the system is a function $\xi : \mathbb{R}^n \times \mathbb{R}^{\geq 0} \to \mathbb{R}^n$ such that $\xi(x_0,0) = x_0$ and for any time $t \geq 0$, $\dot{\xi}(x,t) = f(\xi(x,t),v(t))$. Let $\mathcal{U}$ be the set $\{u | u : [0,\infty) \to \mathbb{R}^p\}$ of all input signals. Lemmas 12 and 13 can be extended to overapproximate the uncertainties introduced by both the initial states and the inputs.

Lemma 20. *Let $\Theta \subset \mathbb{R}^n$ be the initial set, $S \subseteq \mathbb{R}^n$ be a compact set, and $\mathcal{U}$ be the set of input signals, such that for any state $x \in \Theta$, we have $\xi(x,t) \in S$ with input $u(t) \in \mathcal{U}$ for $t \in [t_1,t_2]$. Let $J_x(x,u)$ and $J_u(x,u)$ be the Jacobian matrices with respect to x and u, respectively. Let M be a positive definite $n \times n$ matrix. If there exists interval matrices $\mathcal{A}$ and $\mathcal{B}$ such that*

(a) $\forall\, x \in S, \forall u \in \mathcal{U}$, $J_x(x,u) \in \mathcal{A}$, *and* $J_u(x,u) \in \mathcal{B}$,
(b) $\exists\, \hat{\gamma} \in \mathbb{R}, \forall\, A_i \in \mathsf{VT}(\mathcal{A})$, $A_i^T M + M A_i \preceq \hat{\gamma} M$, *and*
(c) $\hat{B} = |||\mathcal{B}|||_2$,

then for any $x_1, x_2 \in \Theta, u_1, u_2 \in \mathcal{U}$ *and* $t \in [t_1, t_2]$, *where* u_1 *are* u_2 *are the inputs for* $\xi(x_1, t), \xi(x_2, t)$, *respectively:*

$$\|\xi(x_1,t) - \xi(x_2,t)\|_M \leq e^{\frac{\hat{\gamma}+1}{2}(t-t_1)}\|x_1 - x_2\|_M + \hat{B}\sqrt{\int_{t_1}^{t} e^{(\hat{\gamma}+1)(t-\tau)}\|u_2(\tau) - u_1(\tau)\|_M^2 d\tau}. \tag{20}$$

Proof. Let $y(t) = \xi(x_2, t) - \xi(x_1, t)$ and $v(t) = u_2(t) - u_1(t)$. From the high-order mean value theorem, we obtain

$$\begin{aligned}\dot{y}(t) &= f(\xi(x_2,t), u_2(t)) - f(\xi(x_1,t), u_1(t)) \\ &= \int_0^1 J_x\left(\xi(x_1,t) + sy(t), u_2(t)\right) y(t) ds + \int_0^1 J_u\left(\xi(x_1,t), u_1(t) + \tau v(t)\right) v(t) d\tau.\end{aligned} \tag{21}$$

From the hypothesis of the lemma, we have that for some time interval $[t_1, t_2]$, $\forall s \in [0,1]$, $J_x\left(\xi(x_1,t) + sy(t), u_2(t)\right)$ is contained in the interval matrix $\mathcal{A}$ and $\forall \tau \in [0,1]$, *and* $J_u\left(\xi(x_1,t), u_1(t) + \tau v(t)\right)$ is contained in $\mathcal{B}$. Let $M = C^T C$. It follows from (a) and Lemma 6 that for any $t \in [t_1, t_2]$, there exists matrices $A \in \mathcal{A}$ and $B \in \mathcal{B}$ such that $\dot{y}(t) = Ay(t) + Bv(t)$, 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

$$\begin{aligned}\frac{d\|y(t)\|_M^2}{dt} &= y^T(t)\left(A^T M + MA\right) y(t) + 2(CBv(t))^T (Cy(t)) \\ &\leq y^T(t)\left(A^T M + MA\right) y(t) + y^T(t) M y(t) + v^T(t) B^T M B v(t) \\ &\quad \text{[using inequality } \forall a, b \in \mathbb{R}^n, 2a^T b \leq a^T a + b^T b\text{]} \\ &= y^T(t)\left(\sum_{i=1}^{N} \alpha_i (A_i^T M + MA_i) + M\right) y(t) + v^T(t) B^T M B v(t) \\ &\leq y^T(t)\left(\sum_{i=1}^{N} \alpha_i \hat{\gamma} M + M\right) y(t) + \|Bv(t)\|_M \\ &= (\hat{\gamma}+1) y^T(t) M y(t) + \|Bv(t)\|_M^2 = (\hat{\gamma}+1)\|y(t)\|_M^2 + \|Bv(t)\|_M^2.\end{aligned}$$

By applying Grönwall's inequality, we obtain

$$\begin{aligned}\|y(t)\|_M^2 &\leq e^{(\hat{\gamma}+1)(t-t_1)}\|y(t_1)\|_M^2 + \int_{t_1}^{t} e^{(\hat{\gamma}+1)(t-\tau)}\|Bv(\tau)\|_M^2 d\tau \\ &\leq e^{(\hat{\gamma}+1)(t-t_1)}\|y(t_1)\|_M^2 + \hat{B}^2 \int_{t_1}^{t} e^{(\hat{\gamma}+1)(t-\tau)}\|v(\tau)\|_M^2 d\tau.\end{aligned}$$

By inequality $\forall a, b \in \mathbb{R}^{\geq 0}, \sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, we get that $\|y(t)\|_M \leq e^{\frac{(\hat{\gamma}+1)}{2}(t-t_1)}\|y(t_1)\|_M + \hat{B}\sqrt{\int_{t_1}^{t} e^{(\hat{\gamma}+1)(t-\tau)}\|v(\tau)\|_M^2 d\tau}$. □

The Jacobian matrix $J_u(x,u)$ is an $n \times p$ dimensional matrix and its size also depends on the dimensionality of the input p. To get the interval matrix $\mathcal{B}$, $2n_{\mathcal{I}}$ optimization problems or interval arithmetic needs to be solved if $n_{\mathcal{I}}$ entries in $J_u(x,u)$ are not constant numbers. We notice that the input-to-state discrepancy computed by Lemma 20 can be more conservative than Lemmas 12 and 13 due to the constant $\frac{1}{2}$ that is added to the exponential change rate $\hat{\gamma}$ in Equation (20). In Section 5, we will also discuss the effect of such exponential error.

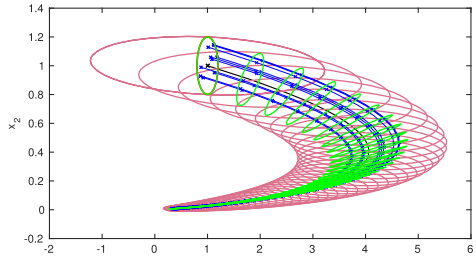

Fig. 2. Reach set overapproximation using Algorithm LDFM (red ellipsoid) and exact reach set (green ellipsoid) of linear system with nilpotent term. Initial set: a circle centered at $[1, 1]^T$ with radius 0.2.

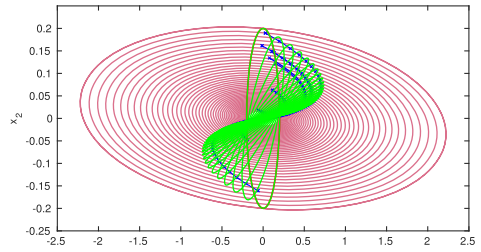

Fig. 3. Reach set overapproximation using Algorithm LDFM (red ellipsoid) and exact reach set (green ellipsoid) of linear system with nilpotent term. Initial set: a circle centered at $[0, 0]^T$ with radius 0.2.

Table 1. Conservativeness of Algorithm LDFM on System $\dot{x} = (-\epsilon I + N)x$

	Algorithm LDFM		Exact Ellipsoid		Exact/ Algorithm LDFM Ratio	
TH(s)	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%

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 sampled overapproximation reach set over the initial set volume. **F/I VR**: the ratio of the volume of the overapproximation 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 overapproximation computed by Algorithm LDFM.

5 EXPERIMENTAL EVALUATION

We first use a small linear example to show the level of conservativeness that Algorithm LDFM adds to the exact reach set. Then, we evaluate the efficiency and accuracy of presented algorithms on a set of benchmarks and compare the results with Flow*.

5.1 Accuracy of Algorithm LDFM

First, we illustrate the accuracy of Algorithm LDFM using a 2-dimensional linear system with a nilponent term: $\dot{x} = (-\epsilon I + N)x$, where $\epsilon = 1/10$, I is the identity matrix, and $N = [0, 1; 0, 0]$.

Figure 2 and Figure 3 show two instances where the initial set Θ 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 with red ellipsoids. The exact reach set snapshots at every 1 second are shown as green ellipsoids; these are computed using the forward image of an ellipsoid in discrete time based on the linear transformation defined by the system (page 99 of [34]). We also randomly generate simulation traces from the initial set, which are shown as blue traces.

Table 1 shows the degree of conservativeness added by LDFM. We compare the volume of the reach set as computed from [34] and by LDFM, both normalized by the volume of the initial set, and report the average volume of the sampled reach set and the final volume of the reach set. Since the underlying system is linear, the results should be independent from the initial set (the converging or diverging rate of linear systems remains unchanged across the state space). The

discrepancy function computed by LDFM for the system 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 1, we can see that the volume of the exact reach set sampled per 1 second is around 5% of the overapproximation 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.

5.2 Comparison of the Algorithms with Flow*

We implemented a prototype tool in MATLAB based on Algorithms LDF2 and LDFM and tested it on several benchmark verification problems. Simulations are generated using the validated simulation engine CAPD [9], which returns a sequence of timestamped rectangles as required. The optimization problems in Equations (11) and (15) and the SDP problems are solved using SDP3 [43] and Yalmip [36].

We evaluated the algorithm on several nonlinear benchmarks. Van der Pol, Moore-Greitzer, and Brusselator are standard low-dimensional models. The Diode Oscillator from [38] is low dimensional but has complex dynamics described by degree 5 polynomials. Robot Arm is a 4-dimensional model from [3]. Powertrain is the benchmark control system proposed in [32]. This system has highly nonlinear dynamics with polynomials, rational functions, and square roots. Laub-Loomis is a molecular network that produces spontaneous oscillations as a case study for NLTOOLBOX [42]. AS-Polynomial is a 12-dimensional polynomial system [1] that is asymptotically stable around the origin. We also study a 28-dimensional linear model of a helicopter [23].[2] For systems with fewer than three dimensions, we use the vertex matrix constraints method, and for systems with higher dimensions, we use the interval matrix norm method.

Since Algorithm LDF2 is a special case of Algorithm LDFM, as expected, 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 Algorithm LDFM proves safety for the same time horizon. In this experiment, we incorporate the coordination transformation method introduced in [21] to reduce the conservativeness of Algorithm LDF2.

We compare the running time and accuracy of Algorithm LDFM against a leading nonlinear verification tool, Flow* [10], and also against the Algorithm LDF2. As a measure of precision, we compare the ratio of the reach set volume to the initial set volume since tools use different set representations. We calculate two volume ratios: (1) average volume of the reach set divided by the initial volume (sampled at the time steps used in Flow*) and (2) the reach set at the final time point T divided by the initial volume.

The results are shown in Table 2. Consider the performance of Algorithm LDFM as compared to Flow*. From Lines 1 to 3 in Table 2, we see that for simple low-dimensional nonlinear systems, the performance of Flow* is comparable to our algorithm. Lines 4 and 5 and 7 to 9 show that for more complicated nonlinear systems (with higher-order polynomials or higher-order dimensionality), our tool performs much better in running time without sacrificing accuracy. Moreover, from Line 6 and Lines 10 and 11, Algorithm LDFM not only finishes reachtube computation much faster but also provides less conservative results for even more complicated systems (with complicated nonlinear dynamics 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 overapproximation, as

[2]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.

Table 2. Reachtube Experiment Results

					Algorithm LDFM			Flow*			Algorithm LDF2		
	System	Dim	ID	TH(s)	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

Dim: Dimension of the system. **ID**: Initial diameter. **RT**: Running time (includes simulation time for CAPD). **TH**, **A/I VR**, **F/I VR** are defined in Table 1.

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 the interval matrix norm method increases polynomially with the dimension.

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.

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 increased computation times. Results produced by Algorithm LDF2 compare favorably with the verification tool Flow* on the examples with higher dimensions or with complex dynamics.

5.3 Comparison Between Two Matrix Measure Computation Methods

We also report in Table 3 the comparison between the vertex matrix constrains method (Section 4.3.1) and the Interval matrix norm method (Section 4.3.2) used in Algorithm LDFM. We use the 2-dimensional Van der Pol system and the 4-dimensional Powertrain system to illustrate the tradeoff between the two different techniques for computing the optimal bound on the matrix measures of the LDFM algorithm. For lower-dimensional systems like Van der Pol, we can see that although it takes less time, the interval matrix norm method constructs a more conservative reachtube compared with the vertex matrix constraints method. This is because the former can only achieve the locally optimal exponential change rate for the center matrix and has an error term being added to the rate. The Powertrain system has seven nonconstant values in the Jacobian matrix, so there are 129 constraints involved in Equation (11) when applying the vertex matrix constraints method. We observe that for the Powertrain, solving Equation (11) for the vertex matrix constraints method is 10 or more times slower than solving Equation (15) for the interval matrix norm method. It is surprising that the vertex matrix constraints method gives a more conservative reachtube. At a single step the vertex matrix constraints method can give a better (smaller) exponential change rate than the interval matrix norm method; however, conservativeness increases over many steps. This is because the "if" condition at Line 6 of Algorithm LDFM fails more frequently when using the vertex matrix method, and the consequence is that extra conservativeness is introduced by Line 12

Table 3. Comparison Between the Vertex Matrix Constrains (VMC) Method and Interval Matrix Norm (IMN) Method

	LDFM with VMC			LDFM with IMN		
System	RT(s)	A/I VR	F/I VR	RT(s)	A/I VR	F/I VR
Van der Pol	0.69	0.28	4.60e-4	0.55	0.33	0.01
Powertrain	313.2	35.29	2.66	3.97	10.14	0.77

Table 4. Results for the Input-to-State Discrepancy Function

				Algorithm LDFM		
System	Dim	ID	TH(s)	RT(s)	A/I VR	F/I VR
Van der Pol	2	0.20	2	39.25	9.08	24.76
Cardiac Cell	2	0.20	100	13.60	5.95	3.40

in Algorithm LDFM, due to the fact that the updated ellipsoid (with principal axes of different directions and lengths) needs to contain the previously computed ellipsoid. Thus, for systems with fast-changing dynamics, the advantage of obtaining a better exponential change rate gained by the vertex matrix constraints method is diminished. As the dimension of systems increases, solving higher-dimensional SDP problems is more computationally intensive. Frequent shape changing also requires solving more high-dimensional SDP problems for the vertex constraint method, which results in longer running time as compared to the interval matrix norm method.

5.4 Input-to-State Discrepancy

We report the performance of the input-to-state discrepancy function as introduced in Section 4.4 in Table 4. The Van der Pol model is the same model as reported earlier but with input disturbance 0.1 at the right-hand side of the ODE. Cardiac cell is a 2-dimensional system modeling a cardiac cell with a pacemaker from [17]; however, the stimulations from the pacemaker are described by sigmoidal functions (i.e., the up and down slope stimulations are sigmoidal functions) with $0 - 0.5$ offset as uncertainties. We replace Line 13 in Algorithm LDFM by Equation (20) in Lemma 20. The corresponding reachtube for the Van der Pol model, computed using Algorithm LDFM, increases dramatically for a time horizon of 2 seconds, instead of converging as in Table 2. This happens due to the exponential error $e^{\frac{1}{2}t}$ added to the input-to-state discrepancy function. The Cardiac cell model also exhibits a diverging input-to-state discrepancy function, as compared to the exponentially converging discrepancy function without inputs, as shown in [17]. Although the results show that an unacceptable amount of error is introduced, most tools, such as Flow*, do not directly support inputs and uncertainty. Nevertheless, future work will include attempts to mitigate the error introduced by the technique presented in Section 4.4.

6 CONCLUSION

We discussed several techniques to overapproximate reachable states of nonlinear dynamical systems from simulation traces and discrepancy. We propose two methods to compute the upper bounds on the matrix measures, which are used to bloat the simulation traces to obtain the overapproximations. The first algorithm is based on the 2-norm matrix measure and provides a relatively coarse overapproximation but allows fast computations. The second class of algorithms compute locally optimal coordinate transformations such that the local exponential change rate of the discrepancy is minimized, which leads to overapproximations that are less conservative, but takes more time. To our knowledge, this is the first result with such local optimality guarantees. These algorithms can be incorporated in tools like C2E2 [17] for bounded verification of nonlinear switched and hybrid models. Our empirical results show that the approaches perform favorably compared to the Flow* tool on examples with higher dimensions or with complex dynamics. An important direction for future research is to extend these methods to handle models with inputs.

REFERENCES

[1] James Anderson and Antonis Papachristodoulou. 2010. Dynamical system decomposition for efficient, sparse analysis. In *CDC* (2010). IEEE, 6565–6570.

[2] David Angeli. 2002. A Lyapunov approach to incremental stability properties. *IEEE Trans. Automat. Control* 47, 3 (2002), 410–421.
[3] David Angeli, Eduardo D. Sontag, and Yuan Wang. 2000. A characterization of integral input-to-state stability. *IEEE Trans. Automat. Control* 45, 6 (2000), 1082–1097.
[4] Nikos Aréchiga, James Kapinski, Jyotirmoy V. Deshmukh, André Platzer, and Bruce Krogh. 2015. Numerically-aided deductive safety proof for a powertrain control system. *Elect. Notes Theoret. Comput. Sci.* 317 (2015), 19–25.
[5] Nels E. Beckman, Aditya V. Nori, Sriram K. Rajamani, Robert J. Simmons, Sai Deep Tetali, and Aditya V. Thakur. 2010. Proofs from tests. *IEEE Transactions on Software Engineering* 36, 4 (2010), 495–508.
[6] V. A. Boichenko and G. A. Leonov. 1998. Lyapunov's direct method in estimates of topological entropy. *J. Math. Sci.* 91, 6 (1998), 3370–3379.
[7] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. 1994. *Linear Matrix Inequalities in System and Control Theory*. Studies in Applied Mathematics (1994), Vol. 15. SIAM, Philadelphia.
[8] M. Bozzano, A. Cimatti, A. Fernandes Pires, D. Jones, G. Kimberly, T. Petri, R. Robinson, and S. Tonetta. 2015. Formal design and safety analysis of AIR6110 wheel brake system. In *CAV* (2015). Springer, 518–535.
[9] CAPD. 2002. Computer assisted proofs in dynamics. Retrieved from http://www.capd.ii.uj.edu.pl/.
[10] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: An analyzer for non-linear hybrid systems. In *CAV* (2013). Springer, 258–263.
[11] Yi Deng, Akshay Rajhans, and A. Agung Julius. 2013. Strong: A trajectory-based verification toolbox for hybrid systems. In *QEST* (2013). Springer, 165–168.
[12] Charles A. Desoer and Mathukumalli Vidyasagar. 1975. *Feedback Systems: Input-Output Properties*. Vol. 55. SIAM (1975).
[13] Alexandre Donzé. 2015. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV* (2015). Springer, 167–170.
[14] Alexandre Donzé and Oded Maler. 2007. Systematic simulation using sensitivity analysis. In *HSCC* (2007). Springer, 174–189.
[15] Parasara Sridhar Duggirala, Chuchu Fan, Sayan Mitra, and Mahesh Viswanathan. 2015. Meeting a powertrain verification challenge. In *CAV* (2015). Springer, 536–543.
[16] Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. 2013. Verification of annotated models from executions. In *EMSOFT* (2013). IEEE Press, 26:1–26:10.
[17] Parasara Sridhar Duggirala, Sayan Mitra, Mahesh Viswanathan, and Matthew Potok. 2015. C2E2: A verification tool for stateflow models. In *TACAS* (2015). Springer, 68–82.
[18] Parasara Sridhar Duggirala, Le Wang, Sayan Mitra, Mahesh Viswanathan, and César Muñoz. 2014. Temporal precedence checking for switched models and its application to a parallel landing protocol. In *Formal Methods* (2014). Springer, 215–229.
[19] A. El-Guindy, D. Han, and M. Althoff. 2016. Formal analysis of drum-boiler units to maximize the load-following capabilities of power plants. *IEEE Trans. on Power Syst.* (2016), vol. 31, 6, 4691–4702.
[20] Chuchu Fan, James Kapinski, Xiaoqing Jin, and Sayan Mitra. 2016. Locally optimal reach set over-approximation for nonlinear systems. In *EMSOFT* (2016). ACM, 6:1–6:10.
[21] Chuchu Fan and Sayan Mitra. 2015. Bounded verification with on-the-fly discrepancy computation. In *ATVA* (2015). Springer, 446–463.
[22] Raena Farhadsefat, Ji Rohn, and Taher Lotfi. 2011. *Norms of Interval Matrices*. Technical Report No. V-1122 (2011). Institute of Computer Science, Academy of Sciences of the Czech Republic.
[23] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable verification of hybrid systems. In *CAV* (2011), Shaz Qadeer Ganesh Gopalakrishnan (Ed.). Springer.
[24] Antoine Girard, Giordano Pola, and Paulo Tabuada. 2010. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Trans. Automat. Control* 55, 1 (2010), 116–126.
[25] Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations* (3rd ed.). Johns Hopkins University Press (1996), Baltimore, MD.
[26] Ashutosh Gupta, Rupak Majumdar, and Andrey Rybalchenko. 2009. From tests to proofs. In *TACAS* (2009). Springer, 262–276.
[27] Zhenqi Huang, Chuchu Fan, Alexandru Mereacre, Sayan Mitra, and Marta Z. Kwiatkowska. 2014. Invariant verification of nonlinear hybrid automata networks of cardiac cells. In *CAV* (2014). Springer, 373–390.
[28] Zhenqi Huang and Sayan Mitra. 2014. Proofs from simulations and modular annotations. In *HSCC* (2014). ACM Press.
[29] Zhihao Jiang, Miroslav Pajic, Salar Moarref, Rajeev Alur, and Rahul Mangharam. 2012. Modeling and verification of a dual chamber implantable pacemaker. In *TACAS* (2012). Springer, 188–203.
[30] A. Agung Julius, Georgios E. Fainekos, Madhukar Anand, Insup Lee, and George J. Pappas. 2007. Robust test generation and coverage for hybrid systems. In *HSCC* (2007). Springer, 329–342.

[31] A. Agung Julius and George J. Pappas. 2009. Trajectory based verification using local finite-time invariance. In *HSCC* (2009). Springer, 223–236.
[32] James Kapinski, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan, and Nikos Aréchiga. 2014. Simulation-guided lyapunov analysis for hybrid dynamical systems. In *HSCC* (2014). ACM, 133–142.
[33] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. 2015. dReach: δ-reachability analysis for hybrid systems. In *TACAS* (2015). Springer, 200–205.
[34] Alexander Kurzhanskii and István Vályi. 2012. *Ellipsoidal Calculus for Estimation and Control*. Nelson Thornes (1997).
[35] Daniel Liberzon. 2012. *Switching in Systems and Control*. Springer Science & Business Media (2012).
[36] J. Löfberg. 2004. YALMIP: A toolbox for modeling and optimization in MATLAB. In *CACSD* (2004). Retrieved from http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Main.HomePage.
[37] Winfried Lohmiller and Jean-Jacques E. Slotine. 1998. On contraction analysis for non-linear systems. *Automatica* 34, 6 (1998), 683–696.
[38] John Maidens and Murat Arcak. 2015. Reachability analysis of nonlinear systems using matrix measures. *IEEE Trans. Automat. Control* 60, 1 (2015), 265–270.
[39] Ned Nedialkov. 2006. *VNODE-LP: Validated Solutions for Initial Value Problem for ODEs*. Technical Report (2006). McMaster University.
[40] Antonis Papachristodoulou and Stephen Prajna. 2005. Analysis of non-polynomial systems using the sum of squares decomposition. In *Positive Polynomials in Control* (2005). Springer, 23–43.
[41] Eduardo D. Sontag. 2010. Contractive systems with inputs. In *Perspectives in Mathematical System Theory, Control, and Signal Processing* (2010). Springer, 217–228.
[42] Romain Testylier and Thao Dang. 2013. NLTOOLBOX: A library for reachability computation of nonlinear dynamical systems. In *ATVA* (2013). Springer, 469–473.
[43] Reha H. Tütüncü, Kim C. Toh, and Michael J. Todd. 2003. Solving semidefinite-quadratic-linear programs using SDPT3. *Math. Program.* 95, 2 (2003), 189–217.
[44] Mahdi Zamani, Giordano Pola, Manuel Mazo, and Paulo Tabuada. 2012. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Trans. Automat. Control* 57, 7 (2012), 1804–1809.

Received February 2017; revised July 2017; accepted July 2017