Reachability analysis of non-linear planar autonomous systems

Hallstein A. Hansen[1,3], Gerardo Schneider[2,3], and Martin Steffen[3]

[1] Buskerud University College, Kongsberg, Norway
[2] Chalmers | University of Gothenburg, Sweden
[3] University of Oslo, Norway

Abstract. Many complex continuous systems are modeled as *non-linear autonomous systems,* i.e., by a set of differential equations with one independent variable. Exact *reachability,* i.e., whether a given configuration can be reached by starting from an initial configuration of the system, is undecidable in general, as one needs to know the solution of the system of equations under consideration.
In this paper we address the reachability problem of planar autonomous systems approximatively. We use an approximation technique which "hybridizes" the state space in the following way: the original system is partitioned into a finite set of polygonal regions where the dynamics on each region is approximated by constant differential inclusions. Besides proving soundness, completeness, and termination of our algorithm, we present an implementation, and its application into (classical) examples taken from the literature.

1 Introduction

Many complex continuous systems can be modeled as *non-linear autonomous systems,* i.e., as a set of differential equations over one independent variable (typically interpreted as the time). Such systems can be found in the fields of mechanics, electrical engineering, etc., with typical textbook examples such as the damped pendulum, and oscillations in an electrical circuit as captured by the van der Pol oscillator equation.

Reachability analysis addresses the question whether, starting from an initial state or configuration, a system can evolve into another given state, i.e., whether it can reach that state. In this paper we investigate how to automate the approximation of non-linear dynamics in order to answer reachability questions for non-linear planar autonomous systems. The technique is based on *hybridizing*[4] the state space: the original system is partitioned into a finite set of polygonal regions where the dynamics on each region is approximated by constant differential inclusions. The resulting abstraction is called a Generalized Polygonal Hybrid System (GSPDI for short), for which reachability has been proved decidable [19]; the tool GSPeeDI [11] is a reachability-checker for such systems.

[4] A *hybrid system* combines both discrete and continuous behaviour.

A well-known, major difficulty in the analysis of differential equations are *critical points* such as sinks and attractors, i.e., objects of the so-called phase portrait. In their absence, our refinement may iteratively be of arbitrary precision (at least in theory). In regions containing critical points, however, the approximating GSPDI may lose precision quite drastically. We introduce an error measure, and an algorithm which allows us to choose the bound on the error arbitrarily small to obtain a practical and useful refinement.

Working on an abstraction of the original system gives a semi-test algorithm: a negative answer to reachability on the approximated system is indeed negative in the real system, whereas a positive answer is inconclusive. Obviously, a recurrent 'yes' answer is not useful unless we can iteratively refine the approximation to arrive, in some cases, at a definite 'no' with our technique, or 'yes' by other techniques.

Our algorithm takes as input a non-linear planar autonomous system S, an initial configuration given as set X_0 of points on the plane, and a final configuration X_f. Our approach performs reachability analysis by abstraction refinement: (1) Obtain a first (coarse) GSPDI H from S; (2) Check whether X_f is reachable from X_0; (3) If not, the algorithm terminates with a negative answer. (4) Otherwise, the situation is inconclusive, so refine the partition to obtain a better approximation H' and repeat from (2). The algorithm terminates after the error measure has been reached.

We prove that the above algorithm terminates and that it is sound and complete. Soundness, as usual, means that the result of the analysis can be relied on, i.e., the resulting GSPDI is indeed an abstraction of the original autonomous system in that it includes all the original behavior. Completeness states that if some state is reachable in the original system, the analysis provides evidence of that. We cannot expect completeness in that strict form, as the obtained GSPDIs will always over-approximate the real system, no matter how much we iterate the refinement procedure sketched above. Each refinement step, which corresponds to a finer partition of the plane, results in a GSPDI representing a more precise over-approximation, and by completeness we mean that we can approximate the original behavior up-to a given margin of error. We have incorporated a proof-of-concept prototype of the theory into the tool GSPeeDI. The prototype uses readily available local optimization libraries, and while this does not ensure that the test results are conservative in all cases, they give a good indication of what a real implementation of the theory would provide. We furthermore show the feasibility of the approach on a number of classical examples taken from the literature, and compare our results to those of related work.

The rest of the paper is organized as follows. Section 2 introduces notation and some mathematical results needed in the subsequent text. Section 3 gives the approximation from the dynamics of an autonomous systems to that of a GSPDI, presents our reachability analysis, and proves soundness, completeness, and termination. We discuss the implementation in Section 4, related work in section 5, and conclude in Section 6.

2 Background

In this section we present notations and definitions needed in the rest of the paper. We assume familiarity with Euclidean geometry, in particular vector operations. In the following we assume that, unless stated otherwise, vectors are normalized, so that two vectors are equal iff their directions are equal. A *unit circle* is a circle with radius 1, and vector $\mathbf{x}$ specifies a point on a unit circle. Henceforth, $\mathbf{x}$ refers to a vector as well as to the corresponding point on the unit circle.

An *arc* $\angle_{\mathbf{a}}^{\mathbf{b}}$ is a portion of the circumference of a unit circle, bounded by its endpoints, $\mathbf{a}$ and $\mathbf{b}$, where $\mathbf{a}$ is assumed located clockwise of $\mathbf{b}$. On the unit cycle, the length of an arc, written $|\angle_{\mathbf{a}}^{\mathbf{b}}|$ is also the angle between $\mathbf{a}$ and $\mathbf{b}$, measured in the interval $[0, 2\pi)$. We write $\mathbf{x} \in \angle_{\mathbf{a}}^{\mathbf{b}}$, if vector $\mathbf{x}$ is located clockwise of $\mathbf{b}$ and counter-clockwise of $\mathbf{a}$. If both $\mathbf{x} \in \angle_{\mathbf{a}}^{\mathbf{b}}$ and $\mathbf{y} \in \angle_{\mathbf{a}}^{\mathbf{b}}$ then we say that $\angle_{\mathbf{x}}^{\mathbf{y}} \subseteq \angle_{\mathbf{a}}^{\mathbf{b}}$ (if $\mathbf{x}$ is located clockwise with respect to $\mathbf{y}$), and so forth.

Many physical systems are modeled by one or more differential equations. Often the behavior of the system describes the development over time, so that the independent variable represents the time t.

Definition 1 (Autonomous system). *A* non-linear planar, autonomous, system *of first-order ordinary differential equations (ODEs) [5] is a system of the form*

$$\frac{dx}{dt} = f(x, y) \tag{1}$$

$$\frac{dy}{dt} = g(x, y). \tag{2}$$

The functions f and g may be non-linear, but neither depend on the independent variable t.

The functions f and g from Equations (1) and (2) represent derivatives of x and y w.r.t. t. The length of that vector gives the rate of change at that point and thus the vector $(f(x, y), g(x, y))$ describes the system's momentary *dynamic* at state (x, y). An *equilibrium point* is a point where the dynamic is the null vector; a system will remain in an equilibrium point forever.

For reachability, it is relevant only whether, not when, some point is reached. Thus we can normalize the dynamic as follows:

Definition 2 (Normalization). *The* normalized dynamics *of an autonomous system S is given by the function $h : \mathbb{R}^2 \to \mathbb{R}^2$:*

$$h(x, y) = (f(x, y)/r(x, y), g(x, y)/r(x, y))$$

where

$$r(x, y) = (\sqrt{f(x, y)^2 + g(x, y)^2}) \ .$$

The function is undefined when both $f(x, y) = 0$ and $g(x, y) = 0$. If $p = (x, y)$ is the input to h, we write $\dot{p}$ for $h(x, y)$.

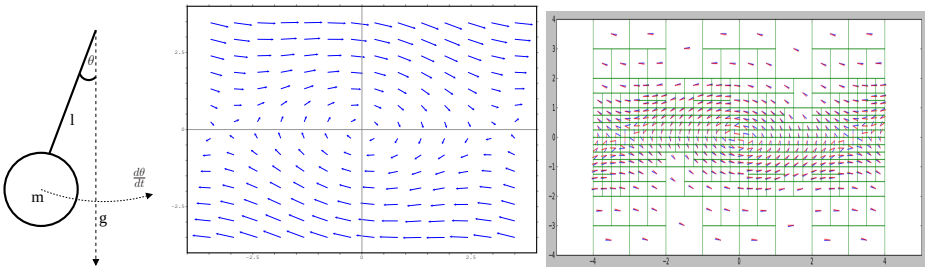

Fig. 1. On the left, the pendulum, with mass m, length l, gravitational acceleration g, angle θ, and angular velocity $\frac{d\theta}{dt}$. In the middle, the phase plane of the pendulum, with $m = 1, c = 2.5, l = 10, g = 10$. On the right, a corresponding GSPDI

Example 1 (Pendulum). The *damped pendulum* of Fig. 1 can be described by the second-order ODE

$$ml^2 \frac{d^2\theta^2}{dt^2} + cl\frac{d\theta}{dt} + mgl \sin\theta = 0$$

where constant c is the magnitude of the damping. Setting $x = \theta$ and $y = \frac{d\theta}{dt}$ allows us to transform the equation into the following autonomous system of two first-order ODEs:

$$\frac{dx}{dt} = y \ , \tag{3}$$

$$\frac{dy}{dt} = -\frac{c}{ml}y - \frac{g}{l}\sin x \ . \tag{4}$$

The associated phase plane is shown in Fig. 1 (middle), for the particular values $m = 1$, $l = 10$, $c = 2.5$, and $g = 10$. That is, $\frac{dx}{dt} = y$, and $\frac{dy}{dt} = -0.25y - \sin x$ and the picture illustrates the vector $\dot{p} = (y_i, -0.25y_i - \sin x_i)$ for several points (x_i, y_i). The equilibrium points are clearly visible, at $(0, 0)$ in the middle and furthermore $(-\pi, 0)$ and $(\pi, 0)$ to the left, respectively to the right. □

We will not consider reachability for autonomous systems directly, but rather by abstracting them into a special form of hybrid systems, known as *generalized polygonal hybrid systems* [4, 19]. The discretization is given by a finite partitioning of the plane into separate regions, and the behavior inside each region is governed by a differential inclusion. More specifically, the dynamics is given by two vectors restricting the direction of the system's behavior.

Definition 3 (GSPDI). *A* Generalized Polygonal Hybrid System *(GSPDI) is a pair* $H = \langle \mathbb{P}, \mathbb{F} \rangle$*, where* $\mathbb{P}$ *is a finite* partition *of the plane. Each* $P \in \mathbb{P}$*,*

called a region, *is a convex polygon with area P_A. The union $\bigcup \mathbb{P}$ of all regions is called the* domain *of the GSPDI and assumed to be a convex polygon of finite area itself. $\mathbb{F}$ is a function associating a pair of* vectors *to each region, i.e., $\mathbb{F}(P) = (\mathbf{a}_P, \mathbf{b}_P)$. Every point on the plane has its dynamics defined according to which polygon it belongs to: if $p \in P$, then $\dot{p} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$. In the following we assume all polygons are convex.*

A trajectory is a "path" through the state space, given as a function on the independent variable, which is often interpreted as time. In case of an autonomous system, possible trajectories are given by the differential equations; for the hybrid representation of GSPDIs, trajectories are determined by their direction of movement, in particular the tangent vector at any point should stay within the bounding angles (per region).

Definition 4 (Trajectory). *Let $I = [0, t]$ be a sub-interval of $\mathbb{R}_{\geq 0}$ (possibly identical to $\mathbb{R}_{\geq 0}$).*

1. *A* trajectory *ξ of an autonomous system S, written $\xi \in S$, is an almost-everywhere differentiable function $\xi : I \to \mathbb{R}^2$ which solves S for a given initial condition $\xi(t_0) = p$.*
2. *A* trajectory *of a GSPDI H, written $\xi \in H$, is an almost-everywhere differentiable function $\xi : I \to \mathbb{R}^2$ s.t. the following holds: whenever $\xi(t) \in P$ for some $P \in \mathbb{P}$, then its derivative $\dot{\xi}(t) \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$.*

We now relate autonomous systems and GSPDIs through an *approximation* relation.

Definition 5 (Approximation). *A GSPDI H* approximates *an autonomous system S (written $H \geq S$) if $\xi \in S$ implies $\xi \in H$.*

Example 2 (Pendulum). Reconsider the damped pendulum from Example 1 given in Equations (3) and (4). An approximating GSPDI of the pendulum is shown in Fig. 1 (right). □

To abstract an autonomous system successfully into a GSPDI, it is crucial to expect a certain "smoothness" of the behavior. This is formulated as a continuity condition, stipulating that if two points p and q are located close to each other, then their dynamics, $\dot{p}$ and $\dot{q}$, do not differ too much.

Definition 6 (Lipschitz continuity.). *A function f is* Lipschitz continuous *(or just Lipschitz, for short) on a polygon P if, for all points $p, q \in P$, there exists a constant K such that $\frac{||\dot{p}-\dot{q}||}{||p-q||} \leq K$. The smallest such K is called the* Lipschitz constant *of the function f on P. The maximum distance $||p - q||$ between any two points p and q in P, the* diameter *of P, is denoted $diam(P)$.*

In the following we assume that the normalized function h describing the dynamics of the system (cf. Definition 2) is Lipschitz continuous on all subsets of $\mathbb{R}^2$ except for arbitrarily small neighborhoods around a finite number of points. Under this assumption, the partition $\mathbb{P}$ of the plane falls into two separate groups of regions, those which are Lipschitz and those which are not, i.e., $\mathbb{P} = \mathbb{P}_L \cup \mathbb{P}_N$.

3 Refinement algorithm

This section presents the algorithm that over-approximates a given autonomous system by a GSPDI.

According to Definition 5, a GSPDI approximates the underlying autonomous system if its trajectories form a superset of the trajectories of the underlying autonomous system. The following lemma spells out a straightforward condition that tells us when that approximation holds.

Lemma 1 (Approximation). *Let S be an autonomous system with domain restricted to $\bigcup \mathbb{P}$, and H a GSPDI. If for all trajectories $\xi \in S$ and all points $\xi(t)$ on those trajectories, it is the case that $\xi(t) \in P$ and $\dot{\xi}(t) \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ (for some $P \in \mathbb{P}$), then $H \geq S$.*

Proof. The lemma follows directly from the Definitions 4 and 5. □

Unavoidably, by going from the autonomous system to the GSPDI, we lose precision. To determine how good the approximation is we measure the precision of the approximating GSPDI by considering the angles that bound the trajectories. More precisely, we use the *maximal* angle of all the regions of the GSPDI. Clearly, the smaller that angle, the better the approximation. We use those angles to *order* GSPDIs and write $H' \leq H$ ("H' refines H" or "H over-approximates H'") for the corresponding order. With regions being convex, an angle of π or larger does not restrict trajectories at all inside a region. Thus π is the maximal angle to consider. Definition 8 formalizes the corresponding *strict* refinement relation $H' < H$, which treats *non-Lipschitz* regions specially: In a non-Lipschitz region, e.g., containing an equilibrium point, one cannot reduce the bounding angle. The only way to strictly refine the system is to partition the region into smaller regions.

We define two numerical parameters to measure the precision of a GSPDI, one using the maximal angle that bounds the behavior in a set X of regions, which will in general be the Lipschitz regions, $\mathbb{P}_L$, and the second one to measure the relative "weight" of the remaining regions Y, in general all the non-Lipschitz regions of the system $\mathbb{P}_N$, compared to the overall domain. In what follows P_A will denote the area of a region P.

Definition 7 (Measures for precision). *Assume an autonomous system S and a GSPDI $H = \langle \mathbb{P}, \mathbb{F} \rangle$, $H \geq S$, and two disjoint sets X, Y such that $\mathbb{P} = X \cup Y$.*

1. *$\theta(X)$ is the maximum angle $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}|$ of all $P \in X$.*
2. *$\delta(Y)$ is the relative weight of the regions of Y, $\frac{\sum_{P \in Y} P_A}{(\cup \mathbb{P})_A}$.*

We can order GSPDIs by how precise they model the system dynamics. A GSPDI *refines* another if its partition is more fine-grained and, in particular, the bounding angles get smaller. For the same reason as in Definition 7, the latter condition applies for Lipschitz regions, only. In abuse of notation, we use $\leq$ to denote the corresponding refinement relation:

Definition 8 (Refinement). *Given two GSPDIs $H = \langle \mathbb{P}, \mathbb{F} \rangle$ and $H' = \langle \mathbb{P}', \mathbb{F}' \rangle$, H'* refines *H* properly*, written $H' < H$, if $\mathbb{P}'$ is a sub-partition of $\mathbb{P}$, and furthermore $|\angle^{\mathbf{a}_{P'}}_{\mathbf{b}_{P'}}| < |\angle^{\mathbf{b}_P}_{\mathbf{a}_P}|$, where P and P' with $P' \subseteq P$ are Lipschitz regions for H, resp. of H', i.e., $P \in \mathbb{P}_L$ and $P' \in \mathbb{P}'_L$.*

The following lemma states that we can choose our approximating GSPDIs as precise as we want them.

Lemma 2 (Bounds). *Given an autonomous system S, an angle θ with $0 < \theta \leq \pi$, and a number $\delta > 0$. Then there exists an approximating GSPDI H such that 1) $\theta(\mathbb{P}_L) \leq \theta$, and 2) $\delta(\mathbb{P}_N) \leq \delta$.*

Proof. The lemma imposes two conditions on the precision of H. 1) For the first one, Definition 6 of Lipschitz continuity gives $||\dot{p} - \dot{q}|| \leq K||p - q||$ for some K, for all points $p, q \in P$, and where K is the Lipschitz constant for P. Thus, $||\dot{p} - \dot{q}|| \leq K * diam(P)$. Since there is a one-to-one correspondence between the distance $||\dot{p} - \dot{q}||$ and the angle $|\angle^{\dot{q}}_{\dot{p}}|$, we can always partition Q such that all $P \in \mathbb{P}$ have a small enough $diam(P)$ such that $||\dot{p} - \dot{q}|| \leq K * diam(P)$ implies $|\angle^{\dot{q}}_{\dot{p}}| \leq \theta$.

2) The second condition is a direct consequence of the earlier assumption that h is Lipschitz on all subsets of $\mathbb{R}^2$ except for arbitrarily small neighborhoods around a finite number of (isolated) points: we can partition Q such that each region P from $\mathbb{P}_N$, the non-Lipschitz regions, contains exactly one such point and is arbitrarily small, which in turn renders the ratio arbitrarily small. □

Lemma 2 guarantees that there is always a GSPDI with $\theta(X)$ and $\delta(Y)$ arbitrarily small, for sets X, Y, trivially by letting $\mathbb{P}_L = X$ and $\mathbb{P}_N = Y$. To actually arrive at such a GSPDI, one can iteratively partition the domain finer and finer. For that purpose, we assume a function `partition`, which when applied to a partition of Q produces a sub-partition, for instance by splitting one particular polygon of the current partition. That, of course, leaves open which particular polygon or polygons are split, i.e., iterating the function `partition` is *non-deterministic*. It should be intuitively clear, that certain strategies for resolving the non-determinism will not improve the quality of the GSPDI, for instance by splitting only one half of the domain, but not improving on the other half, leaving the overall precision unchanged. The next lemma states, however, that there *exist* strategies of applying `partition` "smarter" than the one just mentioned, which eventually lead to partitions such that the corresponding GSPDI is below any predefined measure of precision.

Lemma 3. *Assume an autonomous system S, a polygon Q, an angle θ with $0 < \theta \leq \pi$, and a number $\delta > 0$. Then there exists a strategy to successively apply the `partition` function on Q that in a finite number of steps generates a partition $\mathbb{P}$ such that there exists a GSPDI $H = \langle \mathbb{P}, \mathbb{F} \rangle$ with Q as its domain, and where $\theta(\mathbb{P}_L) \leq \theta$ and $\delta(\mathbb{P}_N) \leq \delta$, such that $H \geq S$.*

Proof. The lemma requires application of `partition` iteratively such that $\theta(\mathbb{P}_L)$ and $\delta(\mathbb{P}_N)$ get smaller than the given upper bounds. This can be guaranteed, if

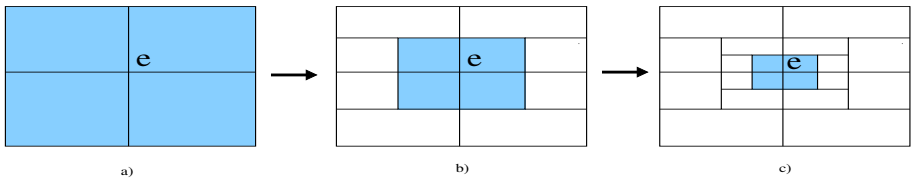

Fig. 2. Using `partition` on a rectangular initial polygon which contains an equilibrium point e. $\mathbb{P}_N$ is colored, $\mathbb{P}_L$ is white.

the strategy assures that all partitions of the domain of H get arbitrarily small (by Lemma 2). This can be achieved by splitting the polygons "uniformely", for instance, by always splitting (one of the) the largest into halves. □

In order to illustrate how one would realize `partition` we present an example. Here partitioning is done by simply splitting rectangles into two, along the rectangle's longest side. In particular the example shows that the number of non-Lipschitz regions remain constant under the chosen partitioning strategy.

Example 3. Consider an initial rectangle with an equilibrium point e at the exact center, see Figure 2. By partitioning twice we get four (colored) regions where the Lipschitz condition does not hold as they all contain e, Figure 2-a). Continuing to partition colored regions we can get a situation like in Figure 2-b), and later like in Figure 2-c). □

Applying the `partition` function as in (the proof of) the lemma above gives an algorithm which takes as input an autonomous system S, an initial polygon Q of finite area as domain of the intended GSPDI, and two bounds Θ and Δ as input. The iteration yields as output a partition $\mathbb{P}$ which forms part of a GSPDI $H = \langle \mathbb{P}; \mathbb{F} \rangle$ with $H \geq S$ and where furthermore $\mathbb{P}$ can be divided into two sets, $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$, such that $\theta(\mathbb{P}_{OK}) \leq \Theta$ and $\delta(\mathbb{P}_{BAD}) \leq \Delta$ (cf. Algorithm 1).

To maintain the successively finer partitioning of the given domain Q, the algorithm uses two collections of regions $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$. As loop invariant of the central iteration, the union of $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$ is a partition of the initial polygon Q. The collection $\mathbb{P}_{OK}$ contains regions P where $|\angle^{\mathbf{b}_P}_{\mathbf{a}_P}|$ is less than or equal to Θ. The collection $\mathbb{P}_{BAD}$, on the other hand, contains those regions whose angles are yet to be computed.

The collection $\mathbb{P}_{BAD}$ keeps the regions in a queue, which entails a form of "breadth-first" strategy: during each iteration, the first region P is removed from the head of the queue. If the corresponding bounding angle is small enough, i.e., if $|\angle^{\mathbf{b}_P}_{\mathbf{a}_P}| \leq \Theta$, then P is considered finished and moved to $\mathbb{P}_{OK}$. Otherwise, P is partitioned, and the subpolygons $P_1, \ldots, P_n$ are placed at the back of the queue $\mathbb{P}_{BAD}$. The while loop is executed until the area of $\mathbb{P}_{BAD}$ is less than or equal to the desired threshold, $\Delta * Q_A$. The return value is the union of $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$, which is a valid partition of Q, satisfying both Θ and Δ.

Note that the algorithm does not compute sets of polygons where underlying autonomous system is Lipschitz or not. Instead, these properties are implicitly

Algorithm 1 Construct a GSPDI on polygon Q on the plane, with precision parameters Δ and Θ.

Input: Convex polygon Q, $\Delta \in (0, 1]$, $\Theta \in (0, \pi]$

Empty queue $\mathbb{P}_{BAD}$, and empty collection $\mathbb{P}_{OK}$

$\mathbb{P}_{BAD}$.insert(Q)
while $(\mathbb{P}_{BAD})_A > \Delta * Q_A$ **do**
  P := $\mathbb{P}_{BAD}$.remove()
  if $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| \leq \Theta$ **then**
    $\mathbb{P}_{OK}$.insert(P)
  else
    $\{P_1, \ldots, P_n\}$:= partition(P)
    $\mathbb{P}_{BAD}$.insert($P_1, \ldots, P_n$)
  end if
end while
return $\mathbb{P}_{OK} \cup \mathbb{P}_{BAD}$

used to allow the computation of two sets $\mathbb{P}_{OK}$ and $\mathbb{P}_{BAD}$ where $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| \leq \Theta$ for all $P \in \mathbb{P}_{OK}$ and where the area of $\bigcup \mathbb{P}_{BAD} \leq \Delta * Q_A$ (cf. also Definition 7 which gives the measures of precision).

One of the precision measures used in the iteration is the angle which bounds the dynamics of the system, per partition: For the termination condition of the refinement process, we rely that for a given polygon P, the minimal bound can be calculated, i.e., the smallest arc $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ such that $\dot{p} \in \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ for all $p \in P$. In the implementation, we use external, numerical routines to implement a corresponding function `getArc` that calculates the value of $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$ (cf. Section 4 later about the implementation).

By the properties of $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$, i.e., with help of `getArc`, Algorithm 1 ensures that all the trajectories of the autonomous system are also trajectories of the generated approximating GSPDI (Lemma 1), that is the algorithm is sound It also satisfies that $\theta(H) \leq \Theta$ and $\delta(H) \leq \Delta$ (Lemma 3), which guarantees completeness, and also termination of the algorithm.

Theorem 1. *Algorithm 1 is sound, complete, and it terminates.*

Proof. The *soundness* of the algorithm is a direct consequence of the approximation Lemma 1: As an invariant, the domain Q is partitioned into regions $\mathbb{P}$ (split into $\mathbb{P}_{BAD}$ and $\mathbb{P}_{OK}$). Initially, the partition consists of one polygon, Q, and the loop either keeps the partition or refines it by replacing one polygon by sub-polygons. Each iteration/partition corresponds to a GSPDI, which approximates the autonomous system by Lemma 1.

As for *completeness:* the algorithm works by successively partitioning the polygons of $\mathbb{P}_{BAD}$. For each P considered, there are two options: Either $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| \leq \Theta$, in which case it is moved from $\mathbb{P}_{BAD}$ to $\mathbb{P}_{OK}$, or not.

The question is whether the area of $\mathbb{P}_{BAD}$ eventually will be less that $\Delta * Q_A$. By Lemma 3 and its proof we know that our strategy for applying `partition`

will generate two sets $\mathbb{P}_L$ and $\mathbb{P}_N$, the area of the latter which can be made arbitrarily small, and that we can find an arbitrarily small upper bound on the angle $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}|$ for each $P \in \mathbb{P}_L$. So we let Θ be an upper bound of these $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}|$, eventually forcing $\mathbb{P}_{BAD} \subseteq \mathbb{P}_N$. By having the upper bound of $(\cup\mathbb{P}_N)_A$ as $\Delta * Q_A$, we have that $\theta(\mathbb{P}_{OK}) \leq \Theta$ and $\delta(\mathbb{P}_{BAD}) \leq \delta(\mathbb{P}_N) \leq \Delta$.

Finally, the algorithm *terminates* when the area of $\mathbb{P}_{BAD}$ is less than $\Delta * Q_A$. The proof of completeness shows that this is always possible to achieve. In addition, Lemma 3 guarantees that there exists as strategy that generates a $\mathbb{P}_N$ with a sufficiently small area in a finite number of steps. Such a strategy is used in the implementation. □

4 Prototype implementation

The tool GSPeeDI contains an implementation of the results introduced in the previous section [10]. The tool answers 'maybe' or 'no' when asked to investigate safety properties. Graphics are also produced, and all the figures of GSPDIs in this paper are screen-shots from the tool. An overview of an older version of the tool has been published in [11].

A key issue was the implementation of the oracle `getArc`, which should return the extremal vectors $\mathbf{a}$ and $\mathbf{b}$ on polygon P, to create the arc $\angle_{\mathbf{a}_P}^{\mathbf{b}_P}$. We extracted the angle of a vector by using the function `atan2`, which is commonly implemented in many programming languages. It gives the angle of a vector with respect to the vector $(1, 0)$ in the interval $(-\pi, \pi]$. Since there is a discontinuity at the point $(-1, 0)$ we also used the function `atan2b` which gives the same angle, though in the interval $[0, 2\pi)$. Extremal vectors were thus obtained by maximizing and minimizing `atan2`, alternatively `atan2b`.

Note that due to the experimental nature of the prototype it does not strictly enforce the conservativeness of the theory presented in the previous sections. We used external, numerical routines for finding the extremal values of these two functions from the extensive Python scientific library Scipy [1]. This library includes a implementation of the limited memory Broyden-Fletcher-Goldfarb-Shanno method with bounds (L-BFGS-B) for non-linear optimization [24]. The bounds in question are box-constraints, which restrict us to rectangular regions.

The empirical results appear correct, as illustrated in figures 3 and 4, but an implementation that guarantees conservative answers should include global optimization methods [23].

A very real scenario when using optimization tools of any kind is that they may fail, depending on starting points, constraints, or the function to be optimized. The ratio of such failures, and their consequences, obviously determines the usability of the tool.

In the event of failure, we also implemented a backup routine that produced arcs that preserved the soundness of Algorithm 1. If that also fails we ultimately give up and declare the offending region to be reach-all.[5] We included a cut-off

[5] A *reach-all* region is one where every point is reachable from any other point.

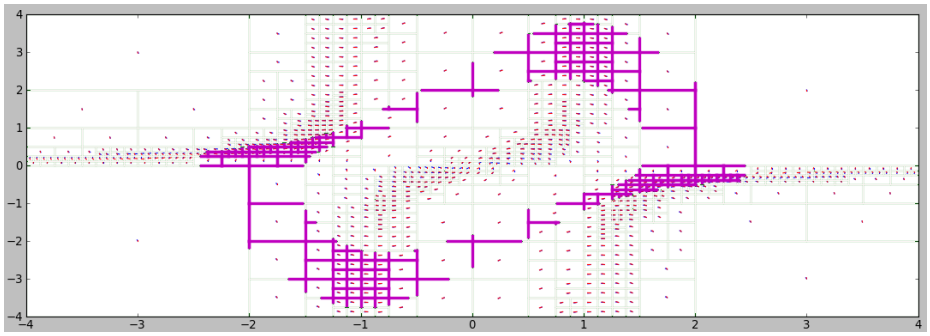

Fig. 3. Reach-set for van der Pol equation with $\Theta = 0.45$ and $\Delta = 5\%$. The parameter $\mu = 1.5$.

parameter to the algorithm implementation to ensure termination, should such failures should prove abundant.

The implementation was restricted to produce only rectangular regions, and the `partition` function is realized as simply splitting a rectangle P with length l and width w, $l \geq w$, into two rectangles with length $l/2$ and width w.

We are interested in assessing the performance of the prototype when applied to real non-linear autonomous systems, as means to decide whether to write a full, conservative implementation of the theory, and so we have performed some case studies.

4.1 Case studies

We present results produced by our prototype when used on models of the damped pendulum (cf. Example 1) and the van der Pol oscillator.

Example 4. The equation of the van der Pol oscillator [6], are used in electrical engineering, neurology, and seismology. The second-order ODE

$$x''(t) = -\mu(x(t) - 1)x'(t) - x(t)$$

can be transformed into a first-order non-linear autonomous system:

$$x'(t) = y(t)$$
$$y'(t) = -\mu(x(t) - 1)y(t) - x(t).$$

The above equation, where the positive constant μ represents the amount of damping in the system, is interesting because it includes a limit cycle: All trajectories in the system converge towards that cycle.

We executed our tool on a laptop with a 1.33 Ghz Intel Atom processor, generating GSPDIs with different values of Θ and Δ, with initial area $Q =$

System	Θ	Δ	Refinement	Graph building	Reach set	Remark
Pendulum	0.125	20%	403s	90s	62s	
Pendulum	0.2	2.5%	298s	69s	41s	Fig. 4
Pendulum	0.5	5%	10s	12s	6s	Fig. 4
Pendulum	0.5	0.1%	33s	28s	58s	
Van der Pol	0.45	5%	36s	32s	485s	Fig. 3
Van der Pol	0.75	5%	12s	17s	2s	

Table 1. Results obtained by running GSPeeDI on the damped pendulum and the van der Pol oscillator with different precision parameters.

$[-4, 4] \times [-4, 4]$. The results are shown in table 1. Also, some results are shown graphically in figures 3 and 4. The desired Δ was attained in all cases. Not shown in the table are the results we got concerning the failure ratio of the `getArc` function: For the van der Pol system it failed only on the initial rectangle Q, and for the pendulum only on Q and the two rectangles Q was split into, independently of Δ or Θ. We did not observe any failures of the backup routine.

5 Related work

In this section we briefly survey related work, both with respect to our theoretical development as well as to our implementation.

5.1 Refinement

The idea of over-approximating systems having complex, often non-linear, dynamics by systems with simpler dynamics in order to investigate safety properties is not novel, and neither is the technique of creating finer and finer partitions to verify safety properties [14]. Our approach aims at a *fully automated* process to answer the reachability question for any non-pathological planar autonomous system by working on adjusting the precision up to a desirable level.

Defining an upper limit on the approximation error (the Θ parameter in our approach) is quite standard and used in many other approaches. However, for non-linear dynamics using only this upper limit is not enough as we cannot guarantee that $|\angle_{\mathbf{a}_P}^{\mathbf{b}_P}| \leq \Theta$ for non-Lipschitz regions P. The Δ parameter is used to put an upper limit to the area of these regions.

We presently consider the autonomous systems as 'black boxes' that are fed to the optimization software, while other related works (e.g., [16, 14, 17]) require manual analysis to find good partitions. Automatic partitioning has been implemented in [8], but not for systems with non-linear dynamics.

5.2 Approximation

As mentioned above the purpose of refinement is to replace complex, possibly non-linear, dynamics with simpler yet less precise dynamics. This can be done

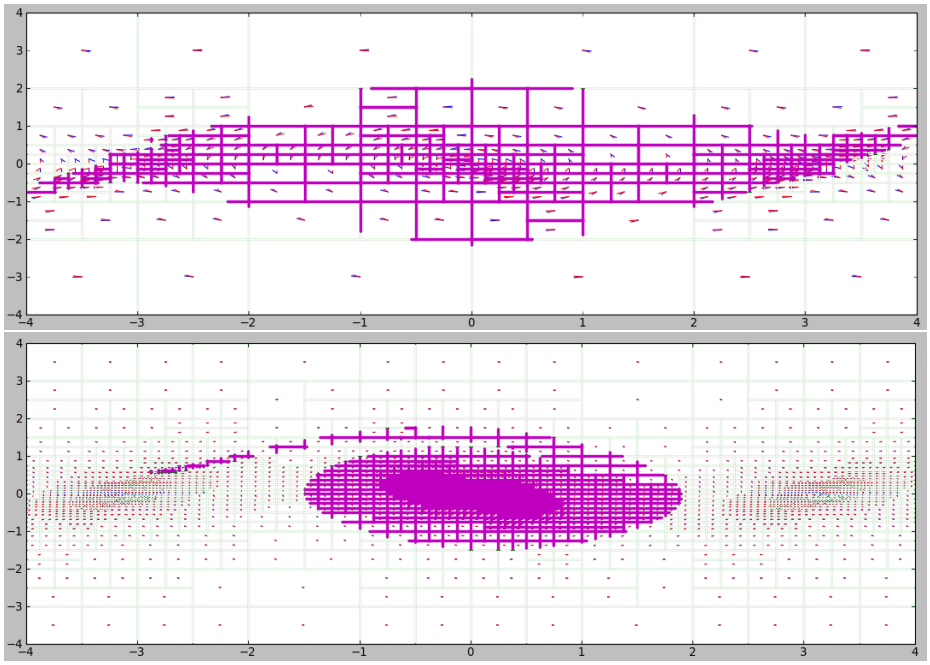

Fig. 4. Pendulum example: Above a coarse GSPDI and reach-set with parameters $\Theta = 0.5$ and $\Delta = 5\%$, and below a finer GSPDI with parameters $\Theta = 0.2$ and $\Delta = 2.5\%$.

by techniques such as rate translation [14], the result of which is a rectangular hybrid automata [8], or linear phase-portrait approximation, generating linear hybrid automata [15]. In both cases the method and resulting approximation are motivated by what automata are accepted by their tools, Hytech [12], and PHAVer [9], respectively. This is also true for our work: we produce approximations in the form of constant differential inclusions (GSPDIs), which can then be analyzed using our tool GSPeeDI [11]. Our method, realized through the `getArc` function, is optimal for any given region provided the external routines succeed in finding the extremal vectors of the function h.

5.3 Comparison with other tools

Known tools that do not directly analyze non-linear autonomous systems (automatically) are HyTech [12], based on linear hybrid automata [13], PHAVer [9], that approximates piece-wise affine dynamics into polyhedral automata, and d/dt [3], which is based on linear differential inclusions. A comparable tool is HSolver [21], which can analyze systems with non-linear dynamics (based on interval constraint propagation). HSolver is based on RSolver, a program for

solving quantified inequality constraints [20], which guarantees conservative results.

We have formulated and run HSolver on the examples mentioned earlier, the damped pendulum, and the van der Pol equation. While our prototype shows promising results in terms of execution time, we will postpone any discussion of this until we have a conservative implementation.

6 Conclusion

In this paper we presented an approach for reachability for non-linear planar autonomous systems by hybridizing the system into a GSPDI using abstraction refinement.

Exploiting Lipschitz continuity for reachability checking and simulation is not new in itself. It is for instance inherent in the *hybridization* approach of [2], and is also used for hybrid computation [7]. A main difference is that we consider systems that may be Lipschitz continuous only in parts of the plane. A Lipschitz continuous system has an upper bound, the Lipschitz constant, on how fast the system's dynamics changes. We exploit the phenomenon that a system may be Lipschitz continuous almost everywhere, and have different Lipschitz constants for different areas of the plane. We minimize the area where the system is not Lipschitz continuous, and treat areas where the Lipschitz constant is large more thoroughly than areas where it is small, to get as good an approximation as possible. This comes with a computational price, as we must identify the Lipschitz constant for each area we consider, using non-linear optimization tools. We need, however, to identify these areas only once, and then we can perform multiple reachability computations based without needing to perform the task again.

Approximation of complex, possibly non-linear, dynamics by simpler, yet less precise, dynamics is a well-studied field [14, 16, 8]. Our work demonstrates a strategy by which, using optimizations tools, we can automate the approximation of non-linear dynamics, using an optimal approximation for GSPDIs.

We are currently working on a conservative implementation based on the current prototype. In the future, also we intend to expand the class of systems that can be analyzed by GSPDIs. Our approach may be used to approximate differential inclusions and switched continuous systems [22], as well. Applying the techniques on more general systems with an arbitrary numbers of variables, different modes, jumps, etc., is also an interesting topic, despite that this will cause loss of decidability in the approximated system. In addition, since we now are able to create GSPDIs with a large number of regions, we will work in improving the tool's performance and furthermore incorporate the theoretical improvements investigated in [18].

References

[1] The Scipy library. `http://www.scipy.org`.

[2] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In *HSCC'03*, volume 2623 of *LNCS*, pages 20–35, 2003.
[3] E. Asarin, T. Dang, O. Maler, and O. Bournez. Approximate reachability analysis of piecewise-linear dynamical systems. In *HSCC'00*, pages 20–31, 2000.
[4] E. Asarin, G. Schneider, and S. Yovine. Algorithmic analysis of polygonal hybrid systems, part I: Reachability. *TCS*, 379(1-2):231–265, 2007.
[5] W. Boyce and R. DiPrima. *Elementary differential equations and boundary value problems*. Wiley New York, 8th edition, 2004.
[6] B. V. der Pol and J. V. der Mark. Frequency Demultiplication. *Nature*, 120, 1927.
[7] J. D. Dora, A. Maignan, M. Mirica-Ruse, and S. Yovine. Hybrid computation. In *ISSAC*, pages 101–108, 2001.
[8] L. Doyen, T. A. Henzinger, and J.-F. Raskin. Automatic rectangular refinement of affine hybrid systems. In *FORMATS*, pages 144–161, 2005.
[9] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC'05*, volume 3414 of *LNCS*, pages 258–273, 2005.
[10] H. A. Hansen. GSPeeDI. `http://heim.ifi.uio.no/hallstah/gspeedi/`.
[11] H. A. Hansen and G. Schneider. GSPeeDI –A Tool for Analyzing Generalized Polygonal Hybrid Systems. In *ICTAC'09*, volume 5684 of *LNCS*, pages 336–342, August 2009.
[12] T. Henzinger, P.-H.Ho, and H.Wong-toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1), 1997.
[13] T. A. Henzinger. The theory of hybrid automata. In *LICS'96*, pages 278–292. IEEE Computer Society, 1996.
[14] T. A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In *CAV*, pages 225–238, 1995.
[15] T. A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In *Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control*, pages 377–388, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.
[16] P.-H. Ho. *Automatic analysis of hybrid systems*. PhD thesis, Ithaca, NY, USA, 1995.
[17] P.-H. Ho and H. Wong-toi. Automated analysis of an audio control protocol. In *In Proc. of CAV'95*, pages 381–394. Springer-Verlag, 1995.
[18] G. Pace and G. Schneider. A compositional algorithm for parallel model checking of polygonal hybrid systems. In *ICTAC'06*, volume 4281 of *LNCS*, pages 168–182, 2006.
[19] G. J. Pace and G. Schneider. Relaxing goodness is still good. In *ICTAC'08*, volume 5160 of *LNCS*, pages 274–289, 2008.
[20] S. Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Transactions on Computational Logic*, 7(4):723–748, 2006.
[21] S. Ratschan and Z. She. Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement. *ACM Transactions in Embedded Computing Systems*, 6(1):573–589, 2007.
[22] O. Stursberg and S. Kowalewski. Approximating switched continuous systems by rectangular automata. In *European Control Conference*, 1999.
[23] T. Weise. *Global Optimization Algorithms Theory and Application* . E-book, 2nd edition, 2009. `http://www.it-weise.de/`.
[24] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 78: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, 1997.