A Novel Approach to the Initial Value Problem with a Complete Validated Algorithm ★

Bingwei Zhang[a,1] (Researcher), Chee Yap[a,*] (Researcher)

[a]*The Courant Institute of Mathematical Sciences, , New York, USA*

ARTICLE INFO

Keywords:
initial value problem, IVP, reachability problem, end-enclosure problem, radical transform, validated algorithms, interval methods, logarithmic norm, matrix measure, contraction maps

ABSTRACT

We consider the first order autonomous differential equation (ODE) $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x})$ where $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz. For $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $h > 0$, the initial value problem (IVP) for $(\boldsymbol{f}, \boldsymbol{x}_0, h)$ is to determine if there is a unique solution, i.e., a function $\boldsymbol{x} : [0, h] \to \mathbb{R}^n$ that satisfies the ODE with $\boldsymbol{x}(0) = \boldsymbol{x}_0$. Write $\boldsymbol{x} = \mathtt{IVP}_{\boldsymbol{f}}(\boldsymbol{x}_0, h)$ for this unique solution.

We pose a corresponding computational problem, called the **End Enclosure Problem**: given $(\boldsymbol{f}, B_0, h, \varepsilon_0)$ where $B_0 \subseteq \mathbb{R}^n$ is a box and $\varepsilon_0 > 0$, to compute a pair of non-empty boxes $(\underline{B}_0, B_1)$ such that $\underline{B}_0 \subseteq B_0$, width of B_1 is $< \varepsilon_0$, and for all $\boldsymbol{x}_0 \in \underline{B}_0$, $\boldsymbol{x} = \mathtt{IVP}_{\boldsymbol{f}}(\boldsymbol{x}_0, h)$ exists and $\boldsymbol{x}(h) \in B_1$. We provide a algorithm for this problem. Under the assumption (promise) that for all $\boldsymbol{x}_0 \in B_0$, $\mathtt{IVP}_{\boldsymbol{f}}(\boldsymbol{x}_0, h)$ exists, we prove the halting of our algorithm. This is the first halting algorithm for IVP problems in such a general setting.

We also introduce novel techniques for subroutines such as StepA and StepB, and a scaffold datastructure to support our End Enclosure algorithm. Among the techniques are new ways refine full- and end-enclosures based on a **radical transform** combined with logarithm norms. Our preliminary implementation and experiments show considerable promise, and compare well with current algorithms.

1. Introduction

We consider the following system of first order ordinary differential equations (ODEs)

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x} = [x_1, \ldots, x_n] \in C^1([1, h] \to \mathbb{R}^n)$ are functions of time and $\boldsymbol{x}' = [x_1', \ldots, x_n']$ indicate differentiation with respect to time, and $\boldsymbol{f} = [f_1, \ldots, f_n] : \mathbb{R}^n \to \mathbb{R}^n$. Since this is an autonomous ODE, we may assume the initial time $t = 0$. Up to time scaling, we often assume that the end time is $h = 1$. This assumption is just for simplicity but our results and implementation allow any value of $h > 0$.

Given $\boldsymbol{p}_0 \in \mathbb{R}^n$ and $h > 0$, the **initial value problem** (IVP) for $(\boldsymbol{p}_0, h)$ is the mathematical problem of finding a **solution**, i.e., a continuous function $\boldsymbol{x} : [0, h] \to \mathbb{R}^n$ that satisfies (1), subject to $\boldsymbol{x}(0) = \boldsymbol{p}_0$. Let $\mathtt{IVP}_{\boldsymbol{f}}(\boldsymbol{p}_0, h)$ denote the set of all such solutions. Since $\boldsymbol{f}$ is usually fixed or understood, we normally omit $\boldsymbol{f}$ in our notations. We say that $(\boldsymbol{p}_0, h)$ is **valid** if the solution exists and is unique, i.e., $\mathtt{IVP}(\boldsymbol{p}_0, h) = \{\boldsymbol{x}_0\}$ is a singleton. In this case, we write $\boldsymbol{x}_0 = \mathtt{IVP}(\boldsymbol{p}_0, h)$. It is convenient to write $\boldsymbol{x}(t; \boldsymbol{p}_0)$ for $\boldsymbol{x}_0(t)$. See Figure 1 for the solution to the Volterra system (Eg1 in Table 1). The IVP problem has numerous applications such as modeling physical, chemical and biological systems, and dynamical system.

The mathematical IVP gives rise to a variety of algorithmic problems since we generally cannot represent a solution $\boldsymbol{x}_0 = \mathtt{IVP}(\boldsymbol{p}_0, h)$. We are interested in **validated algorithms** [1] meaning that all approximations must be explicitly bounded (e.g., numbers are enclosed in intervals). In this setting, we introduce the simplest *algorithmic* IVP problem, that of computing an enclosure for $\boldsymbol{x}(h; \boldsymbol{p}_0)$. In real world applications, only approximate values of $\boldsymbol{p}_0$ are truly meaningful because of modeling uncertainties. So we replace $\boldsymbol{p}_0$ by a **region** $B_0 \subseteq \mathbb{R}^n$: B_0 is a non-empty set like a box

★ This document contain the results of the research funded by NSF Grant #CCF-2212462.

This work demonstrates a_b the formation Y_1 of a complete validated algorithm for solving initial value problems through novel computational approaches.

* Corresponding author

bz2517@nyu.edu (B. Zhang); yap@cs.nyu.edu (C. Yap)

ORCID(s): 0009-0002-2619-9807 (B. Zhang); 0000-0003-2952-3545 (C. Yap)

[1] Both authors contributed equally to this research

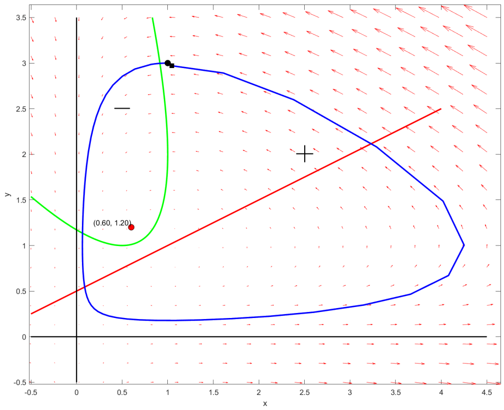

Figure 1: Volterra system (Eg1). The negative zone of the system is the region above the green parabola.

or ball. Let $\mathtt{IVP}(B_0, h) := \bigcup_{p \in B_0} \mathtt{IVP}(p_0, h)$. Call $B_1 \subseteq \mathbb{R}^n$ an **end-enclosure** for $\mathtt{IVP}(B_0, h)$ if we have the inclusion $\{x(h) : x \in \mathtt{IVP}(B_0, h)\} \subseteq B_1$. So our formal algorithmic problem is the following **End Enclosure Problem**:

$\mathtt{EndEncl\_IVP}_f(B_0, \varepsilon) \to (\underline{B}_0, \overline{B}_1)$
INPUT: $\varepsilon > 0$, $B_0 \subseteq \mathbb{R}^n$ is a non-empty box,
such that $\mathtt{IVP}_f(B_0, 1)$ is valid.
OUTPUT: non-empty boxes $\underline{B}_0, \overline{B}_1$ in $\mathbb{R}^n$
with $\underline{B}_0 \subseteq B_0$, $w_{\max}(\overline{B}_1) < \epsilon$
and $\overline{B}_1$ is an end-enclosure of $\mathtt{IVP}_f(\underline{B}_0, 1)$.

(2)

This is called the **Reachability Problem** in the non-linear control systems and verification literature (e.g., [2]). Note that we allow B_0 to be shrunk to some $\underline{B}_0$ in order to satisfy the user-specified bound of ε. If it is promised that $\underline{B}_0 = B_0$ has solution, we can also turn off shrinking. This is a novel feature that will prove very useful in practice. The usual formulation of the IVP problem assumes that B_0 is a singleton $\{p_0\}$. In this case, our algorithm will output $\underline{B}_0 = B_0$.

1.1. What is Achieved

Our formulation of the end-enclosure problem (2) is new. We will present an algorithm for this problem. Our algorithm is **complete** in that sense that if the input is valid, then [C0] the algorithm halts, and [C1] if the algorithm halts, the output $(\underline{B}_0, \overline{B}_1)$ is correct. Algorithms that only satisfy [C1] are said[2] to be **partially correct**. To our knowledge, current validated IVP algorithms are only partially correct since halting is not proved.

The input to $\mathtt{EndEncl\_IVP}_f(B_0, \varepsilon)$ assumes the validity of $(B_0, 1)$. All algorithms have requirements on their inputs, but they are typically syntax requirements which are easily checked. But validity of $(B_0, 1)$ is a semantic requirement which is non-trivial to check. Problems with semantic conditions on the input are called **promise problems** [3]). Many numerical algorithms are actually solutions of promise problems. Checking if the promise holds is a separate decision problem. To our knowledge, deciding validity of $(B_0, 1)$ is an open problem although some version of this question is undecidable in the analytic complexity framework [4, 5].

Hans Stetter [6] summarized the state-of-the-art over 30 years ago as follows: *To date, no programs that could be truly called 'scientific software' have been produced. AWA is state-of-art, and can be used by a sufficiently expert user – it requires selection of step-size, order and suitable choice of inclusion set represention.* Corliss [7, Section 10] made similar remarks. We believe our algorithm meets Stetter's and Corliss' criteria. The extraneous inputs such as step-size,

[2]Completeness and partial correctness are standard terms in theoretical computer science.

order, etc, noted by Stetter are usually called **hyperparameters**. Our algorithm[3] does not require any hyperparamters. Our preliminary implementation shows the viability of our algorithm, and its ability to do certain computations where current IVP software fails.

1.2. In the Shadow of Lohner's AWA

In their comprehensive 1999 review, Nedialkov et al. [8] surveyed a family of validated IVP algorithms that may be[4] called **A/B-algorithms** because each computation amounts to a sequence of steps of the form $\underbrace{ABAB\cdots AB}_{2m} = (AB)^m$ for some $m \geq 1$, where A and B refer to two subroutines which[5] we call `StepA` and `StepB`. It appears that all validated algorithms follow this motif, including Berz and Makino [10] who emphasized their `StepB` based on Taylor models. Ever since Moore [11] pointed out the **wrapping effect**, experts have regarded the mitigation of this effect as essential. The solution based on iterated QR transformation by Lohner [12] is regarded as the best technique to do this. It was implemented in the software called AWA[6] and recently updated by Bunger [13] in a INTLAB/MATLAB implementation. The complexity and numerical issues of such iterated transformations have not studied but appears formidable. See Revol [14] for an analysis of the special case of iterating a fixed linear transformation. In principle, Lohner's transformation could be incorporated into our algorithm. By not doing this, we illustrate the extend to which other techniques could be used to produce viable validated algorithms.

In this paper, we introduce [N1] new methods to achieve variants of `StepA` and `StepB`, and [N2] data structures and subroutines to support more complex motifs than $(AB)^m$ above. Our algorithm is a synthesis of [N1]+[N2]. The methods under [N1] will refine full- and end-enclosures by exploiting logNorm and radical tranforms (see next). Under [N2], we design subroutines and the scaffold data-structure to support new algorithmic motifs such as $(AB^+)^m$, i.e., A followed by one or more B's. Moreover, B^+ is periodically replaced by calling a special "`EulerTube`" to achieve end-enclosures satisfying an priori δ-bound. This will be a key to our termination proof.

1.3. How we exploit Logarithmic Norm and Radical Transform

A **logNorm bound** of $B_1 \subseteq \mathbb{R}^n$ is any upper bound on

$$\mu_2(J_f(B_1)) := \sup\left\{\mu_2(J_f(p)) : p \in B_1\right\} \tag{3}$$

Unlike standard operator norms, logNorms can be negative. We call B_1 a **contraction zone** if it has a negative logNorm bound. Here, J_f is the Jacobian of f and μ_2 is the logNorm function (Subsection 2.5). We exploit the fact that

$$\|x(t; p_0) - x(t; p_1)\| \leq \|p_0 - p_1\| e^{t\overline{\mu}}$$

(Theorem 3 in Subsection 2.5). In the Volterra example in Figure 1, it can be shown that the exact contraction zone is the region above the green parabola. In tracing a solution $x(t; p_0)$ for $t \in [0, h]$ through a contraction zone, we can compute a end-enclosure B for $\mathbf{IVP}(B_0, h, B_1)$ with $w_{\max}(B) < w_{\max}(B_0)$ (i.e., the end-enclosure is "shrinking"). Previous authors have exploited logNorms in the IVP problem (e.g., Zgliczynski [15], Neumaier [16]). We will exploit it in new way via a transform: for any box $B_1 \subseteq \mathbb{R}^n$, we introduce a "radical map" $\pi : \mathbb{R}^n \to \mathbb{R}^n$ (Section 5) with $y = \pi(x)$. Essentially, this transform is

$$y = (x_1^{-d_1}, \ldots, x_n^{-d_n}) \qquad \text{(for some } d_1, \ldots, d_n \neq 0) \tag{4}$$

where $x = (x_1, \ldots, x_n)$. The system $x' = f(x)$ transforms to another system $y' = g(y)$ in which the logNorm of $\pi(B_1)$ has certain properties (e.g., $\pi(B_1)$ is a contraction zone in the (y, g)-space). By computing end-endclosures in the (y, g)-space, we infer a corresponding end-enclosure in the (x, f)-space. Our analysis of the 1-dimensional case (Subsection 5.2), suggests that the best bounds are obtained when the logNorm of $\pi(B_1)$ is close to 0. In our current code, computing $\pi(B_1)$ is expensive, and so we avoid doing a transform if $\mu_2(J_f(B_1))$ is already negative.

[3]Hyperparameters are useful when used correctly. Thus, our implementation has some hyperparameters that may be used to improve performance, but they are optional and have no effect on completeness.

[4]This $(AB)^+$ motif is shared with homotopy path algorithms where A and B are usually called predictor and corrector (e.g., [9]).

[5]Nedialkov et al. called them Algorithms I and II.

[6]"Anfangswertaufgabe", the German term for IVP.

1.4. Brief Literature Review

The validated IVP literature appeared almost from the start of interval analysis, pioneered by Moore, Eijgenraam, Rihm and others [17, 18, 19, 1, 20]. Corliss [7] surveys this early period. Approaches based on Taylor expansion is dominant as they benefit from techniques such automatic differentiation and data structures such as the **Taylor model**. The latter, developed and popularized by Makino and Berz [13, 21, 10], has proven to be very effective. A major activity is the development of techniques to control the "wrapping effect". Here Lohner's approach [22, 12] has been most influential. Another advancement is the C^r-Lohner method developed by Zgliczyński et al. [15, 23]. This approach involves solving auxiliary IVP systems to estimate higher order terms in the Taylor expansion. The field of validated methods, including IVP, underwent great development in the decades of 1980-2000. Nedialkov et al provide an excellent survey of the various subroutines of validated IVP [24, 25, 26, 8].

In Nonlinear Control Theory (e.g., [27, 28]), the End-Enclosure Problem is studied under various **Reachability** problems. In complexity theory, Ker-i Ko [5] has shown that IVP is PSPACE-complete. This result makes the very strong assumption that the search space is the unit square ($n = 1$). Bournez et al [29] avoided this restriction by assuming that $\boldsymbol{f}$ has analytic extension to $\mathbb{C}^d$.

The concept of **logarithmic norm**[7] (or **logNorm** for short) was independently introduced by Germund Dahlquist and Sergei M. Lozinskiĭ in 1958 [30]. The key motivation was to improve bound errors in IVP. Neumaier [31] is one of the first to use logNorms in validated IVP. The earliest survey is T. Ström (1975) [32]. The survey of Gustaf Söderlind [30] extends the classical theory of logNorms to the general setting of functional analytic via Banach spaces.

One of the barriers to the validated IVP literature is cumbersome notations and lack of precise input/output criteria for algorithms. For instance, in the A/B algorithms, it is not stated if a target time $h > 0$ is given (if given, how it is used other than to terminate). Algorithm 5.3.1 in [8] is a form of `StepA` has a $\varepsilon > 0$ argument but how it constraints the output is unclear, nor is it clear how to use this argument in the A/B algorithm. We provide a streamlined notation, largely by focusing on autonomous ODEs, and by introducing high-level data structures such as the scaffold. Besides non-halting and lack of input/output specification, another issue is the use of an indeterminate "failure mode" (e.g., [26, p.458, Figure 1])

1.5. Paper Overview

The remainder of the paper is organized as follows: **Section 2** introduces some key concepts and computational tools. **Section 3** gives an overview of our algorithm. **Section 4** describes our `StepA` and `StepB` subroutines. **Section 5** describes our transform approach to obtain tighter enclosures. **Section 6** describes the `Extend` and `Refine` subroutines. **Section 7** presents our main algorithm and some global experiments. We conclude in **Section 8**. **Appendix A** gives all the proofs. **Appendix B** provide details of the affine transform $\overline{\pi}$.

2. Basic Tools

2.1. Notations and Key Concepts

We use bold fonts such as $\boldsymbol{x}$ for vectors. A point $\boldsymbol{p} \in \mathbb{R}^n$ is viewed as a column vector $\boldsymbol{p} = [p_1, \ldots, p_n]$, with transpose the row vector $\boldsymbol{p}^\top = (p_1, \ldots, p_n)$. Also vector-matrix or matrix-matrix products are indicated by $\bullet$ (e.g., $A \bullet \boldsymbol{p}$). Let $\square\mathbb{R}^n$ denote the set of n-dimensional **boxes** in $\mathbb{R}^n$ where a box B is viewed as a subset of $\mathbb{R}^n$. The **width** and **midpoint** of an interval $I = [a, b]$ are $w(I) := b - a$ and $m(I) := (a + b)/2$, respectively. If $B = \prod_{i=1}^n I_i$, its **width** and **midpoint** are $\boldsymbol{w}(B) := (w(I_1), \ldots, w(I_n))$ and $\boldsymbol{m}(B) := (m(I_1), \ldots, m(I_n))$. Also, **maximum width** and **minimum width** are $w_{\max}(B) := \max_{i=1}^n w(I_i)$ and $w_{\min}(B) := \min_{i=1}^n w(I_i)$. We assume $w_{\min}(B) > 0$ for boxes.

We use the Euclidean norm on $\mathbb{R}^n$, writing $\|\boldsymbol{p}\| = \|\boldsymbol{p}\|_2$. For any function $f : X \to Y$, we re-use 'f' to denote its **natural set extension**, $f : 2^X \to 2^Y$ where 2^X is the power set of X and $f(S) = \{f(x) : x \in S\}$ for all $S \subseteq X$.

The **image** of a function $f : A \to B$ is $\mathtt{image}(f) := \{f(a) : a \in A\}$. The **image** of $\mathtt{IVP}(B_0, h)$ is the union $\bigcup_{\boldsymbol{x} \in \mathtt{IVP}(B_0,h)} \mathtt{image}(\boldsymbol{x})$. A **full-enclosure** of $\mathtt{IVP}(B_0, h)$ is a set $B_1 \subseteq \mathbb{R}^n$ that contains $\mathtt{image}(\mathtt{IVP}(B_0, h))$. If, in addition, (B_0, h) is valid, then call (B_0, h, B_1) an **admissible triple**, equivalently, (h, B_1) is an **admissible pair** for B_0. We then write $\mathtt{IVP}(B_0, h, B_1)$ instead of $\mathtt{IVP}(B_0, h)$. Finally, $\underline{B}_1 \subseteq B_1$ is an **end-enclosure** for $\mathtt{IVP}(B_0, h, B_1)$ if for all solution $\boldsymbol{x} \in \mathtt{IVP}(B_0, h, B_1)$, we have $\boldsymbol{x}(h) \in \underline{B}_1$. Call $(B_0, h, B_1, \underline{B}_1)$ an **admissible quadruple** (or quad).

If $\mathtt{IVP}(B_0, h)$ is valid, then under the assumption $\boldsymbol{f} \not\equiv \boldsymbol{0}$, we have the following: for any $\boldsymbol{x}_0 \in B_0$, if $\boldsymbol{x}(t)$ is a solution with $\boldsymbol{x}(0) = \boldsymbol{x}_0$, then for all $t \in [0, h)$, it holds that $\boldsymbol{f}(\boldsymbol{x}(t)) \neq \boldsymbol{0}$.

[7]This concept goes by other names, including logarithmic derivative, matrix measure and Lozinskiĭ measure.

2.2. Implicit use of Interval Computation

For any $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}$, a **box form** of $\boldsymbol{g}$ is any function $\boldsymbol{G} : \square\mathbb{R}^n \to \square\mathbb{R}$ which is (1) conservative, and (2) convergent. This means (1) $\boldsymbol{g}(B) \subseteq \boldsymbol{G}(B)$ for all $B \in \square\mathbb{R}^n$, and (2) $\boldsymbol{g}(\boldsymbol{p}) = \lim_{i\to\infty} \boldsymbol{G}(B_i)$ for any infinite sequence $B_1, B_2, B_3, \ldots$ that converges to a point $\boldsymbol{p} \in \mathbb{R}^n$. In some proofs, it may appear that we need the additional condition, (3) that $\boldsymbol{G}$ is **isotone**. This means $B \subseteq B'$ implies $\boldsymbol{G}(B) \subseteq \boldsymbol{G}(B')$. In practice, isotony can often be avoided. E.g., in our termination proof below, we will indicate how to avoid isotony.

We normally denote a box form $\boldsymbol{G}$ of $\boldsymbol{g}$ by $\square\boldsymbol{g}$ (if necessary, adding subscripts or superscripts to distinguish various box forms of $\boldsymbol{g}$). See [33, 34]. In this paper, we will often "compute" exact bounds such as "$\boldsymbol{f}(E_0)$" (e.g., in `StepA`, Subsection 4.1). But in implementation, we really compute a box form $\square\boldsymbol{f}(E_0)$. In the interest of clarity, we do not explicitly write $\square\boldsymbol{f}$ since the mathematical function $\boldsymbol{f}(E_0)$ is clearer.

2.3. Normalized Taylor Coefficients

For any solution $\boldsymbol{x}$ to the ODE (1), its ith **normalized Taylor coefficient** is recursively defined as follows:

$$\boldsymbol{f}^{[i]}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} & \text{if } i = 0, \\ \frac{1}{i}\left(J_{\boldsymbol{f}^{[i-1]}} \bullet \boldsymbol{f}\right)(\boldsymbol{x}) & \text{if } i \geq 1 \end{cases} \tag{5}$$

where $J_{\boldsymbol{g}}$ denotes the Jacobian of any function $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x}) \in C^1(\mathbb{R}^n \to \mathbb{R}^n)$ in the variable $\boldsymbol{x} = (x_1, \ldots, x_n)$. For instance, $\boldsymbol{f}^{[1]} = \boldsymbol{f}$ and $\boldsymbol{f}^{[2]}(\boldsymbol{x}) = \frac{1}{2}(J_{\boldsymbol{f}} \cdot \boldsymbol{f})(\boldsymbol{x})$. It follows that the order $k \geq 1$ Taylor expansion of $\boldsymbol{x}$ at the point $t = t_0$ is

$$\boldsymbol{x}(t_0 + h) = \left\{ \sum_{i=0}^{k-1} h^i \boldsymbol{f}^{[i]}(\boldsymbol{x}(t_0)) \right\} + h^k \boldsymbol{f}^{[k]}(\boldsymbol{x}(\xi))$$

where $0 \leq \xi - t_0 \leq h$. If $\boldsymbol{x}(\xi)$ lies in a box $B \in \square\mathbb{R}^n$, then interval form is

$$\boldsymbol{x}(t_0 + h) \in \left\{ \sum_{i=0}^{k-1} h^i \boldsymbol{f}^{[i]}(\boldsymbol{x}(t_0)) \right\} + h^k \boldsymbol{f}^{[k]}(B) \tag{6}$$

These Taylor coefficients can be automatically generated, and they can be evaluated at interval values using automatic differentiation.

2.4. Banach Space X

If X, Y are topological spaces, let $C^k(X \to Y)$ $(k \geq 0)$ denote the set of C^k-continuous functions from X to Y. *We fix $\boldsymbol{f} \in C^k(\mathbb{R}^n \to \mathbb{R}^n)$ throughout the paper, and thus $k \geq 1$ is a global constant.* It follows that $\mathtt{IVP}_{\boldsymbol{f}}(B_0, h) \subseteq C^k([0, h] \to \mathbb{R}^n)$. Let $X := C^k([0, h] \to \mathbb{R}^n)$. Then X is a real linear space where $c \in \mathbb{R}$ and $\boldsymbol{x}, \boldsymbol{y} \in X$ implies $c\boldsymbol{y} \in X$ and $\boldsymbol{x} \pm \boldsymbol{y} \in X$. Let $\boldsymbol{0} \in X$ denote the additive identity in X: $\boldsymbol{x} \pm \boldsymbol{0} = \boldsymbol{x}$. X is also a normed space with norm $\|\boldsymbol{x}\| = \|\boldsymbol{x}\|_{\max} := \max_{t\in[0,h]} \|\boldsymbol{x}(t)\|_2$ where $\|\cdot\|_2$ is the 2-norm. For simplicity, write $\|\boldsymbol{x}\|$ for $\|\boldsymbol{x}\|_{\max}$. If $S \subseteq X$, we let $\|S\| := \sup_{\boldsymbol{x}\in S} \|\boldsymbol{x}\|$. We turn X into a complete metric space (X, d) with metric $d(x, y) = \|x - y\|$. To prove existence and uniqueness of solutions, we need to consider a compact subset $Y \subseteq X$. E.g., let $Y = C^k([0, h] \to B)$ where $B \subseteq \mathbb{R}^n$ is a box or ball. Then Y is also a complete metric space induced by X.

Using this theory, we prove the following fundamenal result:

Lemma 1 (Admissible Triple).

For all $k \geq 1$, if $E_0, F_1 \subseteq \mathbb{R}^n$ are closed convex sets, and $h > 0$ satisfy the inclusion

$$\sum_{i=0}^{k-1} [0, h]^i \boldsymbol{f}^{[i]}(E_0) + [0, h]^k \boldsymbol{f}^{[k]}(F_1) \subseteq F_1, \tag{7}$$

then (E_0, h, F_1) is an admissible triple.

Note that this is very similar to [26, Theorem 4.1] which requires that E_0 lies in the interior of F_1. Our result does not need this additional condition.

2.5. Logarithmic norms

Let $\|A\|_p$ be the induced p-norm of a $n \times n$ matrix A with complex entries. Then the **logarithmic p-norm** of A is defined as

$$\mu_p(A) := \lim_{h \to 0+} \frac{\|I + hA\|_p - 1}{h}.$$

We shall focus on $p = 2$, and call μ_2 the logNorm. If $n = 1$, then $A = a \in \mathbb{C}$ and $\mu_p(A) = \mathtt{Re}(a)$. We have these bounds for logNorm:

Lemma 2.

(a) $\mu_p(A + B) \leq \mu_p(A) + \mu_p(B)$

(b) $\mu_p(A) \leq \|A\|_p$

(c) $\mu_2(A) = \max_{j=1,\ldots,k}(\frac{1}{2}(\lambda_j(A + A^T)))$ *where* $\lambda_1(A), \ldots, \lambda_k(A)$ *is the set of eigenvalues of* A.

(d) Let A *be an* $n \times n$ *matrix and let* $\max_{i=1}^n(\mathtt{Re}(\lambda_i))$ *where* λ_i'*s are the eigenvalues of* A*. Then*

- $\max_{i=1}^n(\mathtt{Re}(\lambda_i)) \leq \mu(A)$ *holds for any logNorm.*
- *For any* $\varepsilon \geq 0$*, there exists an invertible matrix* P *such that*

$$\max_i(Re(\lambda_i)) \leq \mu_{2,P}(A) \leq \max_i(Re(\lambda_i)) + \varepsilon.$$

where $\mu_{2,P}(A) := \mu_2(P^{-1}AP)$.

For parts(a-c) see [35], and part(d), see Pao [36]. In our estimates, we cite these standard bounds:

$$\left.\begin{array}{lcl} \|A\|_2 & = & \max_i(\sqrt{\lambda_i(A^*A)}) \\ \|AB\|_2 & \leq & \|A\|_2\|B\|_2 \end{array}\right\} \tag{8}$$

We have the following result from Neumaier [16, Corollary 4.5] (also [37, Theorem I.10.6]:

Theorem 3 (Neumaier).
Let $\boldsymbol{x} \in \mathtt{IVP}_{\boldsymbol{f}}(\boldsymbol{p}_0, h)$ *and* $\xi(t) \in C^1([0,h] \to \mathbb{R}^n)$ *be any "approximate solution".*
Let[8] P *be an invertible matrix. Assume the constants* $\varepsilon, \delta, \overline{\mu}$ *satisfy*

1. $\varepsilon \geq \|P^{-1} \bullet (\xi'(t) - \boldsymbol{f}(\xi(t)))\|_2$ *for all* $t \in [0,h]$
2. $\delta \geq \|P^{-1} \bullet (\xi(0) - \boldsymbol{p}_0)\|_2$
3. $\overline{\mu} \geq \mu_2\big(P^{-1} \bullet J_{\boldsymbol{f}}(s\boldsymbol{x}(t) + (1-s)\xi(t)) \bullet P\big)$ *for all* $s \in [0,1]$ *and* $t \in [0,h]$

Then for all $t \in [0,h]$,

$$\|P^{-1} \bullet (\xi - \boldsymbol{x})\|_2 \leq \begin{cases} \delta e^{\overline{\mu}|t|} + \frac{\varepsilon}{\overline{\mu}}(e^{\overline{\mu}|t|} - 1), & \overline{\mu} \neq 0, \\ \delta + \varepsilon t, & \overline{\mu} = 0. \end{cases} \tag{9}$$

Corollary 4.
Let $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathtt{IVP}(B_1, h, Ball(\boldsymbol{p}_0, r))$ *and* $\overline{\mu} \geq \mu_2(J_{\boldsymbol{f}}(Ball(\boldsymbol{p}_0, r)))$*. Then for all* $t \in [0,h]$

$$\|\boldsymbol{x}_1(t) - \boldsymbol{x}_2(t)\|_2 \leq \|\boldsymbol{x}_1(0) - \boldsymbol{x}_2(0)\|_2 e^{\overline{\mu}t}. \tag{10}$$

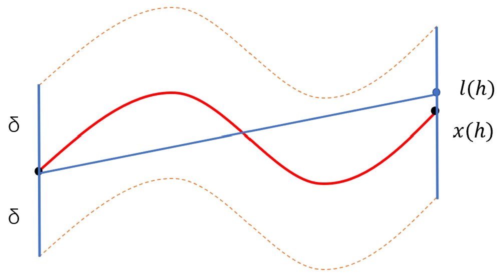

Figure 2: The dashed lines in the figure form a δ-tube around the red solid curve representing $\boldsymbol{x}(t)$. The segment $l(t)$ is a line segment inside this δ-tube.

Euler-tube Method: For any $\boldsymbol{x} \in \mathbf{IVP}(E_0, h)$ and $\delta > 0$, the δ**-tube** of $\boldsymbol{x}$ is the set

$$\mathrm{Tube}_\delta(\boldsymbol{x}) := \left\{(t, \boldsymbol{p}) : \|\boldsymbol{p} - \boldsymbol{x}(t)\|_2 \leq \delta, 0 \leq t \leq h\right\} \quad \left(\subseteq [0, h] \times \mathbb{R}^n\right)$$

We say that a function $\ell : [0, h] \to \mathbb{R}^n$ belongs to the δ-tube of $\boldsymbol{x}$ is for all $t \in [0, h]$, $(t, \ell(t)) \in \mathrm{Tube}_\delta(\boldsymbol{x})$,see graph 2 for illustration.

Lemma 5 (Euler Tube Method).
Let (B_0, H, B_1) be admissible triple, $\overline{\mu} \geq \mu_2(J_{\boldsymbol{f}}(B_1))$ and $\overline{M} \geq \|\boldsymbol{f}^{[2]}(B_1)\|$.
For any $\delta > 0$ and $h_1 > 0$ given by

$$h_1 \leftarrow h^{euler}(H, \overline{M}, \overline{\mu}, \delta) := \begin{cases} \min\left\{H, \frac{2\overline{\mu}\delta}{\overline{M}\cdot(e^{\overline{\mu}H}-1)}\right\} & \text{if } \overline{\mu} \geq 0 \\ \min\left\{H, \frac{2\overline{\mu}\delta}{\overline{M}\cdot(e^{\overline{\mu}H}-1)-\overline{\mu}^2\delta}\right\} & \text{if } \overline{\mu} < 0 \end{cases} \tag{11}$$

Consider the path $Q_{h_1} = (\boldsymbol{q}_0, \boldsymbol{q}_1, \ldots, \boldsymbol{q}_m)$ from the Euler method with uniform step-size h_1. If each $\boldsymbol{q}_i \in B_1$ $(i = 0, \ldots, m)$ then for all $t \in [0, H]$, we have

$$\|Q_{h_1}(t) - \boldsymbol{x}(t; \boldsymbol{q}_0)\| \leq \delta. \tag{12}$$

I.e., $Q_{h_1}(t)$ lies inside the δ-tube of $\boldsymbol{x}(t; \boldsymbol{q}_0)$.

This lemma allows us to refine end- and full-enclosures (see Lemma 7 below).

3. Overview of our Algorithm

We will develop an algorithm for the End-Enclosure Problem (2), by elaborating on the classic Euler method or corrector-predictor framework for homotopy path (e.g., [38, 9]). The basic motif is to repeatedly call two subroutines[9] which we call `StepA` and `StepB`, respectively:

$$\boxed{\begin{array}{l} \texttt{StepA}(E_0) \to (h, F_1): \\ \quad \text{INPUT: } E_0 \in \square\mathbb{R}^n \\ \quad \text{OUTPUT: an admissible pair } (h, F_1) \text{ for } E_0 \end{array}} \tag{13}$$

$$\boxed{\begin{array}{l} \texttt{StepB}(E_0, h, F_1) \to (E_1): \\ \quad \text{INPUT: } (E_0, h, F_1) \text{ is an admissible triple} \\ \quad \text{OUTPUT: an end-enclosure } E_1 \text{ for } \texttt{IVP}(E_0, h) \end{array}} \tag{14}$$

[8]For our purposes, matrix P in this theorem can be the identity matrix.
[9]Nediakov et al. [8] call them Algorithms I and II.

Thus we see this progression

$$E_0 \xrightarrow{\texttt{StepA}} (E_0, h_0, F_1) \xrightarrow{\texttt{StepB}} (E_0, h_0, F_1, E_1) \tag{15}$$

where StepA and StepB successively transforms E_0 to an admissible triple and quad. By iterating (15) with E_1 we can get to the next quad (E_1, h_1, F_2, E_2), and so on. *This is the basis of most validated IVP algorithms.* We encode this as:

```
standard_IVP(E_0) → B_*
    INPUT: E_0 ⊆ ℝ^n where IVP(E_0, 1) is valid.
    OUTPUT: an end-enclosure of IVP(E_0, 1).
  ------------------------------------------------
  t_1 ← 0
  F_0 ← E_0
  While (t_1 < 1)
      (h, F_1) ← StepA(E_0)
      h ← min(h, 1 − t_1)
      t_1 ← t_1 + h
      E_0 ← StepB(E_0, h, F_1)
  Return E_0
```
(16)

Note that the iteration of (15) above is not guaranteed to halt (i.e., to reach $t = 1$). Moreover, we have no control over the length of the end-enclosure. To address this, define an **ε-admissible triple** to be an admissible (E_0, h, F_1) with $h^k \boldsymbol{f}^{[k]}(F_1) \subseteq [-\varepsilon, \varepsilon]^n$. See Lemma 1 for the context of this definition. We now extend StepA to:

```
StepA(E_0, ε, H) → (h, F_1):
  INPUT: E_0 ∈ □ℝ^n, 0 < H ≤ 1 and ε > 0
  OUTPUT: an ε-admissible pair (h, F_1) for E_0
          such that h ≤ H.
```
(17)

3.1. Scaffold Framework

We introduce a data structure called a "scaffold" to encode the intermediate information needed for this computation. Figure 3 shows such a scaffold.

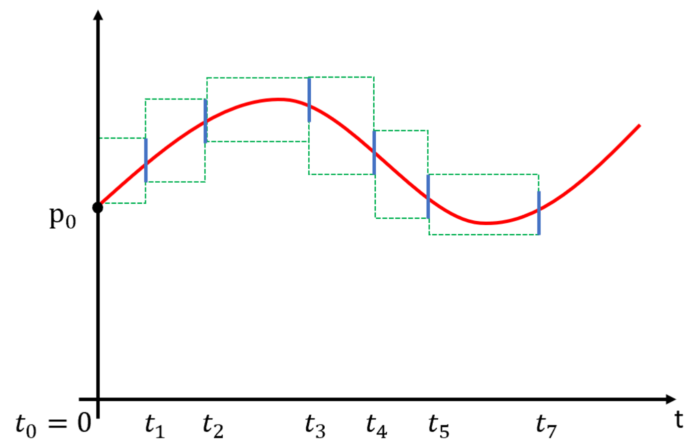

Figure 3: A 7-step scaffold. The horizontal axis represents time, and the vertical axis represents $\mathbb{R}^n$. The red curve corresponds to $\boldsymbol{x}(t)$, the blue line segments represent end-enclosures, and the green boxes, represent full-enclosures.

By a **scaffold** we mean a quad $\mathcal{S} = (\boldsymbol{t}, \boldsymbol{E}, \boldsymbol{F}, \boldsymbol{G})$ where $\boldsymbol{t} = (0 \le t_0 < t_1 < \cdots < t_m \le 1)$, $\boldsymbol{E} = (E_0, \ldots, E_m)$, $\boldsymbol{F} = (F_0, \ldots, F_m)$ and $\boldsymbol{G} = (G_0, \ldots, G_m)$ such that the following holds for all $i = 0, 1, \ldots, m$:

1. E_i is an end enclosure of $\texttt{IVP}(E_{i-1}, t_i - t_{i-1})$ for $i \geq 1$.
2. $(E_{i-1}, \Delta t_i, F_i, E_i)$ is an admissible quadruple.
3. G_i, called the **refinement structure**, is used to store other information about the ith stage (see Section 6).

For $i = 1, \ldots, m$, let $\Delta t_i := t_i - t_{i-1}$ denote the ith **step size**. Call $\mathcal{S}$ an m**-stage** scaffold where the ith **stage** of $\mathcal{S}$ includes the admissible quadruple $(E_{i-1}, \Delta t_i, F_i, E_i)$.

For $i = 0, \ldots, m$, the ith **stage** of $\mathcal{S}$ is $\mathcal{S}[i] = (t_i, E_i, F_i, G_i)$. Thus, the initial and final stages are $\mathcal{S}[0]$ and $\mathcal{S}[m]$, respectively. The **end-** and **full-enclosure** of $\mathcal{S}$ is E_m and F_m (respectively). The **time-span** of $\mathcal{S}$ is the interval $[t_0, t_m]$, and t_m is the **end-time** of $\mathcal{S}$.

A stage (t', E', F', G') is called a **refinement** of (t, E, F, G) if $t = t'$ and $E' \subseteq E$ and $F' \subseteq F$. A m'-stage scaffold $\mathcal{S}'$ is called a **refinement** of $\mathcal{S}$ if $m' = m$ and for all $i = 0, \ldots, m$, $\mathcal{S}'[i]$ is a refinement of $\mathcal{S}[i]$. A scaffold $\mathcal{S}'$ is called a **extension** of $\mathcal{S}$ if $\mathcal{S}$ is a prefix of $\mathcal{S}'$. For any $\delta > 0$, the i-th stage is δ**-bounded** if

$$r_i \leq r_{i-1} e^{\mu_2(J_f(F_i))(t_i - t_{i-1})} + \delta, \tag{18}$$

where $Ball_{p_i}(r_i)$ is the circumscribing ball of E_i. The refinement structure G_i contains a value $\delta_i > 0$, and during the computation, the ith stage is periodically made δ_i-bounded.

Next, we introduce the algorithm $\texttt{Extend}(\mathcal{S}, \varepsilon, H)$ which calls $\texttt{StepA}$ to add a new stage to $\mathcal{S}$. We view $\mathcal{S}$ is an object in the sense of OOPL, and write $\mathcal{S}.\texttt{Extend}(\cdots)$ to self-modify.

> $\mathcal{S}.\texttt{Extend}(\varepsilon_0, H)$
> INPUT: m-stage scaffold $\mathcal{S}$, $\varepsilon_0 > 0$, $H > 0$.
> OUTPUT: $\mathcal{S}'$ is a $m+1$-stage extension $\mathcal{S}$ such that
> $\Delta t_{m+1}(\mathcal{S}') \leq H$, and $(\Delta t_{m+1}(\mathcal{S}'), E_{m+1}(\mathcal{S}'))$ is an ε_0-admissible pair for $E_m(\mathcal{S})$.

(19)

To bound the length of the end enclosure, we refine $\mathcal{S}$ whenever $w_{\max}(E_m) > \varepsilon_0$. The interface for this $\texttt{Refine}$ algorithm is as follows:

> $\mathcal{S}.\texttt{Refine}(\varepsilon_0)$
> INPUT: m-stage scaffold $\mathcal{S}$, $\varepsilon_0 > 0$.
> OUTPUT: $\mathcal{S}'$ is a m-stage refinement of
> $\mathcal{S}$ satisfies $w_{\max}(E_{m'}) \leq \varepsilon_0$.

(20)

Within the $\texttt{Refine}$ procedure, when processing a stage over the interval $[t_{i-1}, t_i]$, we apply a "light-weight" refinement strategy to improve the full- and end-enclosures. Specifically, the interval is uniformly subdivided into mini-steps of size h_i, and refinement is performed on these finer subintervals. This local subdivision helps control the enclosure width without globally modifying the scaffold structure, allowing more efficient and targeted refinement where needed.

3.2. Logarithmic Norm and Radical Transformation method

We now introduce a new technique to compute enclosures more efficiently. It does not depend on Taylor expansions, but is based on logNorm estimates via Theorem 3.

Consider an admissible triple (E_0, h_1, F_1). We compute $\overline{\mu}$, a logarithmic norm bound for $(\boldsymbol{f}, F_1)$ (see (3)). When the step size $h_1 \leq h^{\text{euler}}(H, \overline{M}, \overline{\mu}, \delta)$ (see Lemma 5), we can invoke Corollary 4 to derive improved full- and end-enclosures. Moreover, the end-enclosure will be δ-bounded. The δ-bounded condition cannot be obtained from Taylor methods.

We also introduce another technique for estimation based on **radical transformation** of the original system to reduce the logarithmic norm. We distinguish two cases:

Easy Case: $\overline{\mu} \leq 0$**.** In this case, F_1 is a contraction zone. By Theorem 3, we have $w_{\max}(E_1) \leq w_{\max}(E_0)$. Therefore, we directly estimate the full- and end-enclosures using the logarithmic norm without further transformation.

Hard Case: $\overline{\mu} > 0$**.** The key idea here is to construct an invertible transformation $\pi : \mathbb{R}^n \to \mathbb{R}^n$. Let $\boldsymbol{y} = (y_1, \ldots, y_n) := \pi(\boldsymbol{x})$ and consider the transformed differential system:

$$\boldsymbol{y}' = \boldsymbol{g}(\boldsymbol{y}), \qquad \boldsymbol{g}(\boldsymbol{y}) := J_\pi(\pi^{-1}(\boldsymbol{y})) \cdot \boldsymbol{f}(\pi^{-1}(\boldsymbol{y})). \tag{21}$$

This is considered with the admissible triple $(\pi(E_0), h, \pi(F_1))$.

We define the transformation as a composition:

$$\pi = \widehat{\pi} \circ \overline{\pi}, \tag{22}$$

where $\overline{\pi}$ is an affine map (see Appendix B), and $\widehat{\pi}(\boldsymbol{x}) = (x_1^{-d_1}, \ldots, x_n^{-d_n})$ for some exponent vector $\boldsymbol{d} = (d_1, \ldots, d_n)$ to be determined. The map $\widehat{\pi}$ is invertible provided $d_i \neq 0$ for all i. Due to the component-wise inversion, we refer to π as the **radical transform**.

Assuming $\texttt{IVP}(B_0, h, \boldsymbol{B}_1)$ is valid (Section 2.1), and that $\boldsymbol{B}_1$ is sufficiently small, we can show that $\pi(\boldsymbol{B}_1)$ is a contraction zone for $(\pi(\boldsymbol{B}_1), \boldsymbol{g})$. This brings the problem back to the easy case. After computing a shrunken enclosures in the transformed space, we pull it back to obtain an enclosures for the original IVP. For consistency, in the easy case, we define $(\pi, \boldsymbol{g})$ as $(\mathrm{Id}, \boldsymbol{f})$.

4. Steps A and B

Nedialkov et al [26, 8] provide a careful study of various algorithms for `StepA` and `StepB`. In the following, we provide new forms of `StepA` and `StepB`.

4.1. Step A

We now provide the subroutine $\texttt{StepA}(E_0, H, \varepsilon)$. Its input/output specification has been given in (17). Basically, we can regard its main goal as computing the largest possible $h > 0$ ($h \leq H$) such that (E_0, h, F_1) is ε-admissible for some F_1. When calling `StepA`, we are at some time $t_1 \in [0, 1)$, and so the largest h needed is $H = 1 - t_1$. We therefore pass this value H to our subroutine. In contrast, [26, p.458, Figure 1], uses a complicated formula for H based the previous step.

Lemma 6.

Let $H > 0$, $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$, and $E_0 \subseteq \mathbb{R}^n$. If

$$\overline{B} := \sum_{i=0}^{k-1} [0, H]^i \boldsymbol{f}^{[i]}(E_0) + Box(-\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}) \quad \textit{and} \quad M := \sup_{p \in \overline{B}} \|\boldsymbol{f}^{[k]}(\boldsymbol{p})\|_2,$$

then an $\boldsymbol{\varepsilon}$-admissible pair for E_0 is given by (h, F_1) where

$$h = \min\left\{ H, \min_{i=1}^{n} \left(\frac{\varepsilon_i}{M_i}\right)^{1/k} \right\} \quad \textit{and} \quad F_1 = \sum_{i=0}^{k-1} [0, h]^i \boldsymbol{f}^{[i]}(E_0) + Box(-\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}). \tag{23}$$

Using Lemma 6, we can define $\texttt{StepA}(E_0, H, \varepsilon)$ as computing (h, F_1) as given by (23). Call this the **non-adaptive** `StepA`, denoted $\texttt{StepA}_0$. The non-adaptive h may be too pessimistic. Instead, we propose to compute h adaptively:

> `StepA`$(E_0, H, \varepsilon) \to (h, F_1)$
> INPUT: $E_0 \in \square\mathbb{R}^n$, $H > 0$, $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$
> OUTPUT: $0 < h \le H$, $F_1 \in \square\mathbb{R}^n$ such that (E_0, h, F_1) is ε-admissible.
>
> ---
>
> $h \leftarrow 0$
> While $(H > 2h)$
> $\quad \overline{B} \leftarrow Box\big(\sum_{i=0}^{k-1} [0, H]^i \boldsymbol{f}^{[i]}(E_0)\big) + Box(-\varepsilon, \varepsilon)$
> $\quad \boldsymbol{M} \leftarrow \boldsymbol{w}(Box(\boldsymbol{f}^{[k]}(\overline{B})))$
> $\quad h \leftarrow \min_{i=1}^{n} \left(\frac{\varepsilon_i}{M_i}\right)^{1/k}$ (where $\boldsymbol{M} = (M_1, \ldots, M_n)$)
> $\quad H \leftarrow H/2$
> $F_1 \leftarrow \overline{B}$.
> Return (h, F_1)

In the while-loop of `StepA`, when $H > 2h$, we reduce H to compute a larger value of h. This is an adaptive step size search that adjusts H in order to maximize h under the constraint of satisfying Lemma 6. The resulting value of h is theoretically a factor of 2 from the optimal. The summation $\sum_{i=0}^{k-1}$ in `StepA` should be evaluated with Horner's rule (see [26, p. 458]).

4.2. Step B

For `Step B`, there are several methods such as the "Direct Method" [25, 8], Lohner's method [12], and C^1-Lohner method [39]. The Direct Method, on input (E_0, h, F_1) returns the following end-enclosure

$$E_1 = \underbrace{\sum_{i=0}^{k-1}(h^i \boldsymbol{f}^{[i]}(m(E_0)) + h^k \boldsymbol{f}^{[k]}(F_1)}_{\text{Part(m)}} + \underbrace{(\sum_{i=0}^{k-1} h^i J_{\boldsymbol{f}^{[i]}}(E_0)) \bullet (E_0 - m(E_0))}_{\text{Part(r)}} \tag{24}$$

where Part(m) tracks the midpoint $m(E_0)$ and Part(r) is the correction factor for Part(m). Let us define $\texttt{StepB}_0(E_0, h, F_1)$ as return the value E_1 in (24). We may also call $\texttt{StepB}_0$ the **direct method**.

Both the Lohner and the C^1-Lohner methods are refinements of the Direct method. The Lohner method aims to reduce the wrapping effect introduced in Part(r). The C^1-Lohner method goes further by considering the limit when $k \to \infty$: then $V := \left(\sum_{i=0}^{\infty} h^i J_{\boldsymbol{f}^{[i]}}\right)$ satisfies another ODE: $V' = J_{\boldsymbol{f}} \cdot V$. By solving this ODE, the method effectively reduces the overall error. We will also use the logarithmic norm to estimate the range by Corollary 4, and combine it with the Direct Method in `StepB`:

> `StepB`$((E_0, h, F_1, \mu)) \to E_1$
> INPUT: A admissible triple (E_0, h, F_1) and
> the logNorm$\mu = \mu_2(J_{\boldsymbol{f}}(F_1))$.
> OUTPUT: E_1 is an end-enclosure for (E_0, h, F_1).
>
> ---
>
> $\boldsymbol{p} \leftarrow \sum_{i=0}^{k-1}(h^i \boldsymbol{f}^{[i]}(m(E_0)) + h^k \boldsymbol{f}^{[k]}(F_1)$.
> $r_0 \leftarrow \frac{1}{2} w_{\max}(E_0)$.
> $E_1 \leftarrow \boldsymbol{p} + (\sum_{i=0}^{k-1} h^i J_{\boldsymbol{f}^{[i]}}(E_0)) \bullet (E_0 - m(E_0)) \cap Box_{\boldsymbol{p}}(r_0 e^{\mu h})$.
> Return E_1.

We can also use the range estimation provided by Corollary 4 , in combination with Lohner-type methods.

4.3. Refinement Technique for Full and End Enclosures of a Stage

Let $(E_{i-1}, \Delta t_i, F_i, E_i)$ be an admissible quad for the ith stage. We now introduce a "light weight" technique to refine F_i, E_i using Euler's method.

Eg*	Name	$f(x)$	Parameters	Box B_0	Reference
Eg1	Volterra	$\begin{pmatrix} ax(1-y) \\ -by(1-x) \end{pmatrix}$	$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$	$Box_{(1,3)}(0.1)$	[40], [13, p.13]
Eg2	Van der Pol	$\begin{pmatrix} y \\ -c(1-x^2)y-x \end{pmatrix}$	$c=1$	$Box_{(-3,3)}(0.1)$	[13, p.2]
Eg3	Asymptote	$\begin{pmatrix} x^2 \\ -y^2+7x \end{pmatrix}$	N/A	$Box_{(-1.5,8.5)}(0.01)$	N/A
Eg4	Lorenz	$\begin{pmatrix} \sigma(y-x) \\ x(\rho-z)-y \\ xy-\beta z \end{pmatrix}$	$\begin{pmatrix} \sigma \\ \rho \\ \beta \end{pmatrix} = \begin{pmatrix} 10 \\ 28 \\ 8/3 \end{pmatrix}$	$Box_{(15,15,36)}(0.001)$	[13, p.11]

Table 1
List of IVP Problems

Lemma 7 (Euler Enclosures with logNorm).
Consider an admissible triple (E_0, H, F_1) where $E_0 := Ball(\boldsymbol{p}_0, r_0)$.
Let $\boldsymbol{q}_0 = \boldsymbol{p}_0 + h_1 f(\boldsymbol{p}_0)$ be obtained from $\boldsymbol{p}_0$ by an Euler step of size h_1.
If $h_1 \le h^{euler}(H, \overline{M}, \overline{\mu}, \delta)$ (cf. (11)), where $\overline{\mu} = \mu_2(J_{\boldsymbol{f}}(F_1))$, $\overline{M} = \|\boldsymbol{f}^{[2]}(F_1)\|$, and $\delta > 0$, then:

(a) The linear function $\ell(t) := (1 - t/h_1)\boldsymbol{p}_0 + (t/h_1)\boldsymbol{q}_0$ lies in the δ-tube of $\boldsymbol{x}_0 = \mathtt{IVP}(\boldsymbol{p}_0, H)$.

(b) An end-enclosure for $\mathtt{IVP}(E_0, h_1)$ is given by $Ball(\boldsymbol{q}_0, r_0 e^{\overline{\mu} h_1} + \delta)$.

(c) A full-enclosure for $\mathtt{IVP}(E_0, h_1)$ is given by $CHull(Ball(\boldsymbol{p}_0, r'), Ball(\boldsymbol{q}_0, r'))$ where $r' = \delta + \max(r_0 e^{\overline{\mu} h_1}, r_0)$.

Key idea of the refinement strategy for a stage: suppose stage $S[i]$ is represented by the admissible triple (E_0, H, F_1), and we have a given target $\delta_i > 0$. The goal is to compute a δ_i-bounded end-enclosure for this stage. Using the above lemma, we can compute a $h_i = h^{\text{euler}}(\cdots, \delta_i)$ (see (11)) such that an Euler path with uniform step size h_i will produce δ_i-bounded end-enclosure for stage i. Call this the `EulerTube` Subroutine. Unfortunately, this is too inefficient when $\overline{M}, \overline{\mu}$ is large. We therefore introduce an adaptive method called Bisection to reduce $\overline{M}, \overline{\mu}$:

- Bisection Method: we subdivide the interval $[0, H]$ into 2^ℓ mini-steps of size $h_\ell := H/2^\ell$ (for $\ell = 1, 2, \ldots$). At each **level** ℓ, we can compute full- and end-enclosures $(\boldsymbol{F}_i[j], \boldsymbol{E}_i[j])$ of the jth mini-step ($j = 1, \ldots, 2^\ell$) using the following formula:

$$\boldsymbol{F}_i[j] \leftarrow \sum_{p=0}^{k-1} [0, h_\ell]^p \boldsymbol{f}^{[p]}(\boldsymbol{E}_i[j-1] + [0, h_\ell]^k \boldsymbol{f}^{[k]}(F_1), \tag{25}$$

and

$$\boldsymbol{E}_i[j] \leftarrow \mathtt{StepB}(\boldsymbol{E}_i[j-1], h_\ell, \boldsymbol{F}_i[j], \mu_2(\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{F}_i[j]))). \tag{26}$$

- When ℓ is sufficiently large, i.e., $h_\ell \le h_i$, then we can call the `EulerTube` subroutine above. Our experiments show, this subroutine is more accurate.

4.4. List of Problems and Local Experiments on Steps A and B

Table 1 is a list of problems used throughout this paper for our experiments. Here we will give "local" (single-step) experiments on the effectiveness our Steps A and B. Later in Section 7, we will do "global" experiments based on our overall algorithm. We measure each technique by ratios denoted by σ, such that $\sigma > 1$ shows the effectiveness of the technique. Note that the gains for local experiments may appear small (e.g., 1.0001). But in global m-step experiment, this translates to $(1.0001)^m$ which can be significant.

First, in Table 2, we compare our `StepA` with the non-adaptive `StepA`$_0$. This non-adaptive `StepA`$_0$ is basically the algorithm[10] in [26, p.458, Figure 1].

[10] We replace their $h_{j,0}$ by H, and $2h^k_{j,0}\boldsymbol{f}^{[k]}([\widetilde{y}_{j-1}])$ by $\boldsymbol{\varepsilon}$.

Eg	E_0	ε	H	σ	τ	ρ
Eg1	$[0.9, 1.1], [2.9, 3.1]$	0.1	1.0	38.5	1.15	3.05
			10	2.22×10^6	1.06	3.84
		0.0001	1.0	56.8	1.02	5.26
			10	2.17×10^6	1.27	3.62
	$[2, 4], [3, 5]$	0.1	1.0	3.89×10^4	1.03	2.41
			10	1.89×10^8	1.18	3.17
		0.0001	1.0	4.97×10^3	1.01	1.52
			10	2.65×10^8	1.45	1.72
Eg2	$[-3.1, -2.9], [2.9, 3.1]$	0.1	1.0	5.12×10^3	14.6	8.49
			10	3.50×10^{12}	10.2	13.1
		0.0001	1.0	5.58×10^3	1.00	8.49
			10	4.24×10^{12}	7.35	10.4
	$[-4, -2], [3, 5]$	0.1	1.0	3.69×10^5	1.08	5.21
			10	1.90×10^{14}	1.19	7.41
		0.0001	1.0	4.56×10^5	1.0	2.58
			10	2.60×10^{14}	1.49	3.16
Eg3	$[-1.51, -1.49], [8.49, 8.51]$	0.1	1.0	2.41×10^4	1.83	65.2
			10	2.24×10^9	1.49	30.5
		0.0001	1.0	3.23×10^4	1.005	129
			10	2.28×10^9	1.81	161
	$[-3.5, -3.4], [8.4, 8.5]$	0.1	1.0	4.30×10^4	1.36	21.1
			10	3.00×10^9	1.11	26.2
		0.0001	1.0	4.36×10^4	1.20	31.2
			10	3.05×10^9	1.20	38.7
Eg4	$[14.999, 15.001], [14.999, 15.001], [35.999, 36.001]$	0.1	1.0	2.98×10^3	1.00	59.9
			10	1.81×10^8	1.04	80.5
		0.0001	1.0	3.83×10^3	1.01	2050.88
			10	2.84×10^8	1.03	1356.44
	$[12, 15], [13, 15], [34, 36]$	0.1	1.0	1.63×10^4	1.04	4.05
			10	8.70×10^8	1.08	5.52
		0.0001	1.0	1.95×10^4	1.05	2.05
			10	1.09×10^9	1.07	2.51

Table 2
Comparison of `StepA` with $\texttt{StepA}_0$. Each row of the table is an experiment with one of our examples (Eg1, Eq2, etc), with the indicated values of (E_0, H, ε). The key column is labeled $\sigma = h/h_0$, giving the ratio of the adaptive step size over the non-adaptive size.

Let (h_0, F_0) and t_0 be the admissible pair and computing time for $\texttt{StepA}_0(E_0, H, \varepsilon)$. Let $(h, F), t$ be the corresponding values for $\texttt{StepA}(E_0, H, \varepsilon)$. The performance of these 2 algorithms can be measured by three ratios:

$$\rho := \frac{w_{\max}(F)}{w_{\max}(F_0)}, \qquad \sigma := \frac{h}{h_0}, \qquad \tau := \frac{t}{t_0}.$$

The most important ratio is σ, which we want to be as large as possible and > 1. A large σ will make ρ and τ to be > 1, which is not good when viewed in isolation. But such increases in ρ and τ, in moderation, is a good overall tradeoff.

Table 2 shows that `StepA` can dramatically increase the step size h without incurring a significant increase in computation time. So the adaptive version is highly effective and meaningful.

The Table 3, we combine two comparisons:

1. $\sigma_1 = \frac{w_{\max}(E_1)}{w_{\max}(E_2)}$ compares the maximum width from the C^r-Lohner algorithm (E_1) with that of the combined method based on Corollary 4 , where $E_2 = E_1 \cap B_{\boldsymbol{p}}(r_0 e^{\mu T})$, r_0 is the radius of the initial enclosure, and $\boldsymbol{p}$ is the traced point(Part(m) of (24).

2. $\sigma_2 = \frac{w_{\max}(DB_1)}{w_{\max}(B_1)}$ compares the maximum width from the Direct method (DB_1) with that from `StepB` (B_1).

For each example, we provide an admissible triple and compute the logarithmic norm $\mu \ge \mu_2(J_f(F_1))$.

The data in the Table 3 show that intersecting either the C^r-Lohner method or the Direct method with the estimate from Corollary 4 leads to tighter enclosures, with the improvement being especially pronounced for the Direct method. This effect becomes more noticeable as the step size increases.

The Table 4 compares various examples under a given (E_0, H, F_1), showing the values of $h_1 = h^{\text{euler}}(H, \overline{M}, \overline{\mu}, \delta)$ computed for different choices of δ (see (11)). It also reports the ratio of the maximum widths of the full enclosures obtained using Lemma 7 and (25), respectively.

The data in Table 4 demonstrate that our method described in Lemma 7 yields a better full enclosure than the one obtained from (25). It is worth emphasizing that updating the full enclosure is important, as it allows us to reduce the value of logNorm, which in turn enables further tightening of the end enclosure during subsequent refinement steps.

Eg*	E_0	F_1	h	μ	σ_1	σ_2
Eg1	$Box_{(0.6,1.2)}(0.01)$	$(0.58, 1.17) \pm (0.03, 0.04)$	0.10	-0.23	1.10	1.13
		$(0.575, 1.135) \pm (0.065, 0.075)$	0.22	-0.03	1.16	1.41
		$(0.57, 1.105) \pm (0.16, 0.135)$	0.34	0.28	1.11	2.88
	$Box_{(0.6,1.2)}(10^{-4})$	$(0.585, 1.175) \pm (0.015, 0.025)$	0.10	-0.27	1.10	1.10
		$(0.57, 1.115) \pm (0.08, 0.095)$	0.33	0.09	1.15	5.10
		$(0.57, 1.105) \pm (0.14, 0.125)$	0.37	0.26	1.11	16.09
Eg2	$Box_{(-3,3)}(0.1)$	$(-3.00, 2.96) \pm (0.105, 0.14)$	0.003	7.18	1.01	1.00
		$(-2.92, 2.47) \pm (0.22, 1.13)$	0.05	10.67	1.00	1.02
		$(-2.895, 2.265) \pm (0.295, 2.005)$	0.08	14.33	1.00	1.07
	$Box_{(-3,3)}(10^{-4})$	$(-2.985, 2.925) \pm (0.015, 0.075)$	0.006	6.03	1.01	1.03
		$(-2.895, 2.35) \pm (0.185, 1.57)$	0.085	11.79	1.00	2.25
		$(-2.895, 2.265) \pm (0.295, 2.005)$	0.09	12.66	1.00	2.57
Eg3	$Box_{(-1.5,8.5)}(0.001)$	$(-1.495, 8.475) \pm (0.015, 0.035)$	0.0005	-2.12	1.00	1.01
		$(-1.49, 8.315) \pm (0.02, 0.205)$	0.004	-1.99	1.00	1.10
		$(-1.445, 7.055) \pm (0.065, 3.115)$	0.04	0.37	1.00	2.07
	$Box_{(-1.5,8.5)}(10^{-4})$	$(-1.495, 8.475) \pm (0.005, 0.025)$	0.0005	-2.15	1.00	1.02
		$(-1.495, 8.31) \pm (0.005, 0.20)$	0.004	-2.08	1.00	1.50
		$(-1.445, 7.055) \pm (0.055, 3.105)$	0.04	0.34	1.00	8.61
Eg4	$Box_{(15,15,36)}(0.001)$	$(14.855, 13.475, 37.085) \pm (0.145, 1.595, 1.815)$	0.024	3.19	1.35	1.52
		$(14.74, 12.885, 37.23) \pm (0.28, 2.325, 2.27)$	0.027	3.67	1.36	1.80
		$(14.665, 12.58, 37.275) \pm (0.375, 2.74, 3.215)$	0.031	3.98	1.37	2.58
	$Box_{(15,15,36)}(10^{-4})$	$(14.855, 13.475, 37.085) \pm (0.145, 1.595, 1.815)$	0.020	3.19	1.33	1.79
		$(14.80, 13.16, 37.23) \pm (0.21, 1.97, 2.27)$	0.024	3.42	1.35	3.18
		$(14.665, 12.58, 37.275) \pm (0.375, 2.74, 3.215)$	0.031	3.98	1.37	13.52

Table 3
Comparison of StepB with the Direct method and the C^r-Lohner algorithm. The key column is $\sigma_2 = \frac{w_{\max}(DB_1)}{w_{\max}(B_1)}$, which reflects the ratio of the maximum width produced by the Direct method (DB_1) to that by StepB (B_1), serving as a direct measure of their relative tightness. We also report σ_1 which compares the maximum width from the C^r-Lohner algorithm with that of the combined method based on Corollary 4.

Eg*	(E_0, H, F_1)			δ	h_1	μ	σ
	E_0	H	F_1				
Eg1	$(1.0, 3.0) \pm (0.1, 0.1)$	0.1	$(0.745, 2.955) \pm (0.455, 0.295)$	0.1	0.08	1.31	1.73
				0.01	0.008	1.31	1.09
				0.001	0.0008	1.31	1.01
Eg2	$(-3.0, 3.0) \pm (0.1, 0.1)$	0.05	$(-2.92, 2.40) \pm (0.28, 0.80)$	0.1	0.019	9.57	1.62
				0.01	0.0019	9.57	1.10
				0.001	0.00019	9.57	1.01
Eg3	$(-1.50, 8.50) \pm (0.01, 0.01)$	0.04	$(-1.445, 6.635) \pm (0.165, 1.975)$	0.1	0.0059	-0.0026	2.48
				0.01	0.00059	-0.0026	1.75
				0.001	0.000059	-0.0026	1.14
Eg4	$(15.000, 15.000, 36.000) \pm (0.001, 0.001, 0.001)$	0.027	$(14.736, 12.800, 37.279) \pm (0.365, 2.301, 2.442)$	0.1	0.0026	3.455	1.84
				0.01	0.00026	3.455	1.74
				0.001	0.000026	3.455	1.41

Table 4
Comparison of Full-Enclosures from Lemma 7 and (25). $\sigma := \frac{w_{\max}(F_0)}{w_{\max}(F)}$, where F is the enclosure computed via Lemma 7, and F_0 is the one obtained using (25).

5. Tighter Enclosures using Transformation

In the previous section, we used the logNorm in combination with the Taylor method to obtain tighter enclosures. However, the earlier approach has two main issues:

1. It may only reduce the maximum width of the enclosure, without considering the minimum width.

   For example, consider the ODE system $(x', y') = (7x, y)$, which consists of two independent one-dimensional subsystems. When analyzing this as a two-dimensional system, the logarithmic norm depends only on the component $x' = 7x$, since the logarithmic norm takes the maximum value.

2. For methods like the Direct method—which first track the midpoint and then estimate the range—there is a potential problem: the tracked midpoint can deviate significantly from the true center of the solution set. This deviation may lead to considerable overestimation in the resulting enclosure.

A radical map can be used to address these issues as suggested in our introduction.

Consider an admissible triple (E_0, h, F_1). By the validity of $\mathtt{IVP}(E_0, h)$, the following condition can be achieved if E_0 sufficiently shrunk:

$$\mathbf{0} \notin \overline{F}_1 := Box(\boldsymbol{f}(F_1)) = \prod_{i=1}^{n} \overline{I}_i. \tag{27}$$

This implies that there exists some $i = 1, \ldots, n$ such that $0 \notin \overline{I}_i$. We need such a condition because the radical map (4) is only defined if each $x_i > 0$, which we can achieve by an affine transformation $\overline{\pi}$. Recall in Subsection 3.2 that in the hard case, we compute the map $\pi = \widehat{\pi} \circ \overline{\pi}$. Define the box B_2 and $\check{b}_{\max}$

$$B_2 := Box(\overline{\pi}(F_1)) = \prod_{i=1}^{n} [1, \check{b}_i]. \quad \check{b}_{\max} := \max_{i=1,\ldots,n} \check{b}_i. \tag{28}$$

Using $\overline{\pi}$, we can introduce an intermediate ODE system with new differential variables $\overline{\boldsymbol{y}} := \overline{\pi}(\boldsymbol{x})$ and algebraic function $\overline{\boldsymbol{g}}(\overline{\boldsymbol{y}}) := J_{\overline{\pi}} \bullet \boldsymbol{f}(\overline{\pi}^{-1}(\overline{\boldsymbol{y}}))$ satisfying the ODE: $\overline{\boldsymbol{y}}' = \overline{\boldsymbol{g}}(\overline{\boldsymbol{y}})$ and

$$\overline{\boldsymbol{g}}(\overline{\pi}(F_1)) \geq \mathbf{1} = [1, \ldots, 1]. \tag{29}$$

Note that $(\pi(E_0), h, \pi(F_1))$ is an admissible triple in the $(\boldsymbol{y}, \boldsymbol{g})$-space.

Theorem 8 (Radical Transform).

(a) For any $\boldsymbol{d} = (d_1, \ldots, d_n)$, *we have*

$$\begin{aligned} \mu_2\big(J_{\boldsymbol{g}}(\pi(F_1))\big) \quad \leq \quad & \max\left\{ \frac{-(d_i+1)}{\check{b}_i} \; : \; i = 1, \ldots, n \right\} \\ & + \max_{i=1}^{n} \left\{ d_i \right\} \cdot \|J_{\overline{\boldsymbol{g}}}(\overline{\pi}(F_1))\|_2 \cdot \max_{i=1}^{n} \left\{ \frac{(\check{b}_i)^{d_i+1}}{d_i} \right\}. \end{aligned}$$

(b) If $d_1 = \cdots = d_n = d$ *then*

$$\mu_2\left(J_{\boldsymbol{g}}(\pi(F_1))\right) \leq -(d+1)\frac{1}{\check{b}_{\max}} + (\check{b}_{\max})^{d+1} \|J_{\overline{\boldsymbol{g}}}(\overline{\pi}(F_1))\|_2.$$

Until now, the value of $\boldsymbol{d}$ in the radical map $\widehat{\pi}$ was arbitrary. We now specify $\boldsymbol{d} = \boldsymbol{d}(F_1)$. The definition of $\boldsymbol{d}$ is motivated by Theorem 8. The optimal choice of $\boldsymbol{d}$ is not obvious. So we make a simple choice by restricting $d_1 = \cdots = d_n = d$. In this case, we could choose the upper bound of d:

$$\overline{d}(F_1) := \max\left\{1, \quad 2\|J_{\overline{\boldsymbol{g}}}(\overline{\pi}(F_1))\|_2 - 1\right\}. \tag{30}$$

Lemma 9. *If* $d \geq \overline{d}(F_1)$, *we have:*

(a) $\mu_2\left(J_{\boldsymbol{g}}(\pi(F_1))\right) \leq (-2 + (\check{b}_{\max})^{d+2}) \cdot \frac{\|J_{\overline{\boldsymbol{g}}}(\overline{\pi}(F_1))\|_2}{\check{b}_{\max}}$.

(b) If $\log_2(\check{b}_{\max}) < \frac{1}{d+2}$ *then* $\mu_2\left(J_{\boldsymbol{g}}(\pi(F_1))\right) < 0$.

To use this lemma, we first check if choosing d to be $\overline{d}(F_1)$ satisfies $\mu_2\left(J_{\boldsymbol{g}}(\pi(F_1))\right) < 0$. If so, we perform a binary search over $d \in [1, \overline{d}(F_1)]$ to find an integer d such that $\mu_2\left(J_{\boldsymbol{g}}(\pi(F_1))\right)$ is negative and as close to zero as possible. Otherwise, $\pi = \mathrm{Id}$. As seen in Subsection 5.2, this is a good strategy.

Given an admissible triple (E_0, h, F_1), we introduce a subroutine called `Transform`$(\boldsymbol{f}, F_1)$ to convert the differential equation $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x})$ to $\boldsymbol{y}' = \boldsymbol{g}(\boldsymbol{y})$ according to above map π. However, this transformation depends on the condition (27). So we first define the following predicate `AvoidsZero`$(\boldsymbol{f}, F_1)$:

> `AvoidsZero`$(\boldsymbol{f}, F_1)$ → `true` or `false`.
> INPUT: $F_1 \subseteq \square\mathbb{R}^n$.
> OUTPUT: `true` if and only if $\mathbf{0} \notin \mathrm{Box}(\boldsymbol{f}(\mathbf{F_1}))$.

Now we may define the transformation subroutine:

$\texttt{Transform}(\boldsymbol{f}, F_1, \overline{\mu}_1) \to (\pi, \boldsymbol{g}, \overline{\mu})$
INPUT: $F_1 \subseteq \square\mathbb{R}^n$ and $\overline{\mu}_1 \geq \mu_2(J_{\boldsymbol{f}}(F_1))$.
OUTPUT: $(\pi, \mu, \boldsymbol{g})$ where
π and $\boldsymbol{g}$ satisfy (22) and (21).

If ($\texttt{AvoidsZero}(\boldsymbol{f}, F_1)\texttt{=false}$ & $\overline{\mu}_1 \leq 0$)
Return (Id, $\boldsymbol{f}, \overline{\mu}_1$)
Compute $\overline{\pi}$ to satisfy (29)
Compute π and $\boldsymbol{g}$ according (22) and (21).
$\overline{\mu} \leftarrow \mu_2(J_{\boldsymbol{g}}(\pi(B)))$.
Return $(\pi, \boldsymbol{g}, \overline{\mu})$

5.1. Transformation of Error Bounds

We want to compute a transformation $\delta_x \mapsto \delta_y$ such that if B is a δ_y-bounded end-enclosure for $(\pi(E_0), h, \pi(F_1))$ in the $(\boldsymbol{y}, \boldsymbol{g})$-space, then $\pi^{-1}(B)$ is a δ_x-bounded end-enclosure of (E_0, h, F_1) in the $(\boldsymbol{x}, \boldsymbol{f})$-space. The following lemma achieves this:

Lemma 10.
Let $\boldsymbol{y} = \pi(\boldsymbol{x})$ *and*

$$\begin{aligned} \boldsymbol{x} &= IVP_{\boldsymbol{f}}(\boldsymbol{x}_0, h, F_1), \\ \boldsymbol{y} &= IVP_{\boldsymbol{g}}(\pi(\boldsymbol{x}_0), h, \pi(F_1)). \end{aligned}$$

For any $\delta_x > 0$ *and any point* $\boldsymbol{p} \in \mathbb{R}^n$ *satisfying*

$$\|\pi(\boldsymbol{p}) - \boldsymbol{y}(h)\|_2 \leq \delta_y := \frac{\delta_x}{\|J_{\pi^{-1}}(\pi(F_1))\|_2}, \tag{31}$$

we have

$$\|\boldsymbol{p} - \boldsymbol{x}(h)\|_2 \leq \delta_x.$$

$\texttt{TransformBound}(\delta, \pi, F_1) \to \delta'$
INPUT: $\delta > 0$, π, F_1 as above.
OUTPUT: $\delta' > 0$ satisfying the Lemma 10.

If (π is the identity map)
Return δ.
Else
Return $\frac{\delta}{\|J_{\pi^{-1}}(\pi(F_1))\|_2}$.

5.2. Enclosures via Transformation

Let (E_0, h, F_1) be an admissible triple that has been transformed into $(\pi(E_0), h, \pi(F_1))$. Let $\overline{\mu}^1$ be the logNorm bound for $(\boldsymbol{f}, F_1)$ and $\overline{\mu}^2$ be the corresponding bound for $(\boldsymbol{g}, \pi(F_1))$. Given $\delta_x > 0$, if we trace $m = m(E_0)$ to get a point $\boldsymbol{Q}$ such that $\|\boldsymbol{Q} - \boldsymbol{x}(h; m)\| \leq \delta_x$ then

$$E_1^{\text{std}} := Box_{\boldsymbol{Q}}\left(r_0 e^{\overline{\mu}^1 h} + \delta_x\right). \tag{32}$$

as an end-enclosure for $\texttt{IVP}(E_0, h, F_1)$, r_0 radius of the circumball of E_0. Using our π-transform we can first compute a point $\boldsymbol{q}$ such that $\|\boldsymbol{q} - \boldsymbol{y}(h)\| \leq \delta_y$ and take its inverse, or we can take the inverse of the end-enclosure in $(\boldsymbol{y}, \boldsymbol{g})$-space. These two methods give us two end-enclosures:

$$E_1^{\text{xform1}} := Box_{\pi^{-1}(\boldsymbol{q})}\left(r_0 e^{\overline{\mu}^1 h} + \delta_x\right),$$

$d \backslash e$	-2.5	-2.0	-1.5	-1.0	-0.5	0.5	1.0	1.5	2.0	2.5
-3.5	52.6123	1.0000	1.0000	1.0000	1.0000	1.0000	1	1.0000	1.0000	1.0000
-3.0	23.8482	22.1158	1.0000	1.0000	1.0000	1.0000	1	1.0000	1.0000	1.0000
-2.5	10.8027	10.2084	9.3113	1.0000	1.0000	1.0000	1	1.0000	1.0000	1.0000
-2.0	4.8901	4.7089	4.4287	3.9948	1.0000	1.0000	1	1.0000	1.0000	1.0000
-1.5	2.2121	2.1707	2.1051	1.9992	1.8276	1.0000	1	1.0000	1.0000	1.0000
-1.0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1	1.0000	1.0000	1.0000
-0.5	1.0000	1.0000	1.0000	1.0000	1.0000	1.5647	1	1.0308	1.0168	1.0000
0.5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1	1.0798	1.0464	1.0037
1.0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1	1.0611	1.0594	1.0075
1.5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1	1.0474	1.0492	1.0118

Table 5
This is a table of the ratios $W(e,d)$ using our transform subroutines. The red entries are maximal for each column, and correspond to the choice $d = e - 1$. Note that we exclude the column for $e = 0$ since the ODE $x' = x^e = 1$ is independent of x. We also excluded the row for $d = 0$ as the radical transform $y = x^d = 1$ makes y independent of x. The column for $e = 1$ is literally 1 (other values written "1.0000" are generally approximations).

$$E_1^{\text{xform2}} := \pi^{-1}\left(Box_q((r_0' + d_m)e^{\overline{\mu}^2 h} + \delta_y)\right),$$
$$E_1^{\text{xform}} := E_1^{\text{xform1}} \cap E_1^{\text{xform2}} \tag{33}$$

where r_0' is the radius of the circumball of $\pi(E_0)$ and $d_m \geq \|\pi(m(E_0)) - m(\pi(E_0))\|$. To motivate these transforms, the following will analyze the situation in the special case $n = 1$.

Example 1 (Benefits of Transform ($n = 1$)). Consider the ODE $x' = x^e$ ($e \neq 0$) for e real with corresponding valid $\texttt{IVP}(B_0, h)$ where $B_0 = 0.2 \pm 0.04$ and $h = 1$. Apply the radical transform $y = x^{-d}$ for some real $d \neq 0$. Then we see that $y' = \frac{d}{dx}\left(x^{-d}\right) \cdot x' = -d\, y^{\frac{-e+1+d}{d}}$. Let $W(e,d) := \frac{w_{\max}(E_1^{\text{std}})}{w_{\max}(E_1^{\text{xform}})}$ denote the ratio of the widths of the end-enclosure using (32) and (33). Table 5 shows that the maximum value of $W(e,d)$ for a fixed $e \neq 1$ is achieved when $d = e - 1$, i.e., $y' = -d$.

■

5.2.1. Local Experiments on Transform Methods

We will compare E_1^{std} and E_1^{xform} using two independent ratios:

$$\rho(E_1^{\text{std}}, E_1^{\text{xform}}) := \left(\frac{w_{\max}(E_1^{\text{std}})}{w_{\max}(E_1^{\text{xform}})}, \frac{w_{\min}(E_1^{\text{std}})}{w_{\min}(E_1^{\text{xform}}))}\right). \tag{34}$$

Our current experiments shows that the first ratio in $\rho(E_1^{\text{std}}, E_1^{\text{xform}})$ is always less than the second ratio, and for simplicity, we only show the second ratio, which is denoted by $\sigma(E_1^{\text{std}}, E_1^{\text{xform}})$ in the last column of Table 6.

Table 6 compares a single step of our transform method with the Standard method (32).

Each row represents a single experiment. The columns under (E_0, F_1, h) represent an admissible triple. The column under $\overline{\mu}^1$ (resp. $\overline{\mu}^2$) represents the logNorm bound of F_1 in the $(\boldsymbol{x}, \boldsymbol{f})$-space (resp. $\pi(F_1)$ in the $(\boldsymbol{y}, \boldsymbol{g})$-space). The d column refers to uniform exponent $\boldsymbol{d} = (d, \ldots, d)$ of our radical transform. The last column $\sigma(E_1^{\text{std}}, E_1^{\text{xform}})$ is the most significant, showing the relative improvement of our method over E_1^{std} (32).

Table 7 further investigates the impact of the step size h on the improvement ratio. In this experiment, the initial box E_0 is fixed, while h is gradually increased (from 0.00001 to 0.6), and the corresponding changes in σ are observed.

From the experimental results, we can conclude the following:
1. Applying the transformation consistently yields a tighter end-enclosure. Moreover, this improvement appears to grow exponentially.
2. When the IVP system exhibits significantly faster growth in one coordinate direction over a certain step size range, the benefit of applying the transformation becomes increasingly pronounced as the step size grows. This is clearly observed in examples such as eg2, eg3, and eg4. The case of eg1 with a loop trajectory (see Figure 1), when the step size is small (e.g., $h = 0.00001$), the system is in the positive zone region, and the transformation has a slight noticeable effect. However, for larger step sizes, the trajectory enters the negative zone, where the transformation loses its effectiveness.

Eg*	E_0	F_1	h	$\overline{\mu}^1$	d	$\overline{\mu}^2$	$\sigma(E_1^{\text{std}}, E_1^{\text{xform}})$
Eg1-a	$Box_{(1,3)}(10^{-4})$	$(0.95, 2.95) \pm (0.05, 0.05)$	0.00001	0.07	17 1	-68.30 -5.82	1.0000 1.0006
Eg1-b	$Box_{(1,3)}(10^{-4})$	$(0.95, 2.95) \pm (0.05, 0.05)$	0.0028	0.07	17 1	-68.30 -5.82	1.0000 1.0000
Eg2-a	$Box_{(3,-3)}(10^{-4})$	$(2.95, -2.95) \pm (0.05, 0.05)$	0.00086	5.90	23 1	-140.80 -11.00	1.0002 1.0007
Eg2-b	$Box_{(3,-3)}(10^{-4})$	$(2.95, -2.85) \pm (0.05, 0.15)$	0.01	5.93	23 1	-370.14 -9.20	1.0321 1.0458
Eg3-a	$Box_{(3,-3)}(10^{-4})$	$(2.95, -2.95) \pm (0.05, 0.05)$	0.001	9.75	20 1	-177.05 -9.87	1.0000 1.0008
Eg3-b	$Box_{(3,-3)}(10^{-4})$	$(3.05, -2.80) \pm (0.15, 0.20)$	0.02	10.64	20 1	-163.12 -9.39	1.0035 1.0665
Eg4-a	$Box_{(1.0,3.0,1.0)}(10^{-4})$	$(0.95, 2.95, 0.95) \pm (0.05, 0.05, 0.05)$	0.001	13.60	58 1	-22.10 -2.97	1.0102 1.0186
Eg4-b	$Box_{(1.0,3.0,1.0)}(10^{-4})$	$(1.20, 3.30, 0.95) \pm (0.30, 0.40, 0.05)$	0.02	13.62	10 1	-6.01 -3.01	1.3286 1.3933

Table 6
Comparison of our transform method with E_1^{std} (32). The value δ is fixed at 10^{-7} throughout.

	E_0	0.00001	0.0001	0.001	0.01	0.1	0.2	0.4	0.6
Eg1	$Box_{(1,3)}(10^{-4})$	1.0006	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Eg2	$Box_{(3,-3)}(10^{-4})$	1.0001	1.0008	1.005	1.055	2.628	6.227	32.772	192.823
Eg3	$Box_{(3,-3)}(10^{-4})$	1.0005	1.0013	1.003	1.032	1.706	2.517	7.923	17.892
Eg4	$Box_{(1,3,1)}(10^{-4})$	1.0006	1.0015	1.016	1.156	1.737	3.022	9.027	27.283

Table 7
Comparison of our transform method with E_1^{std} (32). Under Increasing Step Sizes.

6. `Extend` and `Refine` Subroutines

We now present algorithms for `Extend` and `Refine` as specified in equations (19)–(20).

For the i-th stage with admissible triple $(E_i, \Delta t_i, F_i)$, we compute transformation parameters π_i and $\boldsymbol{g}_i$. It turns out that to compute π_i and $\boldsymbol{g}_i$, it is necessary[11] using symbolic methods. Since this computation is expensive, we do not refine stages by splitting a stage into two or more stages. Instead, we use a "light-weight" approach encoded in the refinement structure G_i that does not recompute π_i and $\boldsymbol{g}_i$. Specifically, the time interval Δt_i is uniformly subdivided into 2^{ℓ_i} **mini-steps** where ℓ_i is the **level**. For each mini-step, we compute the full enclosure $\boldsymbol{F}_i$, end enclosure $\boldsymbol{E}_i$, and their associated logarithmic norms (logNorm) $\overline{\boldsymbol{\mu}}_1$ (in the original $\boldsymbol{x}$-space) and $\overline{\boldsymbol{\mu}}_2$ (in transformed $\boldsymbol{y}$-space). Here are the details of G_i:

$$G_i := G_i(\mathcal{S}) = (\pi_i, \boldsymbol{g}_i; \overline{\boldsymbol{\mu}}_i^1, \overline{\boldsymbol{\mu}}_i^2, \delta_i, h_i^{\text{euler}}, (\ell_i, \boldsymbol{E}_i, \boldsymbol{F}_i)) \tag{35}$$

where $\overline{\boldsymbol{\mu}}_{\mathbf{i}}^{\mathbf{1}}, \overline{\boldsymbol{\mu}}_{\mathbf{i}}^{\mathbf{2}}, \boldsymbol{E}_{\mathbf{i}}, \boldsymbol{F}_{\mathbf{i}}$ are arrays of length 2^{ℓ_i} and the parameters in red are extra data needed by the `Refine` subroutine below. We call $(\ell_i, \boldsymbol{E}_i, \boldsymbol{F}_i)$ the **mini-scaffold**.

6.1. `Extend` Subroutine

We now give the details of $\mathcal{S}$.`Extend`$(\cdots)$ introduced in the overview:

$\mathcal{S}$.`Extend`(ε_0, H)
INPUT: m-stage scaffold $\mathcal{S}$, $\varepsilon > 0$, $H > 0$.
OUTPUT: $\mathcal{S}'$ is a $m+1$-stage extension $\mathcal{S}$ such that
$\Delta t_{m+1}(\mathcal{S}') \leq H$, and $(\Delta t_{m+1}(\mathcal{S}'), E_{m+1}(\mathcal{S}'))$ is an ε-admissible pair for $E_m(\mathcal{S})$.

$(\widehat{h}, F_1) \leftarrow$ `StepA`$(E_m(\mathcal{S}), \varepsilon_0, H)$.
$\overline{\mu}_1 \leftarrow \mu_2(J_f(F_1))$.
$E_1 \leftarrow$ `StepB`$(E_m(\mathcal{S}), \widehat{h}, F_1, \overline{\mu}_1)$.
$(\overline{\mu}_2, \pi, \boldsymbol{g}) \leftarrow$ `Transform`$(\boldsymbol{f}, F_1, \overline{\mu}_1)$.
$\delta_1' \leftarrow$ `TransformBound`$(\varepsilon_0, \pi, F_1)$
$h_1 \leftarrow h^{\text{euler}}(\widehat{h}, \|\boldsymbol{g}^{[2]}(\pi(F_1))\|, \overline{\mu}_2, \delta_1')$. ◁ See (11).
Let $\mathcal{S}_{m+1} \leftarrow (t_m + h_1, E_1, F_1)$ and $G_{m+1} \leftarrow (\pi, \boldsymbol{g}; (\overline{\mu}_1), (\overline{\mu}_2), \varepsilon_0, h_1, (0, (E_m(\mathcal{S}), E_1), (F_1)))$.
Return $\mathcal{S}; (\mathcal{S}_{m+1}, G_{m+1})$.

[11] A numerical approach via automatic differentiation is in principle possible, but it gives extremely poor bounds. We need simplification of the expressions, which is symbolic.

6.2. `Refine` Subroutine

The goal of $\mathcal{S}$.`Refine`(ε_0) is to ensure that the end-enclosure of $\mathcal{S}$ has max-width $\le \varepsilon_0$. Each iteration of the main loop of `Refine` is called a **phase**. If $\mathcal{S}$ has m stages, then in each phase, we process stages $i = 1, \dots, m$ in this order. Recall the[12] mini-scaffold $(\ell_i, \boldsymbol{F}_i, \boldsymbol{E}_i)$ of stage i above. This mini-scaffold has a uniform time step of size $(\Delta t_i) \cdot 2^{-\ell_i}$. See Figure 4 for illustration. We will call the following $\mathcal{S}$.`Bisect`(i) to perform a bisection of each mini-step:

$\mathcal{S}$.`Bisect`(i)
INPUT: ith refers to a stage of $\mathcal{S}$.
Let $(\pi_i, g_i, \overline{\boldsymbol{\mu}}_i^1, \overline{\boldsymbol{\mu}}_i^2, \delta_i, \widehat{h}_i, \ell_i, \boldsymbol{E}_i, \boldsymbol{F}_i) \leftarrow G_i(\mathcal{S})$
be the ith refinement structure
OUTPUT: each mini-step of the ith stage
is halved and $\overline{\boldsymbol{\mu}}_i^1, \boldsymbol{E}_i, \boldsymbol{F}_i$ are updated.

Initialize three new vectors $\boldsymbol{\mu} = []$, $\boldsymbol{E}' = [\boldsymbol{E}_i[0]]$ and $\boldsymbol{F}' = []$.
$h \leftarrow (\Delta t_i) 2^{-\ell_i - 1}$
For $j = 1, \dots, 2^{\ell_i}$,
▷ *First half of j step*
$tmpF_1 \leftarrow \sum_{i=0}^{k-1} \left([0,h]^i \boldsymbol{f}^{[i]}(\boldsymbol{E}'.back()) + [0,h]^k \boldsymbol{f}^{[k]}(\boldsymbol{F}_i[j])\right)$.
$\mu' \leftarrow \mu_2(\boldsymbol{J}_{\boldsymbol{f}}(tmpF_1))$; $\boldsymbol{\mu}$.push_back(μ').
$E \leftarrow$ `StepB`$(\boldsymbol{E}'.back(), h, tmpF_1, \mu')$.
$\boldsymbol{E}'$.push_back(E); $\boldsymbol{F}'$.push_back($tmpF_1$);
▷ *Second half of j step*
$tmpF_2 \leftarrow \sum_{i=0}^{k-1} \left([0,h]^i \boldsymbol{f}^{[i]}(E) + [0,h]^k \boldsymbol{f}^{[k]}(\boldsymbol{F}_i[j])\right)$.
$\mu' \leftarrow \mu_2(\boldsymbol{J}_{\boldsymbol{f}}(tmpF_2))$; $\boldsymbol{\mu}$.push_back(μ').
$E \leftarrow$ `StepB`$(E, h, tmpF_2, \mu')$
$\boldsymbol{E}'$.push_back(E); $\boldsymbol{F}'$.push_back($tmpF_2$);
$(\overline{\boldsymbol{\mu}}_i^1, \ell_i, \boldsymbol{E}_i, \boldsymbol{F}_i) \leftarrow (\boldsymbol{\mu}, \ell_i + 1, \boldsymbol{E}', \boldsymbol{F}')$

In the above code, we view $\boldsymbol{E}_i$ and $\boldsymbol{F}_i$ as a vector in the sense of `C++`. We append an item $\boldsymbol{E}$ to the end of the vector by calling $\boldsymbol{E}_i$.push_back(E) and $\boldsymbol{E}_i$.back() returns the last item. When $(\Delta t_i)2^{-\ell_i}$ is less than the bound in (11), we can update the data $E_i, F_i, \overline{\boldsymbol{\mu}}^1, \overline{\boldsymbol{\mu}}^2$ using the `EulerTube` subroutine as described next:

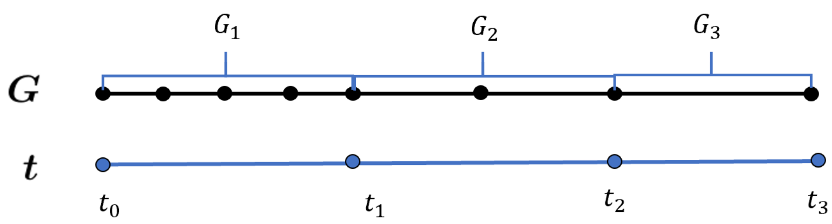

Figure 4: 3-stage scaffold $\mathcal{S}$ with $\ell_1 = 2$, $\ell_2 = 1$ and $\ell_3 = 0$ in G.

[12]Viewing the ith stage as a `bigStep`, the mini-scaffold represent `smallSteps`s of the ith stage.

> $\mathcal{S}$.EulerTube(i)
> INPUT: i refers to the ith stage of $\mathcal{S}$.
> OUTPUT: refine the ith stage using Lemma 7,
> such that the ith stage is $\delta(G_i(\mathcal{S}))$-bounded
> (Note: $E_i(\mathcal{S}), F_i(\mathcal{S}), G_i(\mathcal{S})$ are modified)
>
> ---
>
> Let $(\pi, \boldsymbol{g}, \overline{\boldsymbol{\mu}}^1, \overline{\boldsymbol{\mu}}^2, \delta, \widehat{h}, \ell, \boldsymbol{E}, \boldsymbol{F})$ be $G_i(\mathcal{S})$
> Let $Ball_p(r_0)$ be the circumscribing ball of $\boldsymbol{E}[0]$
> and $Ball_{p'}(r_0')$ be the circumscribing ball of $\pi(\boldsymbol{E}[0])$.
> $\boldsymbol{q} \leftarrow \pi(\boldsymbol{p}),\ d \leftarrow \|\boldsymbol{q} - m(Box(\pi(\boldsymbol{E}[0])))\|$.
> Let H be the step size each mini-step of $\mathcal{S}[i]$.
> For $(j = 1, \dots, 2^\ell)$
> $\quad \boldsymbol{q} \leftarrow \boldsymbol{q} + \boldsymbol{g}(\boldsymbol{q})H,$
> $\quad \delta_1 \leftarrow$ TransformBound$(\delta, \pi, \boldsymbol{F}[j])$.
> $\quad \overline{\boldsymbol{\mu}}^2[j] \leftarrow \mu_2(J_{\boldsymbol{g}}(\pi(\boldsymbol{F}[j])))$.
> $\quad r_1 \leftarrow r_0 e^{j\overline{\boldsymbol{\mu}}^1[j]H} + \delta;\ r_1' \leftarrow (r_0' + d)e^{j\overline{\boldsymbol{\mu}}^2[j]H} + \delta_1$
> $\quad B \leftarrow Box(r_1);\ B' \leftarrow Box(r_1')$.
> $\quad \boldsymbol{F}[j] \leftarrow \boldsymbol{F}[j] \cap \pi^{-1}(Box(center(\pi(\boldsymbol{E}[j-1])) + B', \boldsymbol{q} + B')$ ◁ *Full-enclosure for mini-step*
> $\quad \overline{\boldsymbol{\mu}}^1[j] \leftarrow \mu_2(J_{\boldsymbol{f}}(\boldsymbol{F}[j]))$.
> $\quad \boldsymbol{E}[j] \leftarrow \boldsymbol{E}[j] \cap \pi^{-1}(\boldsymbol{q} + B') \cap (\pi^{-1}(\boldsymbol{q}) + B)$. ◁ *End-enclosure for mini-step*

Observe that EulerTube performs all its Euler computation in transformed space, and only pulls back the enclosures back to primal space. It turns out that EulerTube is extremely efficient compared to Bisect, and moreover, it ensures that the ith stage is now δ_i-bounded.

We are ready to describe the Refine subroutine:

> $\mathcal{S}$.Refine(ε_0)
> INPUT: m-stage augmented scaffold $\mathcal{S}$ and $\varepsilon_0 > 0$.
> OUTPUT: $\mathcal{S}$ remains an m-stage augmented scaffold
> but the length satisfies $w_{\max}(E_m) \le \varepsilon_0$.
>
> ---
>
> $r_0 \leftarrow w_{\max}(E_m(\mathcal{S}))$.
> While $(r_0 > \varepsilon_0)$
> $\quad$ For $(i = 1, \dots, m)$ ◁ *Begin new phase*
> $\quad\quad \ell := G_i(\mathcal{S}).\ell;\ \widehat{h} := G_i(\mathcal{S}).\widehat{h}$.
> $\quad\quad H \leftarrow (\Delta t_i)2^{-\ell}$
> $\quad\quad$ If $(H > \widehat{h})$
> $\quad\quad\quad \mathcal{S}$.Bisect($i$)
> $\quad\quad$ Else
> $\quad\quad\quad \mathcal{S}$.EulerTube($i$)
> $\quad\quad\quad G_i(\mathcal{S}).\delta \leftarrow \delta/2$
> $\quad\quad$ ▷ *Update $\widehat{h}$ in $G_i(\mathcal{S})$:*
> $\quad\quad\quad$ I.e., let $(\pi, \boldsymbol{g}, \overline{\boldsymbol{\mu}}^1, \overline{\boldsymbol{\mu}}^2, \delta, \widehat{h}, \ell, \boldsymbol{E}, \boldsymbol{F})$ be $G_i(\mathcal{S})$
> $\quad\quad\quad\quad E_i(\mathcal{S}) \leftarrow \boldsymbol{E}[2^\ell]$ ◁ *End-enclosure for stage*
> $\quad\quad\quad\quad F_i(\mathcal{S}) \leftarrow \bigcup_{j=1}^{2^\ell} \boldsymbol{F}[j]$ ◁ *Full-enclosure for stage*
> $\quad\quad\quad\quad \mu_2 \leftarrow \max\left\{\overline{\boldsymbol{\mu}}^2[j] : j = 1, \dots, 2^\ell\right\}$
> $\quad\quad\quad\quad \delta_1' \leftarrow$ TransformBound$(\delta, \pi, F_i(\mathcal{S}))$
> $\quad\quad\quad\quad \overline{M} \leftarrow \|\boldsymbol{g}^{[2]}(\pi(F_i(\mathcal{S}))\|_2$
> $\quad\quad\quad\quad htmp \leftarrow \min\left\{\Delta t_i, h^{\text{euler}}(\widehat{h}, \overline{M}, \mu_2, \delta_1')\right\}$ ◁ *(see (11))*
> $\quad\quad G_i(\mathcal{S}).\widehat{h} \leftarrow htmp$
> $\quad$ ◁ *End of For-Loop*
> $\quad E_0(\mathcal{S}) \leftarrow \frac{1}{2}E_0(\mathcal{S})$
> $\quad r_0 \leftarrow w_{\max}(E_m(\mathcal{S}))$
> Return $\mathcal{S}$

THEOREM 11. *The subroutine $\mathcal{S}$.Refine(ε_0) is correct. In particular, it halts.*

This proof is a bit subtle, and can be illustrated by the following **phase-stage** diagram:

$$
\begin{array}{c|ccccc}
 & \text{Stage } 1 & \cdots & \text{Stage } i & \cdots & \text{Stage } m \\
\hline
\text{Phase 1:} & \delta_1^1 & \cdots & \delta_i^1 & \cdots & \varepsilon_0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
\text{Phase } k\text{:} & \delta_1^k & \cdots & \delta_i^k & \cdots & \delta_m^k \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots
\end{array}
\tag{36}
$$

Each phase will refine the stages $1, 2, \dots, m$ (in this order). The ith stage in phase k has a "target bound" $\delta_i^k > 0$ stored as $G_i(\mathcal{S}).\delta$. For $k = 1$, δ_i^1 is ε_0 if $i = m$ and otherwise inherited from $(m-1)$-stage structure before Extend$(\varepsilon_0, \boldsymbol{H})$. For $k > 1$, δ_i^k halved after a call to EulerTube subroutine (see Refine(ε)). The proof in the appendix will show that $\lim_{k\to\infty} \delta_i^k = 0$ for each $i = 1, \dots, m$, which will contradict the non-halting of Refine.

7. The Main Algorithm and Experiments

The following is our algorithm to solve the `EndEncl_IVP` problem of (2):

> $\texttt{EndEnc}_f(B_0, \varepsilon_0) \to (\underline{B}_0, \overline{B}_1)$:
> INPUT: $\varepsilon_0 > 0$, $B_0 \in \square\mathbb{R}^n$
> such that $\texttt{IVP}_f(B_0, 1)$ is valid
> OUTPUT: $\underline{B}_0, \overline{B}_1 \in \square\mathbb{R}^n$, $\underline{B}_0 \subseteq B_0$, $w_{\max}(\overline{B}_1) < \epsilon$
> and $\overline{B}_1$ is an end-enclosure of $\mathbf{IVP}(\underline{B}_0, 1)$
>
> ---
>
> ▷ *Initialize a 0-stage scaffold $\mathcal{S}$:*
> $\mathcal{S} \leftarrow ((t_0), (B_0), (B_0), (Id, \boldsymbol{f}, (\mu_1), (\mu_2), \varepsilon_0, \widehat{h}, (\ell, (B_0, B_0), (B_0))))$
> where $(t_0, \mu_1, \mu_2, \widehat{h}, \ell) \leftarrow (0, 0, 0, 0, 0)$
> $t \leftarrow 0$
> While $t < 1$
> $\mathcal{S}.\texttt{Extend}(\varepsilon_0, 1 - t)$
> $\mathcal{S}.\texttt{Refine}(\varepsilon_0)$
> $t \leftarrow \boldsymbol{t}(\mathcal{S})$.back()
> Return $(E_0(\mathcal{S}), \boldsymbol{E}(\mathcal{S})$.back())

THEOREM 12. *Algorithm* $\texttt{EndEnc}(B_0, \varepsilon_0)$ *halts, provided the interval computation of* `StepA` *is isotonic. The output is also correct.*

7.1. Implementation and Limitations

We implemented `EndEnc` in `C++`. Our implementation follows the explicitly described subroutines in this paper. There are no hidden hyperparameters (e.g., our step sizes are automatically adjusted). Our eventual code will be open-sourced in [41].

Implementation of the various numerical formulas such as Taylor forms implicitly call interval methods as explained in Subsection 2.2. The radical transform requires symbolic computation (Section 6) which we take from the `symEngine` library (https://symengine.org/).

Limitations. The main caveat is that we use machine arithmetic (IEEE standard). There are two main reasons. First, this is necessary to have fair comparisons to existing software and we rely on library routines based on machine arithmetic. In principle, we can implement our algorithm using arbitrary precision number types (which will automatically get a hit in performance regardless of needed precision).

7.2. Global Experiments

In previous sections, we had tables of experimental results evaluating "local" (1-step) operations. In this section, we show three tables (A, B, C) that solve complete IVP problems over time $t \in [0, 1]$ (with one exception in Table C). Table A compares our `StepA`/`StepB` with the standard methods. Table B is an internal evaluation of our transform and `Bisect`/`EulerTube` heuristic. Table C is an external comparison of our main algorithm with other validated software. The problems are from Table 1. We informally verify our outputs by tracing points using `CAPD`'s code to see that their outputs are within our end-enclosures. Timings are taken on a laptop with a 13th Gen Intel Core i7-13700HX 2.10 GHz processor and 16.0 GB RAM.

Our tables indicate two kinds of error conditions: Timeout and No Output. The former means the code took more than 1 hour to run. the latter means the code stopped with no output.

In each table, we are comparing our main enclosure algorithm denoted `Ours` with some algorithm X where X may be variants of `Ours` or other IVP software. We define **speedup over** X as $\sigma(X) := \frac{\text{Time}(X)}{\text{Time(Ours)}}$.

TABLE A: We compare the relative computation times of `Ours` against $\texttt{Ours/StepA}_0$ and $\texttt{Ours/StepB}_0$. Here $\texttt{Ours/StepA}_0$ denotes the algorithm where we replace `StepA` by $\texttt{StepA}_0$ in `Ours`. Similarly for $\texttt{Ours/StepB}_0$. Recall $\texttt{StepA}_0$ is the non-adaptive stepA in Subsection 4.1, and $\texttt{StepB}_0$ is the direct method of (24). The speedup $\sigma(\texttt{Ours/StepB}_0)$ is a good measure of relative performance of `StepB` and $\texttt{StepB}_0$ when `Ours` and $\texttt{Ours/StepB}_0$ have about the same number of phases. So we include the statistic $\rho(\texttt{Ours/StepB}_0) := \frac{\text{phases(Ours/stepB}_0)}{\text{phases}(Ours)}$.

Example	E_0	ε	Time(Ours)	σ(Ours/StepA$_0$)	σ(Ours/StepB$_0$)	ρ(Ours/StepB$_0$)
Eg1	$Box_{(1,3)}(0.1)$	0.1	0.018	5.44	1.00	4/4
		0.01	0.073	1.57	1.08	5/5
		0.001	0.643	2.465	1.75	8/7
		0.0001	12.803	1.49	1.03	9/9
Eg2	$Box_{(-3,3)}(0.1)$	0.1	0.031	653.22	1.00	4/4
		0.01	0.086	334.875	1.025	7/7
		0.001	1.437	> 1000	1.074	10/10
		0.0001	11.491	> 1000	1.102	13/13
Eg3	$Box_{(-1.5,8.5)}(0.01)$	10.0	0.043	> 1000	1.03	1/1
		5.0	0.027	> 1000	1.04	1/1
		1.0	0.022	> 1000	1.06	1/1
		0.1	2.159	> 1000	5.81	3/3
Eg4	$Box_{(15,15,36)}(0.001)$	10.0	0.142	> 1000	1.24	1/1
		5.0	0.130	> 1000	2.24	5/1
		1.0	0.122	> 1000	14.38	7/1
		0.1	0.205	> 1000	> 1000	Timeout

Table A: Comparison of StepA and StepB with StepA$_0$ and direct-method. all run with $order = 20$.

Eg	ε	B_0	Method	$\underline{B}_0$	$\overline{B}_1$	#(miniSteps)	Time(s)
$x' = x^2$	0.01	$[0.8, 0.9]$	OurSimple	0.8495 ± 0.0005	5.6665 ± 0.0045	20861	45.630
			OurSimpleT	0.8495 ± 0.0005	5.6665 ± 0.0045	18	11.854
			OurNoT	0.8495 ± 0.0005	5.6665 ± 0.0045	297	15.073
			OurNoEuler	0.8495 ± 0.0005	5.6665 ± 0.0045	31	71.846
			Ours	0.8495 ± 0.0005	5.6665 ± 0.0045	16	5.966
$x' = x^2$	0.001	$[0.98, 0.99]$	OurSimple	0.985 ± 0.0005	65.66665 ± 0.00035	128890	2104.03
			OurSimpleT	0.985 ± 0.0005	65.66665 ± 0.00035	103	31.898
			OurNoT	0.985 ± 0.0005	65.66665 ± 0.00035	22763	85.051
			OurNoEuler	0.985 ± 0.0005	65.66665 ± 0.00035	72489	327.89
			Ours	0.985 ± 0.0005	65.66665 ± 0.00035	105	49.861
Eg1	0.01	$Box_{(1,3)}(0.1)$	OurSimple	$(0.995, 2.995) \pm (0.005, 0.005)$	$(0.077, 1.4635) \pm (0.001, 0.0035)$	1454	13.877
			OurSimpleT	$(0.995, 2.995) \pm (0.005, 0.005)$	$(0.077, 1.4635) \pm (0.001, 0.0035)$	1888	15.370
			OurNoT	$(0.995, 2.995) \pm (0.005, 0.005)$	$(0.077, 1.4635) \pm (0.001, 0.0035)$	47	9.386
			OurNoEuler	$(0.995, 2.995) \pm (0.005, 0.005)$	$(0.077, 1.4635) \pm (0.001, 0.0035)$	47	9.415
			Ours	$(0.995, 2.995) \pm (0.005, 0.005)$	$(0.077, 1.4635) \pm (0.001, 0.0035)$	47	9.232
Eg2	0.1	$Box_{(-3,3)}(0.1)$	OurSimple	$(-2.995, 3.0) \pm (0.025, 0.025)$	$(-2.13, 0.56) \pm (0.05, 0.02)$	997	15.343
			OurSimpleT	$(-2.995, 3.0) \pm (0.025, 0.025)$	$(-2.13, 0.56) \pm (0.05, 0.02)$	1367	16.540
			OurNoT	$(-2.995, 3.0) \pm (0.025, 0.025)$	$(-2.13, 0.56) \pm (0.05, 0.02)$	14	14.785
			OurNoEuler	$(-2.995, 3.0) \pm (0.025, 0.025)$	$(-2.13, 0.56) \pm (0.05, 0.02)$	14	15.119
			Ours	$(-2.995, 3.0) \pm (0.025, 0.025)$	$(-2.13, 0.56) \pm (0.05, 0.02)$	14	14.590
Eg3	0.1	$Box_{(-1.5,8.5)}(0.01)$	OurSimple	$(-1.495, 8.495) \pm (0.005, 0.005)$	$(-0.595, -6.685) \pm (0.005, 0.045)$	2908	25.710
			OurSimpleT	$(-1.495, 8.495) \pm (0.005, 0.005)$	$(-0.595, -6.685) \pm (0.005, 0.045)$	3223	23.110
			OurNoT	$(-1.495, 8.495) \pm (0.005, 0.005)$	$(-0.595, -6.685) \pm (0.005, 0.045)$	35	23.773
			OurNoEuler	$(-1.495, 8.495) \pm (0.005, 0.005)$	$(-0.595, -6.685) \pm (0.005, 0.045)$	1034	455.548
			Ours	$(-1.495, 8.495) \pm (0.005, 0.005)$	$(-0.595, -6.685) \pm (0.005, 0.045)$	25	11.795
Eg4	5	$Box_{(15,15,36)}(0.001)$	OurSimple	$(15, 15, 36) \pm (0.0005, 0.0005, 0.0005)$	$(-6.94, 1.81, 35.52) \pm (1.64, 2.33, 2.5)$	26	294.019
			OurSimpleT	$(15, 15, 36) \pm (0.0005, 0.0005, 0.0005)$	$(-6.94, 1.81, 35.52) \pm (1.64, 2.33, 2.5)$	26	166.969
			OurNoT	$(15, 15, 36) \pm (0.0005, 0.0005, 0.0005)$	$(-6.945, 2.99, 35.14) \pm (1.005, 2.15, 1.88)$	61	146.360
			OurNoEuler	$(15, 15, 36) \pm (0.0005, 0.0005, 0.0005)$	$(-6.94, 1.81, 35.52) \pm (1.64, 2.33, 2.5)$	26	159.028
			Ours	$(15, 15, 36) \pm (0.0005, 0.0005, 0.0005)$	$(-6.945, 2.99, 35.14) \pm (1.005, 2.15, 1.88)$	61	123.64

Table B: The global effects of radical transform and bisection for our algorithm, all run with $order = 3$.

We conclude from Table A that our StepA significantly improves efficiency (σ(Ours/StepA$_0$)). Moreover, StepB also yields a noticeable performance gain and effectively reduces the number of required phases (σ(Ours/StepB$_0$) and ρ(Ours/StepB$_0$)).

TABLE B: We conduct experiments to show the benefits of various techniques used in our algorithm. The algorithms X being compared in Table B differ from Ours only in the use of variants of the Refine subroutine. Specifically: X = OurSimple uses E_1^{std} (32) to compute the end-enclosure, without performing our Bisect subroutine. X = OurSimpleT is similar, except that we use E_1^{xform} (33) instead. Similarly, X = OurNoT represents a variant of our algorithm with the transform step disabled (i.e., the transformation π is set to the identity map). Finally, X = OurNoEuler is our algorithm with EulerTube disabled.

We run all the experiments with order $k = 3$ because with high order (e.g. $k = 20$), the number of stages is too small (see Table C).

We conclude from Table B that the transform method improves efficiency overall (compare OurSimple vs. OurSimpleT, and Ours vs. OurNoT). In certain cases, such as the example $x' = x^2$ and Eg3, the transform method can significantly improve performance. For the two-dimensional examples, Eg1, Eg2 and Eg3, the comparison between OurSimpleT and Ours shows that our Bisect subroutine enhances the overall performance of the algorithm. Also, comparing OurNoEuler with Ours shows that EulerTube can significantly improve performance when there are many mini-steps (e.g., $x' = x^2$ and Eg3).

TABLE C: The time span is $t \in [0, 1]$ in all the experiments except for Eg1-b, where $t \in [0, 5.5]$ corresponding to one full loop. This is an example that AWA cannot solve [13, p. 13]. For each example of order $k = 20$, we use our algorithm to compute a scaffold $\mathcal{S}(\varepsilon_0)$ for an initial value of ε_0; subsequently, this scaffold is refined using a smaller ε_i ($i = 1, 2, \ldots$) to obtain $\mathcal{S}(\varepsilon_1)$, $\mathcal{S}(\varepsilon_2)$, …. The total number of mini-steps in all the stages of $\mathcal{S}(\varepsilon_i)$ is shown in column

Case	Method	ε	$\underline{B}_0$	$\overline{B}_1$	#(miniSteps)	Time(s)	$\sigma(X)$
Eg1-a	Ours	1.0	$Box_{(1.0,3.0)}(0.1)$	$(0.08, 1.46) \pm (0.06, 0.16)$	7	0.010	1
	Refine	0.05	$Box_{(1.0,3.0)}(0.05)$	$(0.08, 1.46) \pm (0.02, 0.05)$	13	0.009	N/A
	Refine	0.03	$Box_{(1.0,3.0)}(0.025)$	$(0.08, 1.46) \pm (0.006, 0.02)$	25	0.017	N/A
	$\mathtt{StepB}_{\mathtt{Direct}}$	N/A	N/A	$(0.08, 1.47) \pm (0.06, 0.15)$	N/A	0.014	1.4
	$\mathtt{StepB}_{\mathtt{Lohner}}$	N/A	N/A	$(0.08, 1.46) \pm (0.06, 0.15)$	N/A	0.031	3.1
	CAPD	N/A	N/A	$(0.08, 1.46) \pm (0.03, 0.10)$	N/A	0.018	1.8
Eg1-b	Ours	3.3	$Box_{(1.0,3.0)}(0.0125)$	$(0.95, 3.00) \pm (0.20, 0.30)$	294	0.404	1
	Refine	0.15	$Box_{(1.0,3.0)}(0.00625)$	$(0.95, 3.00) \pm (0.10, 0.14)$	587	0.326	N/A
	Refine	0.07	$Box_{(1.0,3.0)}(0.00313)$	$(0.95, 3.00) \pm (0.05, 0.7)$	1173	0.638	N/A
	$\mathtt{StepB}_{\mathtt{Direct}}$	N/A	N/A	Timeout	N/A	Timeout	-
	$\mathtt{StepB}_{\mathtt{Lohner}}$	N/A	N/A	Timeout	N/A	Timeout	-
	CAPD	N/A	N/A	No Output	N/A	No Output	-
Eg2	Ours	1.0	$Box_{(-3.1,3.1)}(0.1)$	$(-2.14, 0.57) \pm (0.28, 0.28)$	10	0.016	1
	Refine	0.1	$Box_{(-3.1,3.1)}(0.05)$	$(-2.14, 0.57) \pm (0.08, 0.04)$	19	0.012	N/A
	Refine	0.05	$Box_{(-3.1,3.1)}(0.025)$	$(-2.14, 0.57) \pm (0.03, 0.01)$	37	0.023	N/A
	$\mathtt{StepB}_{\mathtt{Direct}}$	N/A	N/A	$(-2.14, 0.57) \pm (0.26, 0.23)$	N/A	0.506	31.6
	$\mathtt{StepB}_{\mathtt{Lohner}}$	N/A	N/A	$(-2.14, 0.57) \pm (0.26, 0.23)$	N/A	0.904	56.5
	CAPD	N/A	N/A	$(-2.14, 0.57) \pm (0.29, 0.29)$	N/A	0.012	0.75
Eg3	Ours	1.0	$Box_{(-1.51,8.51)}(0.01)$	$(-0.6, -6.69) \pm (0.00, 0.19)$	10	0.012	1
	Refine	0.06	$Box_{(-1.51,8.51)}(0.005)$	$(-0.6, -6.69) \pm (0.0008, 0.06)$	19	0.012	N/A
	Refine	0.03	$Box_{(-1.51,8.51)}(0.0025)$	$(-0.6, -6.69) \pm (0.0004, 0.02)$	37	0.022	N/A
	$\mathtt{StepB}_{\mathtt{Direct}}$	N/A	N/A	$(-0.60, -6.69) \pm (0.01, 0.19)$	N/A	4.113	342.7
	$\mathtt{StepB}_{\mathtt{Lohner}}$	N/A	N/A	$(-0.60, -6.69) \pm (0.01, 0.19)$	N/A	6.044	503.6
	CAPD	N/A	N/A	$(-0.60, -6.69) \pm (0.01, 0.19)$	N/A	0.017	1.4
Eg4	Ours	4.5	$Box_{(15.0,15.0,36.0)}(0.001)$	$(-6.94, 2.99, 35.14) \pm (0.09, 0.15, 0.15)$	23	0.053	1
	Refine	0.6	$Box_{(15.0,15.0,36.0)}(0.0003)$	$(-6.94, 2.99, 35.14) \pm (0.05, 0.06, 0.06)$	89	0.161	N/A
	Refine	0.03	$Box_{(15.0,15.0,36.0)}(0.0001)$	$(-6.94, 2.99, 35.14) \pm (0.02, 0.02, 0.02)$	177	0.203	N/A
	$\mathtt{StepB}_{\mathtt{Direct}}$	N/A	N/A	$(-6.95, 3.00, 35.14) \pm (31.56, 176.50, 173.98)$	N/A	3.830	72.2
	$\mathtt{StepB}_{\mathtt{Lohner}}$	N/A	N/A	$(-6.95, 3.00, 35.14) \pm (31.23, 169.99, 166.39)$	N/A	8.398	158.4
	CAPD	N/A	N/A	$(-6.94, 2.99, 35.14) \pm (0.03, 0.01, 0.04)$	N/A	0.088	1.66

Table C: Experiments on `EndEnc` and `Refine`: comparison to `CAPD`and `simple_IVP`, all executed with $order = 20$. $\sigma(X) := \frac{\text{Time}(X)}{\text{Time(Ours)}}$.

#(`miniSteps`); the timing for each refinement is incremental time. This nice refinement feature gives us to better precision control with low additional cost after the initial S.

We compared our algorithm with 3 other algorithms:

The first algorithm `CAPD` is from [42] and github. In Table 7.2, we invoke the method `ICnOdeSolver` with Taylor order 20, based on the C^r-Lohner algorithm [15, 43]. The method accepts an interval input such our B_0.

The other two algorithms are `simple_IVP` algorithm in (16), where `StepA` is $\mathtt{StepA}_0$ and `StepB` is either the $\mathtt{StepB}_{\mathtt{Direct}}$(see (24)) and well as $\mathtt{StepB}_{\mathtt{Lohner}}$. In Table 7.2, they are called $\mathtt{StepB}_{\mathtt{Direct}}$ and $\mathtt{StepB}_{\mathtt{Lohner}}$, respectively.

We conclude from Table C that our method outperforms `simple_IVP` in terms of efficiency and is nearly as efficient as `CAPD`. Note that we deliberately choose ε so that our final $\underline{B}_0$ is equal to the input B_0 in order to be comparable to the other methods. The only case where $\underline{B}_0 \neq B_0$ is $Eg1$-b: here, our method successfully computes a solution while all the other methods fail to produce any output. Since our current method does not directly address the wrapping effect, the resulting end-enclosure is less tight than that of `CAPD`, as seen in $Eg4$. In addition, when higher precision (smaller ε) is required, our `Refine` algorithm can efficiently compute solutions to meet the desired accuracy.

8. Conclusion

We have presented a complete validated IVP algorithm with the unique ability to pre-specify the an ε-bound on the width of the end-enclosure. Preliminary implementations show promise in comparison to current validated software. This paper introduces a more structured approach to IVP algorithms, opening the way for considerable future development of such algorithms. We introduced several novel techniques for Step A and Step B, including a new exploitation of logNorms combined with the radical transform.

For future work, we plan to do a full scale implementation that includes the ability to use arbitrary precision arithmetic, in the style of `Core Library` [41, 44]. We will also explore incorporating the Lohner-type transform into our radical transform.:w

Nedialkov et al. [8, Section 10], “*Some Directions for Future Research*”, presented a list of challenges that remain relevant today. Our algorithm is one answer to their call for automatic step sizes, order control (interpreted as error control) and tools for quasi-monotone problems (i.e., contractive systems).

A. Appendix A: Proofs

Note that the numberings of lemmas and theorems in this Appendix are the same as corresponding results in the text, and have hyperlinks to the text.

Corollary 4

Let $\boldsymbol{x}_1 \in \mathtt{IVP}(\boldsymbol{p}_1, h, Ball(\boldsymbol{p}_0, r))$ *and* $\boldsymbol{x}_2 \in \mathtt{IVP}(\boldsymbol{p}_2, h, Ball(\boldsymbol{p}_0, r))$.
If $\overline{\mu} \geq \mu_2(\frac{\partial f}{\partial \boldsymbol{x}}(Ball(\boldsymbol{p}_0, r)))$ *then for all* $t \in [0, h]$

$$\|\boldsymbol{x}_1(t) - \boldsymbol{x}_2(t)\|_2 \leq \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|_2 e^{\overline{\mu}t}.$$

Proof. Note that $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are solutions of (1) with different initial values. Therefore, we have $\boldsymbol{x}_1' = \boldsymbol{f}(\boldsymbol{x}_1)$ and $\boldsymbol{x}_2' = \boldsymbol{f}(\boldsymbol{x}_2)$. This implies that

$$\boldsymbol{x}_1'(t) - \boldsymbol{f}(\boldsymbol{x}_1(t)) = \boldsymbol{f}(\boldsymbol{x}_1(t)) - \boldsymbol{f}(\boldsymbol{x}_1(t)) = 0 = \varepsilon.$$

If $\overline{\mu} \neq 0$, then (10) is the first case of (9) since $\delta = \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|_2$. If $\overline{\mu} = 0$, it comes from the second case since $\varepsilon = 0$.
Q.E.D.

The following is a useful lemma:

Lemma A.1

Let (B_0, H, B_1) *be an admissible triple with* $\overline{\mu} \geq \mu_2(J_{\boldsymbol{f}}(B_1))$, *and* $\overline{M} \geq \|\boldsymbol{f}^{[2]}(B_1)\|$. *Denote the Euler step at* $\boldsymbol{q}_0 \in B_0$ *by the linear function*

$$\ell(t; \boldsymbol{q}_0) = \boldsymbol{q}_0 + t\boldsymbol{f}(\boldsymbol{q}_0).$$

Then for any $\boldsymbol{p}_0 \in B_0$ *and* $t \in [0, H]$,

$$\|\boldsymbol{x}(t; \boldsymbol{p}_0) - \ell(t; \boldsymbol{q}_0)\| \leq \|\boldsymbol{p}_0 - \boldsymbol{q}_0\| e^{\overline{\mu}t} + \frac{1}{2}\overline{M}t^2$$

Proof. By Corollary 4,

$$\|\boldsymbol{x}(t; \boldsymbol{p}_0) - \boldsymbol{x}(t; \boldsymbol{q}_0\| \leq \|\boldsymbol{p}_0 - \boldsymbol{q}_0\| e^{\overline{\mu}t} \tag{37}$$

We also have

$$\begin{aligned}
\boldsymbol{x}(t; \boldsymbol{q}_0) &= \boldsymbol{q}_0 + t \cdot \boldsymbol{f}(q_0) + \tfrac{1}{2}t^2 \boldsymbol{x}''(\tau) \quad \text{(for some } \tau \in [0, t]) \\
\|\boldsymbol{x}(t; \boldsymbol{q}_0) &- (\boldsymbol{q}_0 + t \cdot \boldsymbol{f}(q_0))\| \\
&\leq \|\tfrac{1}{2}t^2 \boldsymbol{x}''(\tau)\| \\
&= \|\tfrac{1}{2}t^2 \boldsymbol{f}^{[2]}(\boldsymbol{x}(\tau; \boldsymbol{q}_0)\| \\
&\leq \|\tfrac{1}{2}t^2 \overline{M}\| \quad \text{(since } \overline{M} \geq \boldsymbol{f}^{[x]}(B_1))
\end{aligned}$$

Combined with (37), the triangular inequality shows our desired bound. **Q.E.D.**

Lemma 5

Let (B_0, H, B_1) *be admissible triple,* $\overline{\mu} \geq \mu_2(J_{\boldsymbol{f}}(B_1))$ *and* $\overline{M} \geq \|\boldsymbol{f}^{[2]}(B_1)\|$. *For any* $\varepsilon > 0$, *if* $h_1 > 0$ *is given by*

$$h_1 \leftarrow h^{euler}(H, \overline{M}, \overline{\mu}, \varepsilon) := \begin{cases} \min\left\{H, \frac{2\overline{\mu}\varepsilon}{\overline{M}\cdot(e^{\overline{\mu}H}-1)}\right\} & \text{if } \overline{\mu} \geq 0 \\ \min\left\{H, \frac{2\overline{\mu}\varepsilon}{\overline{M}\cdot(e^{\overline{\mu}H}-1)-\overline{\mu}^2\varepsilon}\right\} & \text{if } \overline{\mu} < 0 \end{cases}$$

consider the path $Q_{h_1} = (\boldsymbol{q}_0, \boldsymbol{q}_1, \ldots, \boldsymbol{q}_m)$ from the Euler method with step-size h_1. If each $\boldsymbol{q}_i \in B_1$ $(i = 0, \ldots, m)$, then for all $t \in [0, H]$, we have

$$\|Q_{h_1}(t) - \boldsymbol{x}(t; \boldsymbol{q}_0)\| \le \varepsilon.$$

I.e., $Q_{h_1}(t)$ lies inside the ε-tube of $\boldsymbol{x}(t; \boldsymbol{q}_0)$.

Proof. For simplicity, we only prove the lemma when H/h_1 is an integer. We first show that the Euler method with step size $h_1 > 0$ has the following error bound:

$$\|\boldsymbol{q} - \boldsymbol{x}(H)\| \le \begin{cases} \frac{\overline{M} h_1}{2\overline{\mu}}(e^{\overline{\mu} H} - 1) & \overline{\mu} \ge 0, \\ \frac{\overline{M} h_1}{2\overline{\mu} + \overline{\mu}^2 h_1}(e^{\overline{\mu} H} - 1) & \overline{\mu} < 0. \end{cases} \tag{38}$$

To show (38), assume $(\boldsymbol{p}_0 = \boldsymbol{x}(0), \boldsymbol{p}_1, \ldots, \boldsymbol{p}_m = \boldsymbol{q})$ are obtained by the Euler method corresponding to $t_0 = 0, t_1, \ldots, t_m = H$. Let $g_i = \|\boldsymbol{p}_i - \boldsymbol{x}(t_i)\|_2$ be the error bound. Then we have

$$\begin{aligned} g_m &\le g_{m-1} e^{\overline{\mu} h_1} + \frac{\overline{M} h_1^2}{2} && \text{(by Taylor formula)} \\ &\le g_{m-2} e^{\overline{\mu} h_1} + \frac{\overline{M} h_1^2}{2} e^{\overline{\mu} h_1} + \frac{\overline{M} h_1^2}{2} && \text{(by expanding } g_{m-1}) \\ &\vdots \\ &\le \frac{\overline{M} h_1^2}{2}(1 + e^{\overline{\mu} h_1} + \cdots e^{\overline{\mu} h_1 (m-1)}) && \text{(since } g_0 = 0) \\ &\le \frac{\overline{M} h_1^2}{2} \frac{e^{\overline{\mu} H} - 1}{e^{\overline{\mu} h_1} - 1} \\ &\le \begin{cases} \frac{\overline{M} h_1}{2\overline{\mu}}(e^{\overline{\mu} H} - 1) & \text{if } \overline{\mu} \ge 0, \\ \frac{\overline{M} h_1}{2\overline{\mu} + \overline{\mu}^2 h_1}(e^{\overline{\mu} H} - 1) & \text{if } \overline{\mu} < 0. \end{cases} \end{aligned}$$

If $\overline{\mu} \ge 0$, then the last formula is justified by $e^{\overline{\mu} h_1} - 1 \ge \overline{\mu} h_1$, and so $g_m \le \frac{\overline{M} h_1}{2\overline{\mu}}(e^{\overline{\mu} H} - 1)$. If $\overline{\mu} < 0$, then the formula is justified by $e^{\overline{\mu} h_1} - 1 \le \overline{\mu} h_1 + \frac{1}{2}\overline{\mu}^2 h_1^2$ (use the fact that $f(x) = e^x - 1 - x - \frac{1}{2}x^2 < 0$ when $x < 0$; check that $f'(x) = e^x - 1 - x > 0$ for all $x < 0$). This proves (38). Note that

$$\begin{aligned} \overline{\mu} h_1 + \tfrac{1}{2}\overline{\mu}^2 h_1^2 &= \overline{\mu} h_1 (1 + \tfrac{1}{2}\overline{\mu} h_1) \\ &= \overline{\mu} h_1 \left(1 + \frac{\overline{\mu}^2 \varepsilon}{\overline{M} \cdot (e^{\overline{\mu} H} - 1) - \overline{\mu}^2 \varepsilon}\right) \\ & \quad \left(\text{Choose } h_1 = \frac{2\overline{\mu}\varepsilon}{\overline{M} \cdot (e^{\overline{\mu} H} - 1) - \overline{\mu}^2 \varepsilon}\right) \\ &= \overline{\mu} h_1 \left(\frac{\overline{M} \cdot (e^{\overline{\mu} H} - 1)}{\overline{M} \cdot (e^{\overline{\mu} H} - 1) - \overline{\mu}^2 \varepsilon}\right) \le 0. \end{aligned}$$

Focusing on the case $\overline{\mu} < 0$: we claim that

$$\delta > \frac{\overline{M} h_1}{2\overline{\mu} + \overline{\mu}^2 h_1}(e^{\overline{\mu} H} - 1)$$

is equivalent to

$$h_1 < \frac{2\overline{\mu}\delta}{\overline{M} \cdot (e^{\overline{\mu} H} - 1) - \overline{\mu}^2 \delta}.$$

This is verified by direct algebraic manipulation. **Q.E.D.**

Lemma 6

Let $H > 0$, $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$, and $E_0 \subseteq \mathbb{R}^n$. If

$$\overline{B} := \sum_{i=0}^{k-1} [0, H]^i \boldsymbol{f}^{[i]}(E_0) + Box(-\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}) \quad \textit{and} \quad M := \sup_{p \in \overline{B}} \|\boldsymbol{f}^{[k]}(\boldsymbol{p})\|_2,$$

then an ε-admissible pair for E_0 is given by (h, F_1) where

$$h = \min\left\{H, \min_{i=1}^{n}\left(\frac{\varepsilon_i}{M_i}\right)^{1/k}\right\} \quad and \quad F_1 = \sum_{i=0}^{k-1}[0,h]^i \boldsymbol{f}^{[i]}(E_0) + Box(-\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}). \tag{39}$$

Proof. To verify (7), we only need to verify

$$[0,h]^k \boldsymbol{f}^{[k]}(F_1) \subseteq Box(-\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}).$$

We have:

$$\begin{aligned} h^k \boldsymbol{f}^{[k]}(F_1) &\subseteq h^k[-\overline{M}, \overline{M}]^n && \text{(by the definition of } \overline{M}\text{)} \\ &\subseteq Box(-\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}). && \text{(by the definition of } h\text{)} \end{aligned}$$

Q.E.D.

Lemma 7
Consider an admissible triple (E_0, H, F_1) where $E_0 := Ball(\boldsymbol{p}_0, r_0)$.
Let $\boldsymbol{q}_0 = \boldsymbol{p}_0 + h_1 f(\boldsymbol{p}_0)$ be obtained from $\boldsymbol{p}_0$ by an Euler step of size h_1.
If $h_1 \le h^{euler}(H, \overline{M}, \overline{\mu}, \delta)$ (cf. (11)), where $\overline{\mu} = \mu_2(J_{\boldsymbol{f}}(F_1))$, $\overline{M} = \|\boldsymbol{f}^{[2]}(F_1)\|$, and $\delta > 0$, then:

(a) The linear function $\ell(t) := (1 - t/h_1)\boldsymbol{p}_0 + (t/h_1)\boldsymbol{q}_0$ lies in the δ-tube of $\boldsymbol{x}_0 = \mathtt{IVP}(\boldsymbol{p}_0, H)$.

(b) An end-enclosure for $\mathtt{IVP}(E_0, h_1)$ is given by $Ball(\boldsymbol{q}_0, r_0 e^{\overline{\mu} h_1} + \delta)$.

(c) A full-enclosure for $\mathtt{IVP}(E_0, h_1)$ is given by $CHull(Ball(\boldsymbol{p}_0, r'), Ball(\boldsymbol{q}_0, r'))$ where $r' = \delta + \max(r_0 e^{\overline{\mu} h_1}, r_0)$.

Proof.

(a) By Lemma 5 we have $\ell(t)$ lies in the δ-tube of $\boldsymbol{x}_0$, since for any $t \in [0, h_1]$, $\|\ell(t) - \boldsymbol{x}_c(t)\| \le \delta$.

(b) By Corollary 4 we have for any $\boldsymbol{x} \in \mathtt{IVP}(Ball(\boldsymbol{p}_0, r_0), h_1, F_1)$, $\|\boldsymbol{x}(h_1) - \boldsymbol{x}_0(h_1)\| \le r_0 e^{\overline{\mu} h_1}$. Since $\|\boldsymbol{q}_0 - \boldsymbol{x}_0(h_1)\|_2 \le \delta$, then by the triangular inequality we have

$$\|\boldsymbol{q}_0 - \boldsymbol{x}(h_1)\|_2 \le \|\boldsymbol{x}(h_1) - \boldsymbol{x}_0(h_1)\| + \|\boldsymbol{q}_0 - \boldsymbol{x}_0(h)\|_2 \le r.$$

So, $\boldsymbol{x}(h_1) \in Ball(\boldsymbol{q}_0, r_0 e^{\overline{\mu} h} + \delta)$.

(c) We show that for any $T \in [0, h_1]$, the end-enclosure of $\mathtt{IVP}(E_0, T)$ is a subset of $Box(Ball(\boldsymbol{p}_0, r' + \delta), Ball(\boldsymbol{q}_0, r' + \delta))$. Note that $E_1 = Ball(l(T), r_0 e^{\overline{\mu} T} + \delta)$ is the end-enclosure for $\mathtt{IVP}(E_0, T)$.

Let $l(T)_i$ denote the i-th component of $l(T)$ and $r(T) := r_0 e^{\overline{\mu} T} + \delta$. Then, we only need to prove that for any $i = 1, \ldots, n$, the interval $l(T)_i \pm r(T)$ satisfies

$$l(T)_i \pm r(T) \subseteq Box((\boldsymbol{p}_0)_i \pm (r' + \delta), (\boldsymbol{q}_0)_i \pm (r' + \delta)),$$

where $(\boldsymbol{p}_0)_i$ and $(\boldsymbol{q}_0)_i$ are the i-th components of $\boldsymbol{p}_0$ and $\boldsymbol{q}_0$, respectively.

Since $l(T)$ is a line segment, it follows that

$$\min((\boldsymbol{q}_0)_i, (\boldsymbol{p}_0)_i) \le l(T)_i \le \max((\boldsymbol{q}_0)_i, (\boldsymbol{p}_0)_i).$$

Additionally, we have $r(T) \le r' + \delta$.

Combining these observations, we conclude that

$$l(T)_i \pm r(T) \subseteq Box((\boldsymbol{p}_0)_i \pm (r' + \delta), (\boldsymbol{q}_0)_i \pm (r' + \delta)).$$

Q.E.D.

Lemma A.2

(a) $\boldsymbol{g}(\boldsymbol{y}) = J_{\widehat{\pi}}(\widehat{\pi}^{-1}(\boldsymbol{y})) \bullet \overline{\boldsymbol{g}}(\widehat{\pi}^{-1}(\boldsymbol{y}))$

$= \mathtt{diag}(-d_i y_i^{1+\frac{1}{d_i}} : i = 1, \ldots, n) \bullet \overline{\boldsymbol{g}}(\widehat{\pi}^{-1}(\boldsymbol{y})).$

(b) The Jacobian matrix of $\boldsymbol{g}$ *with respect to* $\boldsymbol{y} = (y_1, \ldots, y_n)$ *is:*

$$J_{\boldsymbol{g}}(\boldsymbol{y}) = A(\boldsymbol{y}) + P^{-1}(\boldsymbol{y}) \bullet J_{\overline{\boldsymbol{g}}}(\widehat{\pi}^{-1}(\boldsymbol{y})) \bullet P(\boldsymbol{y}), \tag{40}$$

where

$$A(\boldsymbol{y}) = \mathtt{diag}\Big(- (d_i + 1) y_i^{\frac{1}{d_i}} \cdot (\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_i) : i = 1, \ldots, n\Big)$$

and

$$P(\boldsymbol{y}) = \mathtt{diag}\Big(\frac{\overline{\pi}^{-1}(\boldsymbol{y})_i^{d_i+1}}{d_i} : i = 1, \ldots, n\Big).$$

Proof.

(a) For each $i = 1, \ldots, n$, we have from (21) that $y_i' = g_i(\boldsymbol{y})$ where $\boldsymbol{y} = (y_1, \ldots, y_n)$, $\boldsymbol{g} = (g_1, \ldots, g_n)$, i.e.,

$$\begin{aligned}
g_i(\boldsymbol{y}) = y_i' &= \left(\frac{1}{\overline{y}_i^{d_i}}\right)' \qquad \text{(by (21) and } y_i = \overline{y}_i^{-d_i}) \\
&= -d_i \overline{y}_i^{-(d_i+1)} \overline{y}_i' \\
&= -d_i y_i^{1+\frac{1}{d_i}} \Big(\overline{\boldsymbol{g}}\big(y_1^{-\frac{1}{d_1}}, \ldots, y_n^{-\frac{1}{d_n}}\big)\Big)_i \\
&= -d_i y_i^{1+\frac{1}{d_i}} (\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_i.
\end{aligned}$$

Thus,

$$\boldsymbol{g}(\boldsymbol{y}) = (g_1(\boldsymbol{y}), \ldots, g_n(\boldsymbol{y})) = \mathtt{diag}(-d_i y_i^{1+\frac{1}{d_i}}, i = 1, \ldots, n) \bullet \overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y}))$$

(b) By plugging $g_i(\boldsymbol{y}) = -d_i y_i^{1+\frac{1}{d_i}} (\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_i$ into the Jacobian, we get

$$\begin{aligned}
J_{\boldsymbol{g}}(\boldsymbol{y}) &= \begin{bmatrix} \nabla(g_1(\boldsymbol{y})) \\ \vdots \\ \nabla(g_n(\boldsymbol{y})) \end{bmatrix} = \begin{bmatrix} \nabla(-d_i y_1^{1+\frac{1}{d_i}} (\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_1) \\ \vdots \\ \nabla(-d_n y_n^{1+\frac{1}{d_n}} (\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_n) \end{bmatrix} \\
&= \begin{bmatrix} \nabla(-d_1 y_1^{1+\frac{1}{d_1}})(\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_1 \\ \vdots \\ \nabla(-d_n y_n^{1+\frac{1}{d_n}})(\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_n \end{bmatrix} + \begin{bmatrix} -d_1 y_1^{1+\frac{1}{d_1}} \nabla((\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_1) \\ \vdots \\ -d_n y_n^{1+\frac{1}{d_n}} \nabla((\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_n) \end{bmatrix}
\end{aligned} \tag{41}$$

Note that for any $i = 1, \ldots, n$,

$$\nabla(-d_i y_i^{1+\frac{1}{d_i}})(\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_i = \left(0, \ldots, 0, -d_i \left(1 + \tfrac{1}{d_i}\right) y_i^{\frac{1}{d_i}} (\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_i, \ldots, 0\right)$$

and

$$\begin{aligned}
-d_i y_i^{1+\frac{1}{d_i}} \nabla((\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_i) &= \left(- d_i y_i^{1+\frac{1}{d_i}} \frac{\partial(\overline{\boldsymbol{g}}(\boldsymbol{x}))_i}{\partial x_j}(\overline{\pi}^{-1}(\boldsymbol{y}))\frac{\partial \overline{\pi}^{-1}(\boldsymbol{y})}{\partial \boldsymbol{y}} \,:\, j = 1, \ldots, n \right) \\
&= \left(\frac{d_i}{d_j} \left(\frac{y_i^{1+\frac{1}{d_i}}}{y_j^{1+\frac{1}{d_j}}} \right) \frac{\partial(\overline{\boldsymbol{g}}(\boldsymbol{x}))_i}{\partial x_j}(\overline{\pi}^{-1}(\boldsymbol{y})) \,:\, j = 1, \ldots, n \right) \\
&= \left(\frac{d_i}{d_j} \overline{\pi}^{-1}(\boldsymbol{y})_j^{d_j+1} \frac{\partial(\overline{\boldsymbol{g}}(\boldsymbol{x}))_i}{\partial x_j}(\overline{\pi}^{-1}(\boldsymbol{y}))\overline{\pi}^{-1}(\boldsymbol{y})_i^{-d_i-1} \,:\, j = 1, \ldots, n \right).
\end{aligned}$$

Thus,

$$J_{\boldsymbol{g}}(\boldsymbol{y}) = A(\boldsymbol{y}) + P^{-1}(\boldsymbol{y}) \bullet J_{\overline{\boldsymbol{g}}}(\overline{\pi}^{-1}(\boldsymbol{y})) \bullet P(\boldsymbol{y}),$$

where

$$A(\boldsymbol{y}) = \mathtt{diag}(-(d_1+1)y_1^{\frac{1}{d_1}}(\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_1), \ldots, -(d_n+1)y_n^{\frac{1}{d_n}}(\overline{\boldsymbol{g}}(\overline{\pi}^{-1}(\boldsymbol{y})))_n)$$

and

$$P(\boldsymbol{y}) = \mathtt{diag}(\frac{\overline{\pi}^{-1}(\boldsymbol{y})_1^{d_1+1}}{d_1}, \ldots, \frac{\overline{\pi}^{-1}(\boldsymbol{y})_n^{d_n+1}}{d_n}).$$

Q.E.D.

Theorem 8

(a)

$$\begin{aligned}
\mu_2\big(J_{\boldsymbol{g}}(\pi(F_1))\big) \quad \leq \quad & \max\left\{ \frac{-(d_i+1)}{\check{b}_i} \,:\, i = 1, \ldots, n \right\} \\
& + \max_{i=1}^n \left\{ d_i \right\} \cdot \|J_{\overline{\boldsymbol{g}}}(\overline{\pi}(F_1))\|_2 \cdot \max_{i=1}^n \left\{ \frac{(\check{b}_i)^{d_i+1}}{d_i} \right\}.
\end{aligned}$$

(b) If $d_1 = \cdots = d_n = d$ then

$$\mu_2\left(J_{\boldsymbol{g}}(\pi(F_1))\right) \leq -(d+1)\frac{1}{\check{b}_{\max}} + (\check{b}_{\max})^{d+1}\|J_{\overline{\boldsymbol{g}}}(\overline{\pi}(F_1))\|_2.$$

Proof. From **Lemma A.2(b)** we have for any $\boldsymbol{p} = (p_1, \ldots, p_n) \in \overline{\pi}(F_1)$,

$$J_{\boldsymbol{g}}(\widehat{\pi}(\boldsymbol{p})) = A(\boldsymbol{p}) + P^{-1}(\boldsymbol{p})\frac{\partial \overline{\boldsymbol{g}}}{\partial \boldsymbol{x}}(\boldsymbol{p})P(\boldsymbol{p}) \tag{42}$$

where $P(\boldsymbol{p}) = \mathtt{diag}\big(\frac{p_i^{d_i+1}}{d_i} \,:\, i = 1, \ldots, n\big)$ and $A(\boldsymbol{p}) = \mathtt{diag}(a_1, \ldots, a_n)$ with

$$a_i := -\, d_i(1 + \frac{1}{d_i})p_i^{-1} \cdot (\overline{\boldsymbol{g}}(\boldsymbol{p}))_i. \tag{43}$$

Thus, A, P are diagonal matrices and p_i^{-1} is well-defined since $\boldsymbol{p} \in B_2 \geq \boldsymbol{1}$, (28).

By Lemma 2(b) and (43), we conclude that the form

$$\mu_2(A(\boldsymbol{p})) = \mu_2(\mathtt{diag}(a_1, \ldots, a_n)) = \max\left\{ a_i \,:\, i = 1, \ldots, n \right\}. \tag{44}$$

From (29), we conclude that

$$\begin{aligned}
\mu_2\big(J_g(\widehat{\pi}(\boldsymbol{p}))\big) &= \mu_2\left(A(\boldsymbol{p}) + P^{-1}(\boldsymbol{p})\tfrac{\partial \overline{g}}{\partial \boldsymbol{x}}(\boldsymbol{p})P(\boldsymbol{p})\right) \\
&\quad \text{(by (42))} \\
&\le \mu_2(A(\boldsymbol{p})) + \mu_2\left(P^{-1}(\boldsymbol{p})\tfrac{\partial \overline{g}}{\partial \boldsymbol{x}}(\boldsymbol{p})P(\boldsymbol{p})\right) \\
&\quad \text{(by Lemma 2(a))} \\
&\le \mu_2(A(\boldsymbol{p})) + \left\|P^{-1}(\boldsymbol{p})\tfrac{\partial \overline{g}}{\partial \boldsymbol{x}}(\boldsymbol{p})P(\boldsymbol{p})\right\|_2 \\
&\quad \text{(by Lemma 2(b))} \\
&\le \max\left\{\tfrac{-(d_i+1)}{\check{b}_i} : i = 1,\dots,n\right\} \\
&\quad + \left\|P^{-1}(\boldsymbol{p})\right\| \left\|\tfrac{\partial \overline{g}}{\partial \boldsymbol{x}}(\boldsymbol{p})\right\| \left\|P(\boldsymbol{p})\right\| \\
&\quad \text{(by (8))} \\
&\le \max\left\{\tfrac{-(d_i+1)}{\check{b}_i} : i = 1,\dots,n\right\} \\
&\quad + \max_{i=1}^n \left\{d_i\right\} \cdot \|J_{\overline{g}}(\overline{\pi}(F_1))\|_2 \cdot \max_{i=1}^n \left\{\tfrac{(\check{b}_i)^{d_i+1}}{d_i}\right\}.
\end{aligned}$$

Q.E.D.

Lemma 9 *If $d \ge \overline{d}(F_1)$, we have:*

(a) $\mu_2\left(J_g(\pi(F_1))\right) \le (-2 + (\check{b}_{\max})^{d+2}) \cdot \frac{\|J_{\overline{g}}(\overline{\pi}(F_1))\|_2}{\check{b}_{\max}}$.

(b) If $\log_2(\check{b}_{\max}) < \frac{1}{d+2}$ *then* $\mu_2\left(J_g(\pi(F_1))\right) < 0$.

Proof.

(a) By Theorem 8 we have

$$\begin{aligned}
\mu_2\left(J_g(\pi(F_1))\right) &\le -(d+1)\frac{1}{\check{b}_{\max}} + (\check{b}_{\max})^{d+1}\|J_{\overline{g}}(\overline{\pi}(F_1))\|_2 \\
&= \left(\tfrac{-(d+1)}{\|J_{\overline{g}}(\overline{\pi}(F_1))\|_2} + (\check{b}_{\max})^{d+2}\right) \cdot \tfrac{\|J_{\overline{g}}(\overline{\pi}(F_1))\|_2}{\check{b}_{\max}} \\
&\quad \text{(by factoring)} \\
&\le \left(-2 + (\check{b}_{\max})^{d+2}\right) \cdot \tfrac{\|J_{\overline{g}}(\overline{\pi}(F_1))\|_2}{\check{b}_{\max}} \\
&\quad \text{(By eqn.(30), we have } (d+1) \ge 2(\|J_{\overline{g}}(\overline{\pi}(F_1))\|_2)\text{).}
\end{aligned}$$

(b) Since $(\check{b}_{\max})^{d+2} < 2$ is equivalent to $\log_2(\check{b}_{\max}) < \frac{1}{d+2}$, we conclude that $\mu_2\left(J_g(\pi(F_1))\right) < 0$.

Q.E.D.

Lemma A.4

Let $\boldsymbol{p}, \boldsymbol{q} \in B \subseteq \mathbb{R}^n$ and $\phi \in C^1(F_1 \to \mathbb{R}^n)$, then $\|\phi(\boldsymbol{p}) - \phi(\boldsymbol{q})\|_2 \le \|J_\phi(B)\|_2 \cdot \|\boldsymbol{p} - \boldsymbol{q}\|_2$

Proof.

$$\begin{aligned}
\|\phi(\boldsymbol{p}) - \phi(\boldsymbol{q})\|_2 &\le \|\phi(\boldsymbol{q}) + J_\phi(\xi) \bullet (\boldsymbol{p} - \boldsymbol{q}) - \phi(\boldsymbol{q})\|_2 \\
&\quad \text{(by Taylor expansion of } \phi(\boldsymbol{p}) \text{ at } \boldsymbol{q}\text{)} \\
&= \|J_\phi(\xi) \bullet (\boldsymbol{p} - \boldsymbol{q})\|_2 \\
&\le \|J_\phi(\xi)\|_2 \cdot \|(\boldsymbol{p} - \boldsymbol{q})\|_2 \\
&\le \|J_\phi(B)\|_2 \cdot \|(\boldsymbol{p} - \boldsymbol{q})\|_2,
\end{aligned}$$

where $\xi \in B$. **Q.E.D.**

Lemma 10
Let $\boldsymbol{y} = \pi(\boldsymbol{x})$ *and*

$$\begin{array}{rcl} \boldsymbol{x} & \in & IVP_f(\boldsymbol{x}_0, h, F_1), \\ \boldsymbol{y} & \in & IVP_g(\pi(\boldsymbol{x}_0), h, \pi(F_1)). \end{array}$$

For any $\delta > 0$ *and any point* $\boldsymbol{p} \in \mathbb{R}^n$ *satisfying*

$$\|\pi(\boldsymbol{p}) - \boldsymbol{y}(h)\|_2 \le \frac{\delta}{\|J_{\pi^{-1}}(\pi(F_1))\|_2},$$

we have

$$\|\boldsymbol{p} - \boldsymbol{x}(h)\|_2 \le \delta.$$

Proof.

$$\begin{array}{rcl} \|\boldsymbol{p} - \boldsymbol{x}(h)\|_2 & = & \|\pi^{-1}(\pi(\boldsymbol{p})) - \pi^{-1}(\pi(\boldsymbol{x}(h)))\|_2 \\ & = & \|\pi^{-1}(\pi(\boldsymbol{p})) - \pi^{-1}(\boldsymbol{y}(h))\|_2 \\ & \le & \|J_{\pi^{-1}}(\pi(F_1))\|_2 \cdot \|\pi(\boldsymbol{p}) - \boldsymbol{y}(h)\|_2 \\ & & \text{(by } \textbf{Lemma A.4}\text{)} \\ & \le & \delta \qquad \text{(by condition (31).)} \end{array}$$

Q.E.D.

Theorem 11 *The subroutine* $\mathcal{S}.\mathtt{Refine}(\varepsilon)$ *is correct. In particular, it halts.*

Proof. The proof is in two parts: (a) partial correctness and (b) termination. Assume the scaffold $\mathcal{S}$ has m stages and the input for `Refine` is $\varepsilon > 0$.

(a) Partial correctness is relatively easy, so we give sketch a broad sketch: we must show that if the `Refine` halts, then its output is correct, i.e., $w_{\max}(E_m(\mathcal{S})) < \varepsilon$. The first line of `Refine` initializes r_0 to $w_{\max}(E_m(\mathcal{S}))$. If $r_0 < \varepsilon$, then we terminate without entering the while-loop, and the result hold. If we enter the while-loop, then we can only exit the while-loop if the last line of the while-body assigns to r_0 a value $w_{\max}(E_m(\mathcal{S}))$ less than ε. Again this is correct.

(b) The rest of the proof is to show that `Refine` halts. We will prove termination by way of contradiction. If `Refine` does not terminate, then it has infinitely many **phases** where the kth phase ($k = 1, 2, \ldots$) refers to the k iterate of the while-loop.

(H1) We will show that $\lim_{k\to\infty} \delta_i^k = 0$ for each $i = 1, \ldots, m$ in (36). This will yield a contradiction.

(H2) For a fixed stage i, we see the number of times that `Refine` calls `EulerTube` is

$$d_i^k := \log_2\left(\frac{\delta_i^1}{\delta_i^k}\right).$$

Similarly, the number of times `Refine` calls `Bisect` is

$$\ell_i^k - \ell_i^1$$

where ℓ_i^k is the level of the (k, i) phase-stage. So,

$$k = d_i^k + (\ell_i^k - \ell_i^1) \tag{45}$$

since each phase calls either `Bisect` or `EulerTube`. Hence $k \to \infty$.

(H3) CLAIM: $\lim_{k\to\infty} d_i^k \to \infty$, i.e., `EulerTube` is called infinitely often. By way of contradictor, suppose d_i^k has an upper bound, say $\overline{d}_i^k$. Since μ_2, Δt_i, and $\overline{M}$ are bounded, we have $\hat{h} \geq C \cdot 2^{\overline{d}_i^k}$ in `Refine` (see (11)), where $C > 0$ is a constant.

Note that each `Bisect`(i) increments the level and thus halves H. Therefore, once $H < C \cdot 2^{\overline{d}_i^k}$, `Bisect` will no longer be called. This implies that $\lim_{k\to\infty}(\ell_i^k - \ell_i^1)$ is finite. This is a contradicts the fact that $k \to \infty$ since both d_i^k and $(\ell_i^k - \ell_i^1)$ are bounded. Thus, our CLAIM is proved.

It follows from the CLAIM that $\lim_{k\to\infty} \delta_i^k = 0$, since `EulerTube` is called infinitely often, and after each call, δ_i^k is halved.

(H4) Consider (k, i) as a **phase-stage**: define r_i^k as the radius of the circumball of $E_i(S)$ at phase k. For instance, we terminate in phase k if $k \geq 0$ is the first phase to satisfy $r_m^k < \frac{1}{2}\varepsilon_0$.

Since we call `EulerTube` infinitely often, and each call ensures that the target δ_i^k in (18) is reached:

$$r_i^k \leq r_{i-1}^k e^{\mu_i^k \Delta t_i} + \delta_i^k, \tag{46}$$

where μ_i^k (computed as μ_2 in `Refine`) is an upper bound for the logarithmic norm over the full enclosure of the ith stage.

(H5) A **chain** is a sequence $C_1 = (1 \leq k(1) \leq k(2) \leq \cdots \leq k(m))$ such that `EulerTube` is called in phase-stage $(k(i), i)$ for each $i = 1, \ldots, m$. The chain contains m inequalities of the form (46), and we can telescope them into a single inequality.

But first, to simplify these inequalities, let $\overline{\mu}$ be the largest value of μ_i^1 for $i = 1, \ldots, m$, $\Delta = \Delta(C_1)$ is the maximum of $\delta_i^{k(1)}$, and h_i be the step size of the ith stage:

$$\begin{aligned}
r_m^{k(m)} &\leq (r_{m-1}^{k(m)})e^{\mu_i^{k(m)}h_i} + \delta_i^{k(m)} && \text{(by (46) for } (k(m), m))\\
&\leq (r_{m-1}^{k(m-1)})e^{\mu_i^{k(m)}h_i} + \delta_i^{k(m)} && \text{(since } r_{m-1}^{k(m)} \leq r_{m-1}^{k(m-1)})\\
&\leq ((r_{m-2}^{k(m-1)})e^{\mu_{i-1}^{k(m-1)}h_{i-1}} + \delta_{i-1}^{k(m-1)})e^{\mu_i^{k(m)}h_i} + \delta_i^{k(m)} && \text{(by (46) for } (k(m-1), m-1))\\
&\leq ((r_{m-2}^{k(m-1)})e^{\overline{\mu}h_{i-1}} + \Delta)e^{\overline{\mu}h_i} + \Delta && \text{(simplify using } \overline{\mu}, h_i, \Delta)\\
&\vdots\\
&\leq (r_0^{k(1)})e^{\overline{\mu}\sum_{j=1}^m h_j} + \Delta\sum_{j=0}^{m-1} e^{\overline{\mu}\sum_{i=j+1}^m h_i}\\
&\leq e^{\overline{\mu}}(r_0^{k(1)} + \Delta \cdot m) && \text{(since } 1 \geq \textstyle\sum_{i=1}^m h_i)
\end{aligned}$$

To summarize what we just proved[13] about a chain C_1, let $r_m(C_1)$ denote $r_m^{k(m)}$ and $r_0(C_1)$ denote $r_0^{k(1)}$ the following C_1**-inequality**:

$$r_m(C_1) \leq e^{\overline{\mu}}(r_0(C_1) + \Delta(C_1) \cdot m). \tag{47}$$

(H6) If $C = (1 \leq k(1) \leq \cdots \leq k(m))$ and $C' = (1 \leq k'(1) \leq \cdots \leq k'(m))$ are two chains where $k(i) < k'(i)$ for $i = 1, \ldots, m$, then we write $C < C'$.

LEMMA: If $C < C'$ then

$$r_m(C') \leq \tfrac{1}{2}e^{\overline{\mu}}(r_0(C) + \Delta(C) \cdot m)$$

(H7) It is easy to show that there exists an infinite sequence of chains

$$C_1 < C_2 < C_3 < \cdots.$$

[13] This proof assumes that $\mu_i^k \leq \mu_i^{k-1}$. This is true if our interval computation of μ_2 is isotonic. But we can avoid isotony by defining μ_i^k to be μ_i^{k-1} if the computation returns a larger value.

This comes from the fact that for each $i = 1, \ldots, m$, there are infinitely many phases that calls `EulerTube`. It follows by induction using the previous LEMMA that, for each $i \geq 2$,

$$r_m(C_i) \leq (\tfrac{1}{2})^i e^{\overline{\mu}}(r_0(C_1) + \Delta(C_1) \cdot m)$$

This proves that $\lim_{i\to\infty} r_m(C_i) = 0$. This contradicts the non-termination of `Refine`.

Q.E.D.

Theorem 12 *Algorithm* $\mathtt{EndEnc}(B_0, \varepsilon_0)$ *halts, provided the interval computation of* `StepA` *is isotonic. The output is also correct.*

Proof.

If the algorithm terminates, its correctness is ensured by the conclusions in Section 4.

We now proceed to prove the termination of the algorithm. Specifically, we need to show that the loop in the algorithm can terminate, which means that the time variable t can reach 1. It suffices to demonstrate that for any inputs B_0 and $\varepsilon_0 > 0$, there exists a lower bound $\underline{h} > 0$ such that for the ith iteration of the loop has step size $\Delta t_i = h_i \geq \underline{h}$.

First we define the set $\overline{E} := \mathtt{image}(\mathtt{IVP}(B_0, 1)) + [-\varepsilon_0, \varepsilon_0]^n$. Since $\mathtt{IVP}(B_0, 1)$ is valid, $\overline{E}$ is a bounded set. Let the pair $(\underline{h}, \overline{F})$ be the result of calling the subroutine $\mathtt{StepA}(\overline{E}, 1, \varepsilon_0)$. Note that `StepA` is implicitly calling box functions to compute $\underline{h}, \overline{F}$ (see Subsection 2.2), and thus $\underline{h}$ is positive. Whenever we call `StepA` in our algorithm, its arguments are (E, H, ε_0) for some $E \subseteq \overline{E}$ and $H \leq 1$. If $\mathtt{StepA}(E, H, \varepsilon_0) \to (h, F)$, then $h \geq \underline{h}$, provided[14] `StepA` is isotonic. This proves that the algorithm halts in at most $\lceil 1/\underline{h} \rceil$ steps. **Q.E.D.**

B. Appendix B: The affine map $\overline{\pi}$

Consider the condition (27). Without loss of generality, assume $0 \notin \overline{I}_1$. To further simplify our notations, we assume

$$\overline{I}_1 > 0. \tag{48}$$

In case $\overline{I}_1 < 0$, we shall indicate the necessary changes to the formulas. We first describe an invertible linear map $\widetilde{\pi} : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\widetilde{\pi}(\boldsymbol{f}(B_1)) > \mathbf{1} = (1, \ldots, 1) \qquad \text{(Greater-than-One Property of } \widetilde{\pi}) \tag{49}$$

Note that (49) means that for each $i = 1, \ldots, n$, the ith component $(\widetilde{\pi}(\boldsymbol{f}(B_1)))_i$ is greater than one.

To define $\widetilde{\pi}$, we first introduce the box $\widetilde{B}_1$:

$$\begin{aligned} \widetilde{B}_1 &:= Box(\boldsymbol{f}(B_1)) \\ &= \textstyle\prod_{i=1}^n \overline{I}_i \quad \text{(implicit definition of } \overline{I}_i) \\ &= \textstyle\prod_{i=1}^n [\overline{a}_i, \overline{b}_i] \quad \text{(implicit definition of } \overline{a}_i, \overline{b}_i) \end{aligned} \tag{50}$$

where $Box(S) \in \square\mathbb{R}^n$ is the smallest box containing a set $S \subseteq \mathbb{R}^n$. For instance, $\overline{I}_i = f_i(B_1)$ where $\boldsymbol{f} = (f_1, \ldots, f_n)$. The assumption (48) says that $\overline{I}_1 > 0$, i.e., either $\overline{a}_1 > 0$.

We now define the map $\widetilde{\pi} : \mathbb{R}^n \to \mathbb{R}^n$ as follows: $\widetilde{\pi}(x_1, \ldots, x_n) = (\widetilde{x}_1, \ldots, \widetilde{x}_n)$ where

$$\widetilde{x}_i := \begin{cases} \frac{x_i}{\overline{a}_i} & \text{if } \overline{a}_i > 0, \quad \text{(i.e., } f_i(B_1) > 0) \\ \frac{x_i}{\overline{b}_i} & \text{else if } \overline{b}_i < 0, \quad \text{(i.e., } f_i(B_1) < 0) \\ x_i + x_1\left(\frac{1+\overline{b}_i - \overline{a}_i}{\overline{a}_1}\right) & \text{else} \quad \text{(i.e., } 0 \in f_i(B_1)). \end{cases} \tag{51}$$

[14] If computes $h > \underline{h}$, we could not "simply" set h to be $\underline{h}$ because we do not know how to compute a corresponding full enclosure. Note that $\overline{E}$ is a full enclosure, but we do not know how to compute it.

Note that if $\overline{I}_1 < 0$, we only need to modify the third clause in (51) to $x_i + x_1\left(\frac{1+\overline{b}_i-\overline{a}_i}{\overline{b}_1}\right)$.

Observe that $\widetilde{\pi}(\widetilde{B}_1)$ is generally a parallelopiped, not a box. Even for $n = 2$, $\widetilde{\pi}(\widetilde{B}_1)$ is a parallelogram. So we are interested in the box $Box(\widetilde{\pi}(\widetilde{B}_1))$:

$$\begin{aligned} B_1' := Box(\widetilde{\pi}(\widetilde{B}_1)) &= \prod_{i=1}^n I_i' && \text{(implicit definition of } I_i') \\ &= \prod_{i=1}^n [a_i', b_i'] && \text{(implicit definition of } a_i', b_i') \end{aligned} \tag{52}$$

Then we have the following results:

Lemma B.1

(a) $\widetilde{\pi}$ is an invertible linear map given by

$$\widetilde{\pi}(\boldsymbol{x}) = \overline{A} \bullet \boldsymbol{x} \tag{53}$$

$\frac{1}{\overline{a}_i}$, $\frac{1}{\overline{b}_i}$ *or* 1 *along the diagonal and other non-zero entries in column 1 only, Here's the revised version with improved clarity and formatting:*

$$\begin{bmatrix} v_1 & & & & \\ c_2 & v_2 & & & \\ c_3 & & v_3 & & \\ \vdots & & & \ddots & \\ c_n & & & & v_n \end{bmatrix}$$

where

$$v_i = \begin{cases} \frac{1}{\overline{a}_i} & \text{if } \overline{a}_i > 0, \\ \frac{1}{\overline{b}_i} & \text{else if } \overline{b}_i < 0, \\ 1 & \text{else} \end{cases}$$

$$c_i = \begin{cases} 0 & \text{if } 0 \notin f_i(B_1), \\ \frac{1+\overline{b}_i-\overline{a}_i}{\overline{a}_1} & \text{else.} \end{cases}$$

(b) The box $Box(\widetilde{\pi}(\widetilde{B}_1)) = \prod_{i=1}^n I_i'$ is explicitly given by

$$I_i' = \begin{cases} \left[1, \frac{\overline{b}_i}{\overline{a}_i}\right] & \text{if } \overline{a}_i > 0, \\ \left[1, \frac{\overline{a}_i}{\overline{b}_i}\right] & \text{else if } \overline{b}_i < 0, \\ \left[1+\overline{b}_i, \frac{\overline{b}_1}{\overline{a}_1}\left(1+\overline{b}_i\left(1+\frac{\overline{a}_1}{\overline{b}_1}\right)-\overline{a}_i\right)\right] & \text{else .} \end{cases} \tag{54}$$

(c) The map $\widetilde{\pi}$ has the positivity property of (49).

Proof.

(a) From the definition of $\widetilde{\pi}$ in (51), we see that the matrix $\overline{A}$ matrix the form described in the lemma. This matrix is clearly invertible.

(b) We derive explicit formulas for I_i' in each of the 3 cases:

- If $\overline{a}_i > 0$, then it is clear that $(\widetilde{\pi}(\boldsymbol{B}_1))_i = \left[1, \frac{\overline{b}_i}{\overline{a}_i}\right]$.
- Else if $\overline{b}_i < 0$, it is also clear that $(\widetilde{\pi}(\boldsymbol{B}_1))_i = \left[1, \frac{\overline{a}_i}{\overline{b}_i}\right]$.
- Else, we consider an arbitrary point $\boldsymbol{x} = (x_1, \ldots, x_n) \in \widetilde{\boldsymbol{B}}_1$:

$$\begin{array}{rcll}
(\widetilde{\pi}(\boldsymbol{x}))_i & = & x_i + x_1\left(\frac{1+\overline{b}_i-\overline{a}_i}{\overline{a}_1}\right) & \text{(by definition)} \\
& \geq & \overline{a}_i + \overline{a}_1\left(\frac{1+\overline{b}_i-\overline{a}_i}{\overline{a}_1}\right) & \\
& & (x_j \in [\overline{a}_j, \overline{b}_j]\ (\forall\, j)\ \&\ (1+\overline{b}_i-\overline{a}_i)/\overline{a}_1 > 0)) & \\
& = & 1 + \overline{b}_i. & \\
(\widetilde{\pi}(\boldsymbol{x}))_i & = & x_i + x_1\left(\frac{1+\overline{b}_i-\overline{a}_i}{\overline{a}_1}\right) & \\
& \leq & \overline{b}_i + \overline{b}_1\left(\frac{1+\overline{b}_i-\overline{a}_i}{\overline{a}_1}\right) & \\
& & (x_j \in [\overline{a}_j, \overline{b}_j] \text{ and } (1+\overline{b}_i-\overline{a}_i)/\overline{a}_1 > 0) & \\
& = & \frac{\overline{b}_1}{\overline{a}_1}\left(1 + \overline{b}_i(1 + \frac{\overline{a}_1}{\overline{b}_1}) - \overline{a}_i\right). &
\end{array}$$

Since both the upper and lower bounds are attainable, they determine the interval I_i' as claimed.

(c) It is sufficient to show that $I_i' \geq 1$. This is clearly true for the first two clauses of (54). For the last two clauses, we have $I_i' \geq 1 + \overline{b}_i$ by part(b). The result follows since $0 \leq \overline{b}_i$.

Q.E.D.

Let

$$\begin{array}{rcll}
B_1^* & := & Box(\widetilde{\pi}(B_1)) & \\
& = & \prod_{i=1}^n I_i^* & \text{(implicit definition of } I*_i) \\
& = & \prod_{i=1}^n [a_i^*, b_i^*] & \text{(implicit definition of } a_i^*, b_i^*)
\end{array} \tag{55}$$

We now define the affine map $\overline{\pi} : \mathbb{R}^n \to \mathbb{R}^n$:

$$\begin{array}{rcl}
\overline{\pi}(\boldsymbol{x}) & = & (\overline{\pi}_1(x_1), \overline{\pi}_2(x_2), \ldots, \overline{\pi}_n(x_n)) \\
& & \quad \text{where } \boldsymbol{x} = (x_1, \ldots, x_n) \text{ and} \\
\overline{\pi}_i(x) & := & \widetilde{\pi}(x) - a_i^* + 1.
\end{array} \tag{56}$$

Then we have the following results, which is property (Q2):

Lemma B.2 $\overline{\pi}(B_1) > \mathbf{1}$.

Proof. The conclusion follows from the fact that $\widetilde{\pi}(B_1) \subseteq \prod_{i=1}^n [a_i^*, b_i^*]$ and $\overline{\pi}(B_1) = \widetilde{\pi}(B_1) - (a_1^*, \ldots, a_n^*) + 1$.

Q.E.D.

References

[1] Ramon E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

[2] Kai Shen, Dillard L. Robertson, and Joseph K. Scott. Tight reachability bounds for constrained nonlinear systems using mean value differential inequalities. *Automatica*, 134:109911, 2021.

[3] Oded Goldreich. On Promise Problems (a survey). In *Theoretical Computer Science: Essays in memory of Shimon Even*, pages 254 – 290. Springer, 2006. LNCS. Vol. 3895.

[4] D.S. Graça, N. Zhong, and J. Buescu. Computability, noncomputability and undecidability of maximal intervals of IVPS. *Trans. AMS*, 361(6):2913–2927, 2009.

[5] Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.

[6] Hans J. Stetter. Validated solution of initial value problems for ODE. In Christian Ullrich, editor, *Computer Arithmetic and Self-Validating Numerical Methods*, volume 7 of *Notes and Reports in Mathematics, Science and Engineering*, pages 171–187. Academic Press, 1990.

[7] G. F. Corliss. Survey of interval algorithms for ordinary differential equations. *Appl.Math.Comp.*, 31, 1989.

[8] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[9] Juan Xu, Michael Burr, and Chee Yap. An approach for certifying homotopy continuation paths: Univariate case. In *Int'l Symp. Symbolic and Alge. Comp.*, pages 399–406, 2018. 43rd ISSAC. July 16-19. New York City.

[10] K. Makino and M. Berz. Verified computations using taylor models and their applications. In *LNCS No. 10381*, pages 3–13, 2017.

[11] Karl Nickel. How to fight the wrapping effect. In *Proc. Intl Symp. on Interval mathematics*, pages 121–132, Berlin, 1986. Springer-Verlag.

[12] R.J. Lohner. *Einschließung der Lösung gewöhnlicher Anfangs – und Randwertaufgaben und Anwendungen*. Phd thesis, Universität Karlsruhe, 1988.

[13] Florian Bünger. A Taylor Model Toolbox for solving ODEs implemented in MATLAB/INTLAB. *J. Comp. and Appl. Math.*, 368:112511, 2020.

[14] Nathalie Revol. Affine iterations and wrapping effect: Various approaches, 2022.

[15] Piotr Zgliczynski. C^1-Lohner algorithm. *Foundations of Computational Mathematics*, 2(4):429–465, 2002.

[16] Arnold Neumaier. Global, rigorous and realistic bounds for the solution of dissipative differential equations. Part I: Theory. Technical report, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974-0636, 1993.

[17] Pieter Eijgenraam. The solution of initial value problems using interval arithmetic: formulation and analysis of an algorithm. *Mathematical Centre Tracts*, No. 144, 1981.

[18] Ramon E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers. In Louis B Rall, editor, *Error in digital computation, Volume 1*, pages 61–130. Wiley New York, 1965. Proc. of an Advanced Seminar conducted by the Mathematics Research Center, United States Army, University of Wisconsin, Madison, October 5-7, 1964.

[19] Ramon E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers. In Rall [18], pages 61–130. Proc. of an Advanced Seminar conducted by the Mathematics Research Center, United States Army, University of Wisconsin, Madison, October 5-7, 1964.

[20] Robert Rihm. Interval methods for initial value problems in odes. In J. Herzberger, editor, *Topics in validated computations: Proc. of the IMACS-GAMM Int'l Workshop on Validated Computations*, Elsevier Studies in Computational Mathematics, pages 173–207. Elsevier, 1994.

[21] Florian Bünger. Preconditioning of Taylor models, implementation and test cases. *Nonlinear Theory and its Applications, IEICE*, 12(1):2–40, 2021.

[22] E. Adams, D. Cordes, and R. Lohner. Enclosure of solutions of ordinary initial value problems and applications. In E. Adams, R. Ansorge, Chr. Großmann, and H.G. Roos, editors, *Discretization in Differential Equations and Enclosures*, pages 9–28. Akademie-Verlag, Berlin, 1987.

[23] Irmina Walawska and Daniel Wilczak. An implicit algorithm for validated enclosures of the solutions to the variational equations for ODEs. *Applied Mathematics and Computation*, 291:303–322, 2016.

[24] K.R. Jackson and N. S. Nedialkov. Some recent advances in validated methods for ivps for odes. *Applied Numerical Mathematics*, 42:269–284, 2002.

[25] Nedialko Stoyanov Nedialkov. *Computing Rigorous Bounds on the Solution of an Initial Value Problem for an Ordinary Differential Equation*. PhD thesis, Department of Computer Science, University of Toronto, 1999.

[26] N. S. Nedialkov, K. R. Jackson, , and J. D. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7(6):449–465, 2001.

[27] Chuchu Fan, James Kapinski, Xiaoqing Jin, and Sayan Mitra. Simulation-driven reachability using matrix measures. *ACM Trans. on Embedded Computing Systems*, 17(1):21:1–28, 2018. Special Issue on Autonomous Battery-Free Sensing and Communication.

[28] Joseph K. Scott and Paul L. Barton. Bounds on the reachable sets of nonlinear control systems. *Automatica*, 49:93–100, 2013.

[29] Olivier Bournez, Daniel S. Graća, and Amaury Pouly. Solving analytic differential equations in polynomial time over unbounded domains. In *MFCS'11: Proc. 36th Int'l Conf. Math. Foundations of Computer Sci.*, Berlin, Heidelberg, 2011. Springer-Verlag.

[30] Gustaf Söderlind. The logarithmic norm. history and modern theory. *BIT Numerical Mathematics*, 46(3):631–652, 2006.

[31] Arnold Neumaier. Global, rigorous, and realistic bounds for the solution of dissipative differential equations. Part I: Theory. *Computing*, 52:315–336, 1994.

[32] Torsten Strom. On logarithmic norms. *SIAM Journal on Numerical Analysis*, 12(5):741–753, 1975.

[33] Kai Hormann, Lucas Kania, and Chee Yap. Novel range functions via taylor expansions and recursive lagrange interpolation with application to real root isolation. In *Int'l Symp. Symbolic and Alge. Comp.(46th ISSAC)*, pages 193–200, New York, 2021. ACM Press. July 18-23. St. Petersburg, Russia. Software download from https://cs.nyu.edu/exact/core_pages/svn-core.html.

[34] Kai Hormann, Chee Yap, and Yashi (Andrew) Zhang. Range functions of any convergence order and their amortized complexity analysis. In *Computer Algebra in Scientific Computing (25th CASC)*, volume 14139 of *LNCS*, pages 162–182. Springer, 2023. Aug 28 - Sep 1, University of Havana, Cuba. http://www.casc-conference.org/. Software download from https://cs.nyu.edu/exact/core_pages/svn-core.html.

[35] C. Desoer and H. Haneda. The measure of a matrix as a tool to analyze computer algorithms for circuit analysis. *IEEE Trans. on Circuit Theory*, 19(5):480–486, 1972.

[36] C. V. Pao. Logarithmic derivates of a square matrix. *Linear Algebra and its Appl.*, 6:159–164, 1973.

[37] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems*. Springer-Verlag, Berlin, second revised edition, 2008.

[38] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Introduction to numerical algebraic geometry. In Alicia Dickenstein and Ioannis Z. Emiris, editors, *Solving Polynomial Equations: Foundations, Algorithms, and Applications*. Springer, 2010.

[39] Daniel Wilczak and Piotr Zgliczynski. C^r-Lohner algorithm. *Schedae Informaticae*, 20:9–42, 2011. Is there a 2018 update version in ArXiv?

[40] R.E. Moore. Interval Analysis: Differential Equations. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1686–1689. Springer, 2nd edition, 2009.

[41] Core Library Homepage, since 1999. Software download, source, documentation and links: https://cs.nyu.edu/exact/core_pages/svn-core.html.
[42] Computer assisted proofs in dynamics group, 2025.
URL http://capd.ii.uj.edu.pl/.
[43] T. Kapela, M. Mrozek, D. Wilczak, and P. Zgliczyński. CAPD::DynSys: a flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Communications in Nonlinear Sci. and Numerical Simulation*, 101:105578, 2021.
[44] Jihun Yu, Chee Yap, Zilin Du, Sylvain Pion, and Herve Bronnimann. Core 2: A library for Exact Numeric Computation in Geometry and Algebra. In *3rd Proc. Int'l Congress on Mathematical Software (ICMS)*, pages 121–141, Heidelberg, 2010. Springer. Kobe, Japan. Sep 13-17, 2010. LNCS No. 6327.