On the Verification of Neural ODEs with Stochastic Guarantees

Sophie Gruenbacher[1], **Ramin Hasani**[1,2], **Mathias Lechner**[3], **Jacek Cyranka**[4], **Scott A. Smolka**[5], **Radu Grosu**[1]

[1] Technische Universität Wien (TU Wien)
[2] Massachusetts Institute of Technology (MIT)
[3] Institute of Science and Technology Austria (IST Austria)
[4] University of Warsaw
[5] Stony Brook University
sophie.gruenbacher@tuwien.ac.at, rhasani@mit.edu, mathias.lechner@ist.ac.at, jcyranka@gmail.com, sas@cs.stonybrook.edu, radu.grosu@tuwien.ac.at

Abstract

We show that *Neural* ODEs, an emerging class of time-continuous neural networks, can be verified by solving a set of global-optimization problems. For this purpose, we introduce *Stochastic Lagrangian Reachability* (SLR), an abstraction-based technique for constructing a tight *Reachtube* (an over-approximation of the set of reachable states over a given time-horizon), and provide stochastic guarantees in the form of confidence intervals for the Reachtube bounds. SLR inherently avoids the infamous wrapping effect (accumulation of over-approximation errors) by performing local optimization steps to expand safe regions instead of repeatedly forward-propagating them as is done by deterministic reachability methods. To enable fast local optimizations, we introduce a novel forward-mode adjoint sensitivity method to compute gradients without the need for backpropagation. Finally, we establish asymptotic and non-asymptotic convergence rates for SLR.

Introduction

Neural ordinary differential equations (Neural ODEs) (Chen et al. 2018), which are analogous to a continuous-depth version of deep residual networks (He et al. 2016), exhibit considerable computational efficiency on time-series modeling tasks. Although Neural ODEs do not necessarily improve the performance of contemporary deep models, they enable the rich theory and tools from the field of differential equations to be applied to deep models. Examples include a better characterization of Neural ODEs (Rubanova, Chen, and Duvenaud 2019; Dupont, Doucet, and Teh 2019; Durkan et al. 2019; Jia and Benson 2019), and a better understanding of their robustness (Yan et al. 2020), stability (Yang et al. 2020), and controllability (Quaglino et al. 2019; Holl, Koltun, and Thuerey 2020; Kidger et al. 2020).

As the use of Neural ODEs on real-world applications increases (Finlay et al. 2020; Lechner et al. 2020; Erichson et al. 2020; Lechner and Hasani 2020; Hasani et al. 2020b), so does the importance of ensuring their safety through the use of verification techniques. In this paper, we establish a theoretical foundation for the verification of Neural ODE networks.

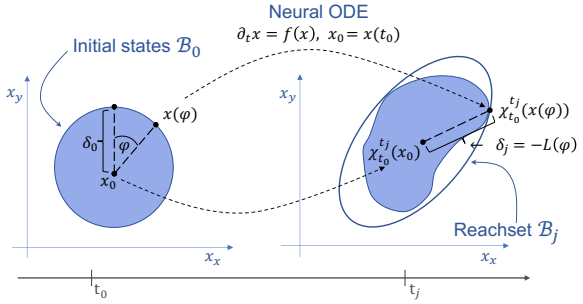

Figure 1: The conservative reachset $\mathcal{B}_j$ at time t_j computed using Lagrangian reachability and global optimization, for a Neural ODE starting from the ball $\mathcal{B}_0$ at time t_0.

In particular, we introduce *Stochastic Lagrangian Reachability* (SLR), a new analysis technique with provable convergence and conservativeness guarantees for Neural ODEs $\partial_t x = f$, with field $f(x, x(0), t, \theta)$, hidden states $x(t)$, and parameters θ. (SLR works in fact for any nonlinear system defined by a set of nonlinear differential equations.)

At the core of SLR is the translation of the reachability problem to a global optimization problem, at every time step t. The latter is solved globally, by uniformly sampling states x from an initial ball $\mathcal{B}_0$, and locally, by computing a local minimum via gradient descent from x. SLR avoids gradient descent if x is within a spherical-cap around a previously sampled state or its corresponding local minimum.

The radius of the cap is derived from the interval computation of the local Lipschitz constant of the objective function within the cap. The minimum computed by SLR at time t stochastically defines an as-tight-as-possible ellipsoid covering all states reached at t by the solution starting in $\mathcal{B}_0$, with tolerance μ and confidence $1-\gamma$, for given values of μ and γ. See Figure 1.

Since SLR employs interval arithmetic only locally to compute the spherical-caps (also called safety or tabu regions), it avoids the infamous wrapping effect (Lohner 1992) of deterministic reachability methods (see Table 1), which

prevents them from being deployed in practice. Consequently, our approach scales up to large-scale, real-life Neural ODEs. To the best of our knowledge, none of the available tools has been successfully applied to Neural ODEs.

We also introduce a novel forward formulation of the adjoint sensitivity method (Pontryagin 2018) to compute the loss gradients in the optimization flow. This enables us to improve the time complexity of the optimization process compared to similar methods (Chen et al. 2018; Zhuang et al. 2020).

Summary of results. In this work, we present a thorough theoretical approach to the problem of providing safety guarantees for the class of time-continuous neural networks formulated as Neural ODEs. As the main result, we develop SLR, a differentiable stochastic Lagrangian reachability framework, formulated as a global optimization problem. In particular, we prove that SLR converges (Theorem 2) to tight ellipsoidal safe regions (Theorem 1), within $\mathcal{O}(-\ln\gamma(\delta_0/r_{bound})^{2n})$ number of iterations (Theorem 3). This implies that for a given confidence level γ, our algorithm terminates according to the proposed rate, which leads to the important conclusion that the problem of constructing an ellipsoid abstraction of the true reachsets with probabilistic guarantees for Neural ODEs is decidable (the computed abstraction is conservative with confidence γ). We summarize our key contributions as follows:

- We introduce a theoretical framework for the verification of Neural ODEs by restating the reachability problem as a set of global-optimization problems.
- We solve each optimization problem globally, via uniform sampling, and locally, through gradient descent (GD), thereby avoiding costly Hessian computations in the process.
- GD is avoided in spherical-caps around the start/end states of previous searches. The cap radius is derived from its local Lipschitz constant, computed via interval arithmetic.
- We design a forward-mode GD algorithm based on the adjoint sensitivity method for (Neural) ODEs.
- We prove convergence properties of SLR, its safety guarantees, and discuss its time and space complexity.

Related Work

Global optimization. The literature on global optimization for continuous problems is vast and includes many different approaches depending on the smoothness assumptions made about the objective function. Evolutionary strategies like those based on the covariance matrix (Hansen and Ostermeier 2001; Igel, Hansen, and Roth 2007) work for general continuous objectives. Deterministic interval-based branch-and-bound methods (Neumaier 2004; Hansen 1980) work for differentiable objectives, and Lipschitz global optimization (Piyavskii 1972; Shubert 1972; Malherbe and Vayatis 2017) for objectives satisfying the Lipschitz condition. Our work is closest to the BRST algorithm (Boender et al. 1982; Rinnooy Kan and Timmer 1987a,b) which for smooth objectives uses Hessians to compute the basins of attraction for local minima as ellipsoidal bounds. Such basins define tabu regions. The final estimate for the global minimum and reasonable confidence bounds are provided.

Stochastic reachability. Existing work is mainly concerned with the verification of safety guarantees for stochastic hybrid systems with continuous dynamics (ODEs) in each mode. Stochasticity is introduced in several ways: uncertainty in the model parameters (Wang et al. 2015; Fränzle, Teige, and Eggers 2010; Shmarov and Zuliani 2015b), uncertainty in the discrete jumps between modes (Fränzle et al. 2011), and uncertainty in the initial state (Huang et al. 2017). The work of (Enszer and Stadtherr 2011) focuses on the probabilistic verification of continuous-time ODEs with uncertainty in parameters and initial states.

Reachability for continuous dynamical systems. Most of the relevant techniques are deterministic and based on interval arithmetic. We provide a qualitative summary of existing reachability methods for continuous-time systems in Table 1.

Setup

In this section, we introduce our notation, preliminary concepts, and definitions required to construct our theoretical setup for the verification of Neural ODEs.

Neural ODE. The derivative of the hidden states x is computed by a neural network f parameterized by θ as follows (Chen et al. 2018):

$$\partial_t x = f(x, x(0), t, \theta), x_0 \in \mathcal{B}_0 \tag{1}$$

We require that the Neural ODE is Lipschitz-continuous and forward-complete. The solution to this initial-value problem can be computed by numerical ODE solvers, from any initial system state $x(0) = x_0$. Consequently, the numerical solution can be trained by reverse-mode automatic differentiation (Rumelhart, Hinton, and Williams 1986), either through the solver, by a vanilla backpropagation algorithm (Hasani et al. 2020a), or by treating the solver as a blackbox and using the adjoint sensitivity method (Pontryagin 2018).

Geometrical deformation in time by a flow χ. To describe the optimization problem, we use Eulerian and Lagrangian coordinates from classical continuum mechanics. We regard the set of initial states, which is the ball $\mathcal{B}_0 = B(x_0, \delta_0)$, as a body that is being deformed in time by a flow χ. Given a point $x \in \mathcal{B}_0$ in Eulerian coordinates (the undeformed configuration), there is at every time $t_j > t_0$ the representation $x(t_j) = \chi_{t_0}^{t_j}(x)$ of that point in Lagrangian coordinates (the configuration deformed by χ).

The deformation of $\mathcal{B}_0$ in time is related to the Neural ODE, where χ is defined as the solution flow of Eq. (1).

Reachset. A reachset is the set of all states reached at a target time t, given the initial states and a flow. More formally:

Definition 1. *Given a set of initial states $\mathcal{B}_0$ at time t_0, the target time $t_j \geq t_0$, and the flow χ of the Neural ODE* (1)*, we call $\mathcal{B}_j(\mathcal{B}_0) \subset \mathbb{R}^n$ a conservative* reachset *enclosure if $\chi_{t_0}^{t_j}(x) \in \mathcal{B}_j(\mathcal{B}_0)$, for all $x \in \mathcal{B}_0$; i.e., the reachset bounds all state-trajectories of the Neural ODE.*

Whenever the initial set $\mathcal{B}_0$ is known from the context, we simply refer to the reachset as *the Reachset at time t_j*, or $\mathcal{B}_j$.

Technique	Determ.	Parallelizable (single step)	Basis	wrapping effect
LRT (Cyranka et al. 2017)	yes	no	Infinitesimal strain theory	yes
CAPD (Kapela et al. 2020)	yes	no	Lohner algorithm	yes
Flow-star (Chen et al. 2013)	yes	no	Taylor models	yes
δ-reachability (Gao et al. 2013)	yes	no	approximate satisfiability	yes
C2E2 (Duggirala et al. 2015)	yes	no	discrepancy function	yes
LDFM (Fan et al. 2017)	yes	yes	simulation, matrix measures	no
TIRA (Meyer et al. 2019)	yes	yes	second-order sensitivity	no
Isabelle/HOL (Immler 2015)	yes	no	proof-assistant	yes
Breach (Donzé et al. 2007)	yes	yes	simulation, sensitivity	no
PIRK (Devonport et al. 2020)	yes	yes	simulation, contraction bounds	no
HR (Li et al. 2020)	yes	no	hybridization	yes
ProbReach (Shmarov et al. 2015a)	no	no	δ-reachability, probability interval	yes
VSPODE (Enszer et al. 2011)	no	no	p-boxes	yes
GP (Bortolussi et al. 2014)	no	no	Gaussian process	no
SLR **Ours**	no	yes	stochastic Lagrangian reachability	no

Table 1: A Perspective on Related Work. Deterministic (Determ.) refers to approaches that provide an overapproximation of the reach-set without any uncertainties. A "No" in the deterministic column indicates a stochastic approach that yields a reach-set with a corresponding confidence interval.

Reachtube. A reachtube is a series of reachsets within a determined time-horizon. Formally:

Definition 2. *Given a set of initial states $\mathcal{B}_0$ at time t_0, and a time horizon T, we use $B(\mathcal{B}_0, T)$ to denote a sequence of time-stamped reachsets $\mathcal{B}_1, \ldots, \mathcal{B}_k$ with $t_0 \leq t_1 \leq \ldots \leq t_k = T$.*

Whenever the initial set, time horizon, and flow are known from the context, we use the term *reachtube overapproximation* or $\mathcal{B}$, for that sequence of reachsets.

Definition 3 (Ellipsoid)**.** *Given $A_j, M_j \in \mathbb{R}^{n\times n}$, $M_j \succ 0$ with $A_j^T A_j = M_j$ and $\|x\|_{M_j} = \sqrt{x^T M_j x}$, we call $B_{M_j}(x_0, \delta)$ a ball in metric M_j (or an ellipsoid) with center x_0 and radius δ if $\|x - x_0\|_{M_j} \leq \delta$ for all $x \in B_{M_j}(x_0, \delta)$.*

Reachability as an optimization problem. Given a time horizon T, an initial ball $\mathcal{B}_0 = B_I(x_0, \delta_0)$ with center x_0 and radius δ_0, and Euclidean metric $M_0 = I$, our goal is to find a tight reachtube $\mathcal{B}$, bounding all state-trajectories of the Neural ODE (1).

We capture the reachsets of $\mathcal{B}$ by ellipsoids $\mathcal{B}_j = B_{M_j}(\chi_{t_0}^{t_j}(x_0), \delta_j)$ with center $\chi_{t_0}^{t_j}(x_0)$, radius δ_j, and metric M_j. At every time t_j, we use as the center $\chi_{t_0}^{t_j}(x_0)$, the numerical integration of x_0, and as the metric M_j, the optimal metric in $\chi_{t_0}^{t_j}(x_0)$ minimizing the volume of the ellipsoid, as proposed in (Gruenbacher et al. 2020).

Thus, our goal is to find at every time step t_j, a radius δ_j which (stochastically) guarantees that $\mathcal{B}_j$ is a conservative reachset. I.e., at each t_j, we want to find the maximal distance of all $\chi_{t_0}^{t_j}(x)$ to center $\chi_{t_0}^{t_j}(x_0)$ in metric M_j for $x \in \mathcal{B}_0$, and define δ_j as this distance. Thus the optimization problem can be defined as follows:

$$\delta_j \geq \max_{x\in\mathcal{B}_0} \left\| \chi_{t_0}^{t_j}(x) - \chi_{t_0}^{t_j}(x_0) \right\|_{M_j} = \max_{x\in\mathcal{B}_0} \operatorname{dist}\left(\chi_{t_0}^{t_j}(x)\right) \tag{2}$$

where we use $\operatorname{dist}(\chi_{t_0}^{t_j}(x))$ to describe the distance in Eq. (2) when metric M_j and starting point x_0 are known.

As we require Lipschitz-continuity and forward-completeness, the map $x \mapsto \chi_{t_0}^{t_j}(x)$ is a homeomorphism and commutes with closure and interior operators. In particular, the image of the boundary of the set $\mathcal{B}_0$ is equal to the boundary of the image $\chi_{t_0}^{t_j}(\mathcal{B}_0)$. Thus, Eq. (2) has its optimum on the surface of the initial ball $\mathcal{B}_0^S = \text{surface}(\mathcal{B}_0)$, and we will only consider points on the surface. In order to be able to optimize this problem, we describe the points on the surface with (n-dimensional) polar coordinates such that every point $x \in \mathcal{B}_0^S$ is represented by a tuple (δ_0, φ), with angles $\varphi = (\varphi_1, \ldots, \varphi_{n-1})$ and center x_0, having a conversion function $x((\delta_0, \varphi), x_0)$ from polar to Cartesian coordinates, defined as follows:

$$x((\delta_0, \varphi), x_0) = \begin{pmatrix} x_{0,1} + \delta_0 \cos(\varphi_1) \\ \vdots \\ x_{0,n-1} + \delta_0 \sin(\varphi_1) \cdot \ldots \cdot \sin(\varphi_{n-2}) \cos(\varphi_{n-1}) \\ x_{0,n} + \delta_0 \sin(\varphi_1) \cdot \ldots \cdot \sin(\varphi_{n-2}) \sin(\varphi_{n-1}) \end{pmatrix} \tag{3}$$

Whenever the center x_0 and the radius δ_0 of the initial ball $\mathcal{B}_0$ are known from the context, we will use the following notation: $x(\varphi)$ for the conversion from polar to Cartesian coordinates and $\varphi(x)$ for Cartesian to polar. Using polar coordinates, we restate the optimization problem (2) as follows:

$$\begin{aligned} \delta_j &= \max_{x\in\mathcal{B}_0} \left\| \chi_{t_0}^{t_j}(x) - \chi_{t_0}^{t_j}(x_0) \right\|_{M_j} \\ &= \max_{\varphi\in\mathbb{R}^{n-1}} \underbrace{\left\| \chi_{t_0}^{t_j}(x(\varphi)) - \chi_{t_0}^{t_j}(x_0) \right\|_{M_j}}_{=-L(\varphi)} \\ &= \min_{\varphi\in\mathbb{R}^{n-1}} L(\varphi) = m^\star, \end{aligned} \tag{4}$$

We call L the *loss function* in polar coordinates at time t_j that we would like to minimize. Note that L also depends on the initial radius δ_0 and initial center x_0; as these are fixed inputs, we do not consider them in the notation.

Algorithm 1 Finding the local minimum

Require: target time t_j, termination tolerance $\epsilon > 0$, learning rate $\gamma > 0$, initial guess $\varphi \in \mathbb{R}^{n-1}$, loss function L, gradient of loss $\nabla_\varphi L$

1: $l \leftarrow L(\varphi)$, $l_{prev} \leftarrow \infty$
2: **while** $|l - l_{prev}|/|l_{prev}| > \epsilon$ **do**
3: **compute** $\nabla_\varphi L$
4: $\varphi \leftarrow \varphi - \alpha \nabla_\varphi L$
5: $l_{prev} \leftarrow l$
6: $l \leftarrow L(\varphi)$
7: **end while**
8: **return** φ, l

Main Results

In this section, we present our verification framework for Neural ODEs, which we call **Stochastic Lagrangian Reachability (SLR)**. As the main results of this paper, we show that the algorithm guarantees safety and converges to the tightest ellipsoid, almost surely, in the limit of the number of samples. We then compute the convergence rate and discuss space and time complexities.

Gradient Computation

Our algorithm uses gradient descent locally when solving the global optimization problem of Eq. (4). Gradient descent is started from uniformly sampled points, which are not contained in already constructed safety regions.

Uniform sampling is used to repeatedly select an initial point from the surface of the ball $\mathcal{B}_0$. Gradient descent is used from this point to find a local minimum. SLR is inspired by the *gradient-only tabu-search* (GOTS) proposed in (Stepanenko 2009). Instead of tabu regions, we use *safety radii* $r(\varphi)$ to construct an area around already visited points φ, where we know for sure what the minimum value inside that region is. In the following, we describe the computational steps of the loss's gradient for the main SLR algorithm in greater detail.

Given the target time t_j, termination tolerance $\epsilon > 0$, learning rate $\gamma > 0$, initial guess $\varphi \in \mathbb{R}^{n-1}$, and loss function L, we seek to compute the gradient of loss $\nabla_\varphi L$. We introduce a new framework to compute the loss's gradient which is needed in Line 3 of Algorithm 1 to find the local minimum. Using the chain rule, we can express the gradient $\nabla_\varphi L$ as follows:

$$\begin{aligned}\frac{\partial L(\cdot)}{\partial \varphi}(\varphi) &= -\frac{\partial \operatorname{dist} \circ \chi_{t_0}^{t_j} \circ x(\cdot)}{\partial \varphi} \\ &= -\underbrace{\frac{\partial \operatorname{dist}}{\partial y}\bigg|_{y=\chi_{t_0}^{t_j}(x(\varphi))}}_{(a)} \cdot \underbrace{\frac{\partial \chi_{t_0}^{t_j}}{\partial x}\bigg|_{x=x(\varphi)}}_{(c)} \cdot \underbrace{\frac{\partial x(\cdot)}{\partial \varphi}}_{(b)}\end{aligned} \tag{5}$$

Part (a) - loss gradient wrt y: The differentiation of the loss function defined in Eq. (2) can be expressed as

$$\partial_y \operatorname{dist}(y) = A_j(y - \chi_{t_0}^{t_j}(x_0)) \operatorname{dist}(y)^{-1} A_j, \tag{6}$$

Algorithm 2 Computation of $\nabla_\varphi L$

Require: target time t_j, initial value $\varphi \in \mathbb{R}^{n-1}$, Neural ODE f, gradients $\partial_x \operatorname{dist}$ and $\partial_\varphi x$

1: $b \leftarrow x(\varphi), F \leftarrow I$
2: $[b, F] \leftarrow \text{solve\_ivp}([f(b,t), (\partial_b f)(b) \cdot F], [0, t_j], [b, F])$
3: $\nabla_\varphi L \leftarrow -\partial_y \operatorname{dist}(y) \cdot F \cdot \partial_\varphi x$
4: **return** $\nabla_\varphi L$ {Required in line 3 of algorithm 1}

with A_j from Def. 3 and M_j as the metric in $\chi_{t_0}^{t_j}(x_0)$ minimizing the volume of the ellipsoid (Gruenbacher et al. 2020).

Part (b) - polar gradient: $x(\varphi)$ describes the transformation from polar coordinates to Cartesian coordinates, as given in Eq. (3). The differentiation with respect to φ is straightforward to obtain using the product rule and the derivatives of $\sin$ and $\cos$:

$$\partial_\varphi x(\varphi) = \begin{pmatrix} -\delta_0 \sin(\varphi_1) \\ \delta_0 \left(\cos(\varphi_1)\cos(\varphi_2) - \sin(\varphi_1)\sin(\varphi_2)\right) \\ \vdots \end{pmatrix} \tag{7}$$

Part (c) - gradient of the flow: The partial derivative $\partial_x \chi_{t_0}^{t_j}(x)$ in x of the Neural ODE solution flow χ with respect to the initial condition is called the gradient of the flow or *deformation gradient* in (Slaughter 2002; Abeyaratne 1998), and the *sensitivity matrix* in (Donzé 2010; Donzé and Maler 2007). Let I be the identity matrix in $\mathbb{R}^{n\times n}$. As we now show, the sensitivity matrix $\partial_x \chi_{t_0}^{t_j}(x)$ is a solution of the *variational equations* associated with (1):

$$\begin{aligned} \partial_x \chi_{t_0}^{t_j}(x) &= F(t_j, x) \\ \partial_t F(t,x) &= (\partial_x f)(\chi_{t_0}^t(x)) F(t,x), \quad F(t_0,x) = I \end{aligned} \tag{8}$$

Proof sketch: By interchanging the differentiation order, we obtain $\partial_t(\partial_x \chi_{t_0}^t(x)) = \partial_x(\partial_t \chi_{t_0}^t(x))$. Since $\chi_{t_0}^t(x)$ is a solution of Eq. (1), $\partial_x(\partial_t \chi_{t_0}^t(x)) = \partial_x(f(\chi_{t_0}^t(x)))$. By the chain rule, we get $\partial_t(\partial_x \chi_{t_0}^t(x)) = (\partial_x f)(\chi_{t_0}^t(x)) \partial_x \chi_{t_0}^t(x)$.

Forward-mode use of adjoint sensitivity method. The integral of Eq. (8) has the same form of the auxiliary ODE used for reverse-mode automatic differentiation of Neural ODEs, when optimized by the adjoint sensitivity method (Chen et al. 2018) with one exception. In contrast to (Chen et al. 2018), which requires one to run the adjoint equation backward and have access to the termination time of the flow, our approach enjoys a simultaneous forward-mode use of the adjoint equation. This is due to the way we determine the loss function in the ODE space. In retrospect, this enables us to obtain the gradients of the loss at the current state-computation step. This property enables us to improve the optimization runtime by 50%, compared to the optimization scheme used in (Chen et al. 2018): we save half of the time because we do not have to go backward to compute the loss.

More precisely, solving Eq. (8) until target time t_j requires knowledge of $\chi_{t_0}^t(x)$ for all $t \in [t_0, t_j]$. This ensures that we already know the value of $\chi_{t_0}^t(x_0)$ when needed

to compute the right side of Eq. (8) during integration of $F(t,x)$. Algorithm 2 demonstrates the computation of the gradient $\nabla_\varphi L$ of the loss function.

Safety-Region Computation

With our global search strategy, we are covering the feasible region $\mathcal{B}_0^S$ with already visited points $\mathcal{V}$. Consequently, we have access to the global minimum in all of those regions:

$$\bar{m} = \min_{\varphi \in \mathcal{V}} L(\varphi) \tag{9}$$

with $\bar{m} \geq m^\star$, where $m^\star$ is the global minimum of Eq. (4). We now identify safety regions for a Neural ODE flow and describe how this is incorporated in the SLR algorithm.

Definition 4 (Safety Region)**.** *Let $\varphi_i \in \mathcal{V} \subseteq \mathbb{R}^{n-1}$ be an already-visited point. A safety-radius $r_{\varphi_i} = r(\varphi_i)$ defines a* safe spherical-cap $B(\varphi_i, r_{\varphi_i})^S = B(x(\varphi_i), r_{\varphi_i}) \cap \mathcal{B}_0^S$*, because $L(\psi) \geq \mu \cdot \bar{m}$ for all ψ s.t. $x(\psi) \in B(\varphi_i, r)^S$.*

Our objective is to use the local Lipschitz constants to define a radius r_φ around an already visited point φ s.t. we can guarantee that $B(\varphi, r_\varphi)^S$ is a safety region.

Definition 5 (Lipschitz)**.** *The local Lipschitz constant (LLC) of a function L in a region A is defined as a $\lambda_A \geq 0$ with*

$$\|L(x) - L(y)\| \leq \lambda_A \|x - y\| \quad \forall x, y \in A.$$

In the following theorem, we use the LLC to define the radius r_φ of the safety (or tabu) region $B(\varphi, r_\varphi)^S$ around an already-visited point $\varphi \in \mathcal{V}$.

Theorem 1 (Radius of Safety Region)**.** *At target time t_j, let $\bar{m}$ be the current global minimum, as in Eq.* (9)*. Let $\varphi \in \mathcal{V}$ be an already-visited point with value $L(\varphi)$ ($\geq \bar{m}$), and let r_φ and $B(\varphi, r_\varphi)^S$ be defined as follows with $\mu \geq 1$:*

$$r_\varphi = \lambda_{\Sigma_\varphi}^{-1} \left(L(\varphi) - \mu \cdot \bar{m} \right) \tag{10}$$

with $\lambda_{\Sigma_\varphi} = \max_{x(\psi) \in \Sigma_\varphi} \|\partial_x \chi_{t_0}^{t_j}(x(\psi))\|_{M_{0,j}}$. If Σ_φ is chosen s.t. $\Sigma_\varphi \supseteq B(\varphi, r_\varphi)^S$, then it holds that:

$$L(\psi) \geq \mu \cdot \bar{m} \quad \forall x(\psi) \in B(\varphi, r_\varphi)^S \tag{11}$$

The full proof is provided in the Appendix. *Proof sketch:* The Lipschitz constant defines a relation between the values in the domain and the ones in the range of the function.

Theorem 1 says that areas around already-visited samples are safe. The size of the safety areas increases if we have a better current global minimum. Therefore, the theorem demonstrates that we can improve the convergence rate if we optimize the loss, by possibly finding a better current global minimum. This justifies the use of gradient descent together with a more global search strategy.

Algorithm 3 computes the radius in Eq. (10) as a fixpoint of the choice of Σ_φ. For an over-approximation of the LLC in Line 2, we use the triangle inequality and the mean value inequality with a change in metric (Cyranka et al. 2017, Lemma 2). We then solve Eq. 8 using interval arithmetic to obtain an interval gradient matrix $[\mathcal{F}_t] \ni \partial_x \chi_{t_0}^{t_j}(x)$ $\forall\, x \in \Sigma_\varphi$, and take the maximum singular value of $[\mathcal{F}_t]$, as proposed in (Gruenbacher et al. 2019). Depending on the

Algorithm 3 Computing the Radius of the Safety Region

Require: target time t_j, visited point φ, termination tolerance $\epsilon > 0$, initial ball $\mathcal{B}_0$ with radius δ_0, minimum of visited points $\bar{m}$, loss function L, tolerance $\mu \geq 1$, region Σ_φ in which to compute the LLC λ.

1: $\Sigma_\varphi \leftarrow \mathcal{B}_0, s \leftarrow \delta_0$
2: $\lambda \leftarrow$ computeLipschitz(Σ_φ)
3: $r \leftarrow 1/\lambda \cdot (L(\varphi) - \mu \cdot \bar{m})$
4: **while** $|r - s|/r > \epsilon$ **or** $s < r$ **do**
5: **set** $s \leftarrow r + |s - r|/2$
6: $\Sigma_\varphi \leftarrow B(\varphi, s)^S$
7: $\lambda \leftarrow$ computeLipschitz(Σ_φ)
8: $r \leftarrow 1/\lambda \cdot (L(\varphi) - \mu \cdot \bar{m})$
9: **end while**
10: **return** r

Neural ODE, it is presumably faster to pick $s = \delta_0$, and to always use the LLC $\lambda_{\mathcal{B}_0}$ of the entire initial ball. As a result of the way we select r_φ in Theorem 1, we are able to increase the radii r_φ as soon as a new region with a smaller local minimum than the previous ones is discovered. Thus: $\bar{m} \leq \bar{m}_{prev} \Rightarrow L(\varphi) - \mu \cdot \bar{m} \geq L(\varphi) - \mu \bar{m}_{prev} \Rightarrow r_\varphi \geq r_{\varphi,prev}$.

Stochastic Lagrangian Reachability

By using local gradient computation, global uniform sampling, and safety regions as in Algorithm 3, we present our SLR verification technique, as outlined in Algorithm 4.

Given a Neural ODE as in Eq. (1) and a set of initial states $\mathcal{B}_0$, we start by specifying a confidence level $\gamma \in (0,1)$ and a tolerance $\mu, \geq 1$ for the entire Reachtube. The algorithm returns radii δ_j, $j \in \{1, \ldots, k\}$, and the stochastic guarantee stating that $\mathcal{B}_j (= B_{M_j}(\chi_{t_0}^{t_j}, \delta_j))$ overestimates by μ the true conservative Reachsets with a probability higher than $1 - \gamma$. This holds also for the whole Reachtube, as it is defined by a series of Reachsets (Def. 2).

As we reinitialize the variables at the beginning of every new timestep t_j, and apply gradient descent to the loss function of the initial polar coordinates φ at time t_0, we do not accumulate errors from one timestep to the next one. This is a prominent advantage compared to methods using interval arithmetic, and thus accumulating the wrapping effect, e.g. (Zgliczynski 2002; Cyranka et al. 2018; Fan et al. 2017). Another advantage is that we can compute the for-loop in line 1 of Algorithm 4 (thus the Reachsets of the Reachtube) in parallel.

At every timestep t_j, we sample random points and construct safety regions around them until we reach the desired probability $1 - \gamma$ of being inside the tolerance region defined by μ. After sampling a new point, we check if this point is already in the covered area. If not, then we apply gradient descent to find a local minimum and compare this local minimum to the smallest value $\bar{m}$. Otherwise, if the sampled point is already in the covered area and thus in at least one safety region, we already know the lower bounds for that region and do not look for the local minimum again.

Algorithm 4 Stochastic Lagrangian Reachability

Require: time horizon T, sequence of timesteps t_j ($t_0 \leq t_1 \leq \cdots \leq t_k = T$), tolerance $\mu \geq 1$, confidence level $\gamma \in (0,1)$, loss function L, gradient of loss $\nabla_\varphi L$

1: **for** $(j = 1; j \leq k; j = j + 1)$ **do**
2: $\mathcal{V}, \mathcal{U} \leftarrow \{\}$ (list of visited and random points)
3: $\mathcal{S} \leftarrow \{\}$ (total covered area)
4: $\bar{p} \leftarrow 0, \bar{m} \leftarrow 0$
5: **while** $\bar{p} < 1 - \gamma$ **do**
6: **sample** $\varphi \in \mathbb{R}^{n-1}$
7: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\varphi\}$
8: $\mathcal{U} \leftarrow \mathcal{U} \cup \{\varphi\}$
9: **if** $x(\varphi) \notin \mathcal{S}$ **then**
10: $\varphi_{min} \leftarrow$ local minimum starting at φ using gradient descent with $\nabla_\varphi L$
11: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\varphi_{min}\}$
12: $m \leftarrow L(\varphi_{min})$
13: **else**
14: $m \leftarrow L(\varphi)$
15: **end if**
16: **if** $m \leq \bar{m}$ **then**
17: $\bar{m} \leftarrow m$
18: **set** $S \leftarrow \{\}$
19: **for all** $\varphi_i \in \mathcal{V}$ **do**
20: **compute** new radius $r = r(\varphi_i)$ such that $L(\psi) \geq \mu \cdot \bar{m}, \quad \forall \psi \colon x(\psi) \in B(\varphi_i, r)^S$
21: **set** $\mathcal{S} \leftarrow \mathcal{S} \cup B(\varphi_i, r)^S$
22: **end for**
23: **else**
24: **compute** radius $r = r(\varphi)$ only for current φ such that $L(\psi) \geq \mu \cdot \bar{m}, \quad \forall \psi \colon x(\psi) \in B(\varphi, r)^S$
25: **set** $\mathcal{S} \leftarrow \mathcal{S} \cup B(\varphi, r)$
26: **end if**
27: $\bar{p} \leftarrow \Pr(\mu \cdot \bar{m} \leq m^\star)$ with $\mu \cdot \bar{m} \leq \min_{\varphi \in \mathcal{S}} L(\varphi)$
28: **end while**
29: $\delta_j \leftarrow -\bar{m}$
30: **end for**
31: **return** $(\delta_1, \ldots, \delta_k)$

This approach is similar to using baisins of attraction, but is more scalable because we do not require Hessian computation. In line 20, we recompute the radii of the safety regions when we find a new smallest value $\bar{m}$. By computing the current probability $\bar{p}$ of having reached the desired confidence level, we check whether we have to resample more points or whether we are able to finish that timestep and save the radius δ_j of the stochastic Reachset at time t_j.

Stochastic Guarantees of Reachsets

In this section, we derive the stochastic convergence guarantees and convergence bounds for finding the global minimum of Eq. (4) using SLR at every timestep t_j.

Let $\bar{m} = \min_{\varphi \in \mathcal{V}} L(\varphi)$ be defined as in Eq. (9), and let $m^\star = \min_{\varphi \in \mathbb{R}^{n-1}} L(\varphi)$ be the global minimum and $\varphi^\star$ an argument s.t. $L(\varphi^\star) = m^\star$. We start by defining the probability of $B(\varphi, r_\varphi)^S$ covering $x(\varphi^\star)$:

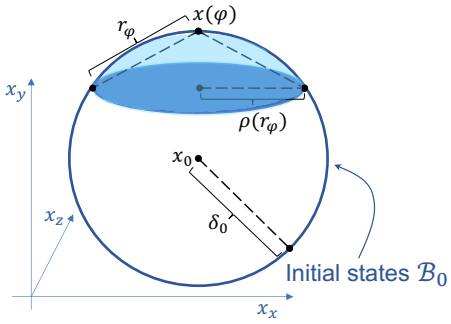

Figure 2: Illustration of a safety region $B(\varphi, r_\varphi)^S$, which is a spherical cap $\mathcal{C}(r_\varphi)$. In this figure, the area of cap $\mathcal{C}(r_\varphi)$ (in light blue) is greater than the volume of an $n-1$-dimensional ball (in dark blue) with radius $\rho(r_\varphi)$, which is used in the convergence rate.

$$\begin{aligned} &\Pr(B(\varphi, r_\varphi)^S \ni x(\varphi^\star)) \\ &= \Pr\left(\|x(\varphi^\star) - x(\varphi)\|_2) \leq r_\varphi\right) \\ &= \Pr(x(\varphi) \in \mathcal{C}(r_\varphi)) = \Pr(\mathcal{C}(r_\varphi)) \end{aligned} \tag{12}$$

with r_φ as defined in Eq. (10) and $\mathcal{C}(r_\varphi) = B(\varphi^\star, r_\varphi)^S$ being the spherical-cap in Fig. 2. By using the area of the spherical cap $\mathcal{C}(r_\varphi)$ and the area of the initial ball's surface $\mathcal{B}_0^S$, the probability defined by Eq. (12) can be described as follows:

$$\Pr(\mathcal{C}(r_\varphi)) = \frac{\text{Area}(\mathcal{C}(r_\varphi))}{\text{Area}(\mathcal{B}_0^S)} \tag{13}$$

The area of $\mathcal{C}(r_\varphi)$ can be computed using the formulas in (Shengqiao 2011). Next we derive some probabilities:

$$\begin{aligned} \Pr(B(\varphi_j, r_{\varphi_j})^S \not\ni \varphi^\star) &= 1 - \Pr(\mathcal{C}(r_{\varphi_j})) \\ \Pr(\forall \varphi \in \mathcal{U} \colon B(\varphi, r_\varphi)^S \not\ni \varphi^\star) &= \prod_{\varphi \in \mathcal{U}} (1 - \Pr(\mathcal{C}(r_\varphi))) \\ \Pr(\exists \varphi \in \mathcal{U} \colon B(\varphi, r_\varphi)^S \ni \varphi^\star) &= 1 - \prod_{\varphi \in \mathcal{U}} (1 - \Pr(\mathcal{C}(r_\varphi))) \end{aligned} \tag{14}$$

Using Theorem 1, if $\varphi^\star \in B(\varphi, r_\varphi)^S$ for some $\varphi \in \mathcal{U}$, then $\mu \cdot \bar{m} \leq L(\varphi^\star) = m^\star$ holds, and thus:

$$\begin{aligned} &\Pr(\mu \cdot \bar{m} \leq m^\star) \geq \\ &\Pr(\exists \varphi \in \mathcal{U} \colon B(\varphi, r_\varphi)^S \ni \varphi^\star) \end{aligned} \tag{15}$$

Theorem 2 (Convergence Guarantees)**.** *Given $\gamma \in (0,1)$, $\mu \geq 1$, local Lipschitz constant $\lambda_{\mathcal{B}_0^S}$ and $N = |\mathcal{U}|$, where $\mathcal{U}$ is the set of uniform-randomly generated points during global search process. Let $\bar{m} = \min_{\varphi \in \mathcal{V}} L(\varphi)$ as defined in Eq.* (9)*, $m^\star = \min_{\varphi \in \mathbb{R}^{n-1}} L(\varphi)$ the global minimum, and $\varphi^\star$ an argument s.t. $L(\varphi^\star) = m^\star$. Then:*

$$\lim_{N \to \infty} \Pr(\mu \cdot \bar{m}_N \leq m^\star) = 1 \tag{16}$$

and thus

$$\forall \gamma \in (0,1), \exists N \in \mathbb{N} \text{ s.t. } \Pr(\mu \cdot \bar{m}_N \leq m^\star) \geq 1 - \gamma \tag{17}$$

The full proof is provided in the Appendix. *Proof sketch:* By creating a lower bound r_{bound} for all r_φ, s.t. $\Pr(\mathcal{C}(r_\varphi) \geq \Pr(\mathcal{C}(r_{bound}))$, we underestimate Eq. (14) by $1 - (1 - \Pr(\mathcal{C}(r_{bound})))^N$. Using this bound and Eq. (15), we show that the convergence guarantee holds.

Theorem 2 shows that in the limit of the number of samples, the reachset constructed by Algorithm 4 converges with probability 1 to the smallest ellipsoid that encloses the true reachable set. Note that the algorithm cannot converge to the true reachable set because we approximate the reachset by ellipsoids, while the true reachset might be of arbitrary geometrical shape. Nonetheless, we proved that it provides the smallest possible ellipsoid that contains a true reachset.

Moreover, although Theorem 2 shows that we achieve the tightest elliptical reachsets, it does not determine whether the algorithm can terminate or not, as the theorem is proven in the case of infinite samples. We now prove that SLR indeed converges at a reasonable rate.

Convergence Rate for SLR

Theorem 3 computes a convergence rate for Algorithm 4.

Theorem 3 (Convergence Rate)**.** *Given $\gamma \in (0, 1)$, $\mu \geq 1$, local Lipschitz constant $\lambda_{\mathcal{B}_0^S}$, and dimension n, let φ_1 be the first random sample point. We can guarantee that $\Pr(\mu \cdot \bar{m} \leq m^\star) \geq 1 - \gamma$ if we perform at most N_{max} iterations of the SLR Algorithm 4, with*

$$N_{max} = \ln \gamma \Bigg/ \ln \left(1 - \frac{1}{2\sqrt{\pi}} \frac{\Gamma(n/2)}{\Gamma((n+1)/2)} \left(\frac{\rho(r_{bound})}{\delta_0}\right)^{n-1}\right) \tag{18}$$

and asymptotically it holds that

$$N_{max} = \mathcal{O}\left(-\ln \gamma \left(\frac{\delta_0}{r_{bound}}\right)^{2n}\right), \tag{19}$$

with $r_{bound} = \lambda^{-1}_{\mathcal{B}_0^S}(1 - \mu)L(\varphi_1)$ and $\rho(r_{bound}) = r_{bound} \cdot \sin(\pi/2 - \arcsin(r/2\delta_0))$.

The full proof is provided in the Appendix. *Proof sketch:* As the radius r_φ of the spherical cap is very small, we underestimate the area of the cap by removing the curvature and using the volume of an $n - 1$ dimensional ball with radius $\rho(r_{bound})$ as shown in Fig. 2.

Thus, after finishing our global search strategy for timestep t_j, we have the stochastic guarantee that the functional values of every $\varphi \in \mathbb{R}^{n-1}$ are greater or equal to $\mu \cdot \bar{m}$. This implies that we should initiate the search with a relatively large $\mu = \mu_1$, obtaining for every φ a relatively large value of r_{φ,μ_1} and therefore obtain a faster coverage of the search space. Subsequently, we can investigate whether the reachset $\mathcal{B}_j$ with radius $\delta_j = -\mu_1 \cdot \bar{m}$ intersects with a region of bad (unsafe) states. If this is not the case, we can proceed to the next timestep t_{j+1}. Otherwise, we reduce μ to $\mu_2 < \mu_1$, which reduces the safety regions $B(\varphi, r_\varphi)^S$ and thus the already-covered-set $\mathcal{S}$. This means that we continue with our search strategy until the desired probability $1 - \gamma$ is reached again for a smaller radius $\delta_j = -\mu_2 \cdot \bar{m}$. Accordingly, we can find a first radius for $\mathcal{B}_j$ faster and refine it as long as $\mathcal{B}_j$ intersects with the region of bad states.

Theorem 3 guarantees convergence of the algorithm. It shows that for a given confidence level γ, our algorithm terminates after at most N_{max} steps. Essentially, the theorem leads us to the significant result that the problem of constructing an ellipsoid abstraction of the true reachset with probabilistic guarantees for a Neural ODE is able to terminate.

Additionally, the theorem assumes that we know the local Lipschitz constant, which is a reasonable assumption for proving convergence. In practice, one can safely replace the true Lipschitz constant by an upper-bound.

Computational Complexity

The complexity of Algorithm 1 depends on the geometry of the loss surface. In particular, Algorithm 1 may terminate after one iteration in case of a flat surface, whereas an exponential number may be needed for ill-posed problems, as is common practice when deriving convergence rates for gradient descent (Nagy and Palmer 2003; Drori 2017)

The runtime of Algorithm 2 is determined by the complexity of the ODE solver for simulating the given differential equation. For example, given the number of integration steps (implicit interpretation of the number of layers in a deep model) L, and the time horizon of the simulation T, Algorithm 2 runs in time $\mathcal{O}(L \times T)$ and constant memory cost $\mathcal{O}(1)$ for each layer of a neural network f.

The complexity of Algorithm 3 depends on the local Lipschitz constant and the smoothness of the flow. Computing the true Lipschitz constant of a neural network is known to be NP-complete (Virmaux and Scaman 2018). However, Algorithm 3 operates correctly when we replace the true Lipschitz constant by an easier-to-compute upper bound, obtained for instance by means of interval arithmetic.

Algorithm 4 implements the main routine of our framework. Its complexity for a given confidence score γ equals the convergence rate N_{max} proven in Theorem 3, Eq. (19) for every Reachset. In particular, the runtime of Algorithm 4 depends exponentially on the dimension of the given Neural ODE and logarithmically on the confidence score.

Conclusions and Future Work

In this paper, we considered the verification problem for Neural ODEs. We introduced the SLR verification scheme, which is based on solving a global optimization problem. We designed a forward formulation of the adjoint method for the gradient descent algorithm. We also established strong convergence guarantees for SLR, showing that it can establish tight ellipsoidal bounds for the Neural ODE under consideration, at an arbitrary time horizon.

An important future direction will be to improve the current convergence rate, which is exponential in the dimensionality of the Neural ODE network. Existing statistical verification methods are mostly concerned with the verification of (hybrid) dynamical systems having various uncertainties in model parameters, discrete jumps between modes,

and/or initial states. We emphasize that reachability computation for Neural ODEs developed at scale will require dedicated methods tailored for that specific purpose.

Acknowledgements

The authors would like to thank the reviewers for their insightful comments. RH and RG were partially supported by Horizon-2020 ECSEL Project grant No. 783163 (iDev40). RH was partially supported by Boeing. ML was supported in part by the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award). SG was funded by FWF project W1255-N23. JC was partially supported by NAWA Polish Returns grant PPN/PPO/2018/1/00029. SS was supported by NSF awards DCL-2040599, CCF-1918225, and CPS-1446832.

Appendix

Theorem 1 (Radius of Safety Region)**.** *At target time t_j, let $\bar{m}$ be the current global minimum $\bar{m} = \min_{\varphi\in\mathcal{V}} L(\varphi)$. Let $\varphi \in \mathcal{V}$ be an already visited point with value $L(\varphi)$ ($\geq \bar{m}$) and let r_φ and $B(\varphi, r_\varphi)^S$ be defined as follows with $\mu \geq 1$:*

$$r_\varphi = \lambda_{\Sigma_\varphi}^{-1} \left(L(\varphi) - \mu \cdot \bar{m}\right) \tag{20}$$

with $\lambda_{\Sigma_\varphi} = \max_{x(\psi)\in\Sigma_\varphi} \|\partial_x \chi_{t_0}^{t_j}(x(\psi))\|_{M_{0,j}}$. If Σ_φ is chosen s.t. $\Sigma_\varphi \supseteq B(\varphi, r_\varphi)^S$, then it holds that

$$L(\psi) \geq \mu \cdot \bar{m} \quad \forall x(\psi) \in B(\varphi, r_\varphi)^S \tag{21}$$

Proof. Given r_φ and λ_{Σ_φ} as defined in the above theorem. Using the mean value inequality for vector valued functions, the triangle inequality, and considering the change of metric (Cyranka et al. 2017, Lemma 2) it holds that:

$$\begin{aligned}
&|L(\varphi_1) - L(\varphi_2)| = \\
&\left| \left\| \chi_{t_0}^{t_j}(x(\varphi_1)) - \chi_{t_0}^{t_j}(x_0) \right\|_{M_j} - \left\| \chi_{t_0}^{t_j}(x(\varphi_2)) - \chi_{t_0}^{t_j}(x_0) \right\|_{M_j} \right| \\
&\leq \left\| \chi_{t_0}^{t_j}(x(\varphi_1)) - \chi_{t_0}^{t_j}(x(\varphi_2)) \right\|_{M_j} \\
&\leq \lambda_{\Sigma_\varphi} \|x(\varphi_1) - x(\varphi_2)\|_I \\
&\forall x(\varphi_1), x(\varphi_2) \in \Sigma_\varphi \supseteq B(\varphi, r_\varphi)^S
\end{aligned}$$

Thus λ_{Σ_φ} is a local Lipschitz constant in the safety region $B(\varphi, r_\varphi)^S$. By definition $\|x(\psi) - x(\varphi)\| \leq r_\varphi$ for $x(\psi), \in B(\varphi, r_\varphi)^S$. Hence:

$$\begin{aligned}
&|L(\psi) - L(\varphi)| \\
&\leq \lambda_{\Sigma_\varphi} \|x(\psi) - x(\varphi)\| \\
&\leq \lambda_{\Sigma_\varphi} r_\varphi = L(\varphi) - \mu \cdot \bar{m} \quad \forall x(\psi) \in B(\varphi, r_\varphi)^S
\end{aligned} \tag{22}$$

To prove that Eq. (21) holds, we distinguish between two cases for ψ: (1) $L(\psi) \geq L(\varphi)$ and (2) $L(\psi) < L(\varphi)$. Case (1) is straightforward: $L(\psi) \geq L(\varphi) \geq \mu \cdot L(\varphi) \geq \mu \cdot \bar{m}$. In case (2), we use Eq. (22) and thus:

$$\begin{aligned}
L(\varphi) - L(\psi) = |L(\psi) - L(\varphi)| \leq L(\varphi) - \mu \cdot \bar{m} \\
\implies L(\psi) \geq \mu \cdot \bar{m},
\end{aligned}$$

proving that Eq. (21) holds no matter if $L(\psi) \geq L(\varphi)$ or if $L(\psi) < L(\varphi)$, for all $x(\psi) \in B(\psi, r_\psi)^S$. □

Theorem 2 (Convergence Guarantees)**.** *Given $\gamma \in (0, 1)$, $\mu \geq 1$, local Lipschitz constant $\lambda_{\mathcal{B}_0^S}$ and $N = |\mathcal{U}|$, where $\mathcal{U}$ is the set of uniform-randomly generated points during global search process. Let $\bar{m} = \min_{\varphi\in\mathcal{V}} L(\varphi)$ be the current minimum, $m^\star = \min_{\varphi\in\mathbb{R}^{n-1}} L(\varphi)$ the global minimum, and $\varphi^\star$ an argument s.t. $L(\varphi^\star) = m^\star$. It holds that*

$$\lim_{N\to\infty} \Pr(\mu \cdot \bar{m}_N \leq m^\star) = 1 \tag{23}$$

and thus

$$\forall \gamma \in (0, 1) \exists N \in \mathbb{N}, \text{ s.t. } \Pr(\mu \cdot \bar{m}_N \leq m^\star) \geq 1 - \gamma \tag{24}$$

Proof. With $\mathcal{C}(r_\varphi) = B(\varphi^\star, r_\varphi)^S$ being the spherical-cap as illustrated in Fig. 2, we derive some probabilities:

$$\begin{aligned}
\Pr(B(\varphi_j, r_{\varphi_j})^S \not\ni \varphi^\star) &= 1 - \Pr(\mathcal{C}(r_{\varphi_j})) \\
\Pr(\forall \varphi \in \mathcal{U} \colon B(\varphi, r_\varphi)^S \not\ni \varphi^\star) &= \prod_{\varphi\in\mathcal{U}} \left(1 - \Pr(\mathcal{C}(r_\varphi))\right) \\
\Pr(\exists \varphi \in \mathcal{U} \colon B(\varphi, r_\varphi)^S \ni \varphi^\star) &= 1 - \prod_{\varphi\in\mathcal{U}} \left(1 - \Pr(\mathcal{C}(r_\varphi))\right)
\end{aligned} \tag{25}$$

Using Theorem 1, if $\varphi^\star \in B(\varphi, r_\varphi)^S$ for some $\varphi \in \mathcal{U}$, then $\mu \cdot \bar{m} \leq L(\varphi^\star) = m^\star$ holds, thus:

$$\begin{aligned}
&\Pr(\mu \cdot \bar{m} \leq m^\star) \geq \\
&\Pr(\exists \varphi \in \mathcal{U} \colon B(\varphi, r_\varphi)^S \ni \varphi^\star)
\end{aligned} \tag{26}$$

Thus, it holds that $\Pr(\mu \cdot \bar{m} \leq m^\star) \geq 1 - \prod_{\varphi\in\mathcal{U}} \left(1 - \Pr(\mathcal{C}(r_\varphi))\right)$, with r_φ as defined in Eq. (20).

$$r_\varphi \geq \lambda_{\mathcal{B}_0^S}^{-1} (\min_{\varphi\in\mathcal{U}} L(\varphi) - \mu \cdot \bar{m}) \tag{27}$$

$$\geq \lambda_{\mathcal{B}_0^S}^{-1} (1 - \mu) \bar{m} \tag{28}$$

$$\geq \lambda_{\mathcal{B}_0^S}^{-1} (1 - \mu) L(\varphi_1) = r_{bound} \quad \forall \varphi \in \mathcal{U}, \tag{29}$$

with φ_1 being the first random sampled point. Hence:

$$\Pr(\mu \cdot \bar{m} \leq m^\star) \geq 1 - \prod_{\varphi\in\mathcal{U}} \left(1 - \Pr(\mathcal{C}(r_{bound}))\right) \tag{30}$$

$$\geq 1 - \left(1 - \Pr(\mathcal{C}(r_{bound}))\right)^N \tag{31}$$

As $\Pr(\mathcal{C}(r_{bound})) \in (0, 1)$, it follows that Eq. (23) holds and thus we are able to guarantee the convergence of our global search strategy. □

Theorem 3 (Convergence Rate)**.** *Given $\gamma \in (0, 1)$, $\mu \geq 1$, local Lipschitz constant $\lambda_{\mathcal{B}_0^S}$ and dimension n, let φ_1 be the first random sample point. We can guarantee that $\Pr(\mu \cdot \bar{m} \leq m^\star) \geq 1 - \gamma$ if we perform at most N_{max} iterations of the SLR Algorithm 4, with*

$$N_{max} = \ln \gamma \Big/ \ln \left(1 - \frac{1}{2\sqrt{\pi}} \frac{\Gamma(n/2)}{\Gamma((n+1)/2)} \left(\frac{\rho(r_{bound})}{\delta_0}\right)^{n-1}\right) \tag{32}$$

and asymptotically it holds that

$$N_{max} = \mathcal{O}\left(-\ln\gamma\left(\frac{\delta_0}{r_{bound}}\right)^{2n}\right), \tag{33}$$

with $r_{bound} = \lambda^{-1}_{\mathcal{B}_0^S}(1-\mu)L(\varphi_1)$ *(as defined in Eq.* (29)*) and* $\rho(r_{bound}) = r_{bound} \cdot \sin(\pi/2 - \arcsin(r/2\delta_0))$.

Proof. If the right-hand side of Eq. (31) equals $1-\gamma$, it would hold that $\Pr(\mu\cdot\bar{m} \leq m^\star) \geq 1-\gamma$. Thus reformulating that equation, we get an upper bound N:

$$1-(1-\Pr(\mathcal{C}(r_{bound})))^N = 1-\gamma \quad \Leftrightarrow \tag{34}$$

$$(1-\Pr(\mathcal{C}(r_{bound})))^N = \gamma \quad \Leftrightarrow \tag{35}$$

$$\frac{\ln(\gamma)}{\ln(1-\Pr(\mathcal{C}(r_{bound})))} = N \tag{36}$$

To get a lower bound of $\Pr(\mathcal{C}(r_{bound}))$ we underestimate the area of the cap by removing the curvature and using the volume of an $n-1$ dimensional ball with radius $\rho(r_{bound})$: Using the results of (Shengqiao 2011) it holds that $\text{Area}(\mathcal{C}(r_{bound})) \geq \text{Vol}_{n-1}(\rho(r_{bound}))$, thus:

$$\Pr(\mathcal{C}(r_{bound})) \geq \frac{\text{Vol}_{n-1}(\rho(r_{bound}))}{\text{Area}(\mathcal{B}_0)} = \frac{\pi^{(n-1)/2}}{\Gamma((n+1)/2)}\rho(r_{bound})^{n-1}\frac{\Gamma(n/2)}{2\pi^{n/2}}\frac{1}{\delta_0^{n-1}} \tag{37}$$

with Γ being the gamma function. Combining Eq. (37) with Eq. (36) we prove equation (32). Using the information that $\ln(1-x) \approx -x$ and using the results of (Zhigljavsky and Zilinskas 2008)[Section 2.2], the asymptotical value of N_{max} equals to Eq. (33) □

References

Abeyaratne, R. 1998. *Continuum Mechanics*. Lecture Notes on The Mechanics of Elastic Solids.

Boender, C.; Rinnooy Kan, A.; Timmer, G.; and Stougie, L. 1982. A stochastic method for global optimization. *Mathematical Programming* 22: 125–140.

Bortolussi, L.; and Sanguinetti, G. 2014. A Statistical Approach for Computing Reachability of Non-linear and Stochastic Dynamical Systems. In Norman, G.; and Sanders, W., eds., *Quantitative Evaluation of Systems*, 41–56. Cham: Springer International Publishing.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 6571–6583. Curran Associates, Inc.

Chen, X.; Ábrahám, E.; and Sankaranarayanan, S. 2013. Flow*: an Analyzer for Non-linear Hybrid Systems. In *CAV*, 258–263.

Cyranka, J.; Islam, M. A.; Byrne, G.; Jones, P.; Smolka, S. A.; and Grosu, R. 2017. Lagrangian Reachabililty. In Majumdar, R.; and Kunčak, V., eds., *CAV'17, the 29th International Conference on Computer-Aided Verification*, 379–400. Heidelberg, Germany: Springer.

Cyranka, J.; Islam, M. A.; Smolka, S. A.; Gao, S.; and Grosu, R. 2018. Tight Continuous-Time Reachtubes for Lagrangian Reachability. In *CDC'18, the 57th IEEE Conference on Decision and Control*, 6854–6861. Miami Beach, FL, USA: IEEE.

Devonport, A.; Khaled, M.; Arcak, M.; and Zamani, M. 2020. PIRK: Scalable Interval Reachability Analysis for High-Dimensional Nonlinear Systems. In Lahiri, S. K.; and Wang, C., eds., *Computer Aided Verification*, 556–568. Cham: Springer International Publishing.

Donzé, A. 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV'10, the 22nd International Conference on Computer Aided Verification*, 167–170. Edinburgh, UK: Springer.

Donzé, A.; and Maler, O. 2007. Systematic simulation using sensitivity analysis. In *International Workshop on Hybrid Systems: Computation and Control*, 174–189. Springer.

Drori, Y. 2017. The exact information-based complexity of smooth convex minimization. *Journal of Complexity* 39: 1–16.

Duggirala, P. S.; Mitra, S.; Viswanathan, M.; and Potok, M. 2015. C2E2: A Verification Tool for Stateflow Models. In Baier, C.; and Tinelli, C., eds., *Tools and Algorithms for the Construction and Analysis of Systems*, 68–82. Berlin, Heidelberg: Springer Berlin Heidelberg.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented neural odes. In *Advances in Neural Information Processing Systems*, 3140–3150.

Durkan, C.; Bekasov, A.; Murray, I.; and Papamakarios, G. 2019. Neural spline flows. In *Advances in Neural Information Processing Systems*, 7511–7522.

Enszer, J. A.; and Stadtherr, M. A. 2011. Verified Solution and Propagation of Uncertainty in Physiological Models. *Reliab. Comput.* 15(3): 168–178. URL http://interval.louisiana.edu/reliable-computing-journal/volume-15/no-3/reliable-computing-15-pp-168-178.pdf.

Erichson, N. B.; Azencot, O.; Queiruga, A.; and Mahoney, M. W. 2020. Lipschitz recurrent neural networks. *arXiv preprint arXiv:2006.12070* .

Fan, C.; Kapinski, J.; Jin, X.; and Mitra, S. 2017. Simulation-Driven Reachability Using Matrix Measures. *ACM Trans. Embed. Comput. Syst.* 17(1).

Finlay, C.; Jacobsen, J.-H.; Nurbekyan, L.; and Oberman, A. 2020. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *International Conference on Machine Learning*, 3154–3164. PMLR.

Fränzle, M.; Hahn, E.; Hermanns, H.; Wolovick, N.; and Zhang, L. 2011. Measurability and safety verification for stochastic hybrid systems. In *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*, 43–52.

Fränzle, M.; Teige, T.; and Eggers, A. 2010. Engineering constraint solvers for automatic analysis of probabilistic hybrid automata. *The Journal of Logic and Algebraic Programming* 79(7): 436 – 466. The 20th Nordic Workshop on Programming Theory (NWPT 2008).

Gao, S.; Kong, S.; and Clarke, E. M. 2013. Satisfiability modulo ODEs. In *2013 Formal Methods in Computer-Aided Design*, 105–112.

Gruenbacher, S.; Cyranka, J.; Islam, M. A.; Tschaikowski, M.; Smolka, S.; and Grosu, R. 2019. Under the Hood of a Stand-Alone Lagrangian Reachability Tool. *EPiC Series in Computing* 61.

Gruenbacher, S.; Cyranka, J.; Lechner, M.; Islam, M. A.; Smolka, S. A.; and Grosu, R. 2020. Lagrangian Reachtubes: The Next Generation. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 1556–1563. ISSN 2576-2370.

Hansen, E. 1980. Global Optimization Using Interval Analysis – the Multi-Dimensional Case. *Numer. Math.* 34(3): 247–270.

Hansen, N.; and Ostermeier, A. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9(2): 159–195.

Hasani, R.; Lechner, M.; Amini, A.; Rus, D.; and Grosu, R. 2020a. Liquid Time-constant Networks. *arXiv preprint arXiv:2006.04439* .

Hasani, R.; Lechner, M.; Amini, A.; Rus, D.; and Grosu, R. 2020b. A Natural Lottery Ticket Winner: Reinforcement Learning with Ordinary Neural Circuits. In *International Conference on Machine Learning*, 4082–4093. PMLR.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Holl, P.; Koltun, V.; and Thuerey, N. 2020. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457* .

Huang, C.; Chen, X.; Lin, W.; Yang, Z.; and Li, X. 2017. Probabilistic Safety Verification of Stochastic Hybrid Systems Using Barrier Certificates. *ACM Trans. Embed. Comput. Syst.* 16(5s). ISSN 1539-9087.

Igel, C.; Hansen, N.; and Roth, S. 2007. Covariance Matrix Adaptation for Multi-objective Optimization. *Evolutionary Computation* 15(1): 1–28.

Immler, F. 2015. Verified Reachability Analysis of Continuous Systems. In Baier, C.; and Tinelli, C., eds., *Tools and Algorithms for the Construction and Analysis of Systems*, 37–51. Berlin, Heidelberg: Springer Berlin Heidelberg.

Jia, J.; and Benson, A. R. 2019. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems*, 9847–9858.

Kapela, T.; Mrozek, M.; Wilczak, D.; and Zgliczynski, P. 2020. CAPD:: DynSys: a flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Pre-Print - ww2.ii.uj.edu.pl* .

Kidger, P.; Morrill, J.; Foster, J.; and Lyons, T. 2020. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926* .

Lechner, M.; and Hasani, R. 2020. Learning Long-Term Dependencies in Irregularly-Sampled Time Series. *arXiv preprint arXiv:2006.04418* .

Lechner, M.; Hasani, R.; Amini, A.; Henzinger, T. A.; Rus, D.; and Grosu, R. 2020. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence* 2(10): 642–652.

Li, D.; Bak, S.; and Bogomolov, S. 2020. Reachability Analysis of Nonlinear Systems Using Hybridization and Dynamics Scaling. In Bertrand, N.; and Jansen, N., eds., *Formal Modeling and Analysis of Timed Systems*, 265–282. Cham: Springer International Publishing.

Lohner, R. 1992. Computation of Guaranteed Enclosures for the Solutions of Ordinary Initial and Boundary Value Problems. In J.R. Cash, I. G., ed., *Computational Ordinary Differential Equations*. Clarendon Press, Oxford.

Malherbe, C.; and Vayatis, N. 2017. Global Optimization of Lipschitz Functions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 2314–2323. JMLR.org.

Meyer, P.-J.; Devonport, A.; and Arcak, M. 2019. TIRA: Toolbox for Interval Reachability Analysis. In *Association for Computing Machinery*, HSCC '19, 224–229. New York, NY, USA.

Nagy, J. G.; and Palmer, K. M. 2003. Steepest descent, CG, and iterative regularization of ill-posed problems. *BIT Numerical Mathematics* 43(5): 1003–1017.

Neumaier, A. 2004. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica* 13: 271–369.

Piyavskii, S. 1972. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics* 12(4): 57 – 67.

Pontryagin, L. S. 2018. *Mathematical theory of optimal processes*. Routledge.

Quaglino, A.; Gallieri, M.; Masci, J.; and Koutník, J. 2019. Snode: Spectral discretization of neural odes for system identification. *arXiv preprint arXiv:1906.07038* .

Rinnooy Kan, A.; and Timmer, G. 1987a. Stochastic global optimization methods part I: Clustering methods. *Mathematical Programming* 39: 27—56.

Rinnooy Kan, A.; and Timmer, G. 1987b. Stochastic global optimization methods part II: Multi level methods. *Mathematical Programming* 39: 57—-78.

Rubanova, Y.; Chen, R. T.; and Duvenaud, D. K. 2019. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, 5320–5330.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *nature* 323(6088): 533–536.

Shengqiao, L. 2011. Concise Formulas for the Area and Volume of a Hyperspherical Cap. *Asian Journal of Mathematics and Statistics* 4.

Shmarov, F.; and Zuliani, P. 2015a. ProbReach: A Tool for Guaranteed Reachability Analysis of Stochastic Hybrid Systems. In Bogomolov, S.; and Tiwari, A., eds., *1st International Workshop on Symbolic and Numerical Methods for Reachability Analysis, SNR@CAV 2015, San Francisco, CA, USA, July 19, 2015*, volume 37 of *EPiC Series in Computing*, 40–48. EasyChair. URL https://easychair.org/publications/paper/z1f.

Shmarov, F.; and Zuliani, P. 2015b. ProbReach: verified probabilistic delta-reachability for stochastic hybrid systems. In Girard, A.; and Sankaranarayanan, S., eds., *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015*, 134–139. ACM.

Shubert, B. O. 1972. A Sequential Method Seeking the Global Maximum of a Function. *SIAM Journal on Numerical Analysis* 9(3): 379–388. ISSN 00361429. URL http://www.jstor.org/stable/2156138.

Slaughter, W. 2002. *The Linearized Theory of Elasticity*. Springer Science and Business Media, LLC.

Stepanenko, S. 2009. *Global Optimization Methods based on Tabu Search : GTS, GOTS, TSPA. Application for conformation searches.* Bod.

Virmaux, A.; and Scaman, K. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, 3835–3844.

Wang, Q.; Zuliani, P.; Kong, S.; Gao, S.; and Clarke, E. M. 2015. SReach: A Probabilistic Bounded Delta-Reachability Analyzer for Stochastic Hybrid Systems. In Roux, O.; and Bourdon, J., eds., *Computational Methods in Systems Biology*, 15–27. Cham: Springer International Publishing.

Yan, H.; Du, J.; Tan, V. Y.; and Feng, J. 2020. On robustness of neural ordinary differential equations. *International Conference on Learning Representations* .

Yang, Y.; Wu, J.; Li, H.; Li, X.; Shen, T.; and Lin, Z. 2020. Dynamical System Inspired Adaptive Time Stepping Controller for Residual Network Families. *Thirty-Fourht AAAI Conference on Artificial Intelligence* .

Zgliczynski, P. 2002. C1 Lohner Algorithm. *Foundations of Computational Mathematics* 429–465.

Zhigljavsky, A.; and Zilinskas, A. 2008. *Stochastic Global Optimization*, volume 9 of *Springer Optimization and Its Applications*. Springer US.

Zhuang, J.; Dvornek, N.; Li, X.; Tatikonda, S.; Papademetris, X.; and Duncan, J. 2020. Adaptive Checkpoint Adjoint Method for Gradient Estimation in Neural ODE. *arXiv preprint arXiv:2006.02493* .