Providing a Basin of Attraction to a Target Region by Computation of Lyapunov-like Functions

Stefan Ratschan
Institute of Computer Science,
Czech Academy of Sciences,
Prague, Czech Republic
stefan.ratschan@cs.cas.cz

Zhikun She
Max-Planck-Institut für Informatik,
Saarbrücken, Germany
zhikun@mpi-sb.mpg.de

***Abstract*— In this paper, we present a method for computing a basin of attraction to a target region for non-linear ordinary differential equations. This basin of attraction is ensured by a Lyapunov-like polynomial function that we compute using an interval based branch-and-relax algorithm. This algorithm relaxes the necessary conditions on the coefficients of the Lyapunov-like function to a system of linear interval inequalities that can then be solved exactly, and iteratively reduces the relaxation error by recursively decomposing the state space into hyper-rectangles. Tests on an implementation are promising.**

I. Introduction

A sufficient condition for verifying stability of ordinary differential equations is the existence of a Lyapunov function [6]. In cases where the differential equation is polynomial, due to decidability of the theory of real-closed fields [24], one can always check, whether for a given polynomial with parametric coefficients, there are instantiations of these parameters resulting in a Lyapunov function. However, all the existing decision procedures are not efficient enough to be able to solve this problem in practice.

Hence one has to fall back to algorithms based on approximation. Usually, one can compute a Lyapunov function of a linearization of the non-linear problem around a given equilibrium point, and compute a basin of attraction for this Lyapunov function wrt. the original, non-linear problem [5]. However, due the information loss introduced by the linearization process, this basin of attraction will usually be very small.

In this paper we will provide an algorithm for computing a Lyapunov-like function for the original, non-linear problem, which will provide us with a basin of attraction to a given target region. This target region can, for example, be the smaller basin of a attraction obtained by linearization, hence ensuring attraction to the equilibrium. The method also seems fit to deal with uncertain systems—a corresponding extension of the algorithms is a simple home-work exercise.

The approach sets up a polynomial with parametric coefficients, substitutes this polynomial into a constraint that formalizes Lyapunov style conditions, and then solves this constraint for the parameters that form the coefficients of the polynomial.

The algorithm for solving this constraint employs a branch-and-relax scheme. It relaxes the constraint to a system of linear interval inequalities that can then be solved exactly [20], and iteratively reduces the relaxation error by recursively decomposing the state space into hyper-rectangles.

We implemented our algorithm and tested our implementation on eight examples.

The structure of the paper is as follows: in Section II we show how to compute a basin of attraction using a relaxed version of Lyapunov functions; in Section III we describe our algorithm for computing such relaxed Lyapunov functions; in Section IV we discuss our implementation; in Section V eight examples are illustrated with computation results; in Section VI we involve some related work; and in Section VII we conclude our paper.

II. Basins of Attraction to a Target Region

Consider a nonlinear autonomous ordinary differential equation: $\dot{x} = f(x)$, where $x \in \mathbb{R}^n$ and $f(x)$ is a continuous and differentiable function. We denote by $x(\cdot, x_0) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ a trajectory of the differential equation starting from x_0.

Usually, using linearization, one can compute a small basin of attraction to a given equilibrium of $\dot{x} = f(x)$. In order to be able to arrive at a larger basin of attraction, we compute a Lyapunov-like function that ensures a basin of attraction to a target region containing the equilibrium. This can be formalized using some notion of practical stability [7]:

Definition 1: Given an n-dimensional differential equation $\dot{x} = f(x)$, and sets U and TR such that $TR \subset U \subseteq \mathbb{R}^n$, the differential equation is *stable* with respect to U and the target region TR if every trajectory starting in a point $x_0 \in U$ will

- always stay in U (for all $t \in \mathbb{R}_{\geq 0}$, $x(t, x_0) \in U$),
- and eventually reach TR (there is a $t_1 \in \mathbb{R}_{\geq 0}$ such that $x(t_1, x_0) \in TR$)

In order to ensure this stability notion, a relaxed form of Lyapunov functions suffices:

Definition 2: For a given differential equation $\dot{x} = f(x)$ with sets B and TR such that $TR \subset B$, a function $V(x)$ is called a *relaxed Lyapunov function* with respect to B and TR

This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS). See www.avacs.org for more information

if and only if

$$\forall x \in B \left[x \notin TR \Rightarrow \frac{d}{dt} V(x) < 0 \right]. \quad (1)$$

Given a closed set B and a Lyapunov function V, we denote by $BA_B(V)$ the set $\{x \in B : V(x) < \min_{x \in \partial B} V(x)\}$.

What exactly is guaranteed by a relaxed Lyapunov function? First of all, it ensures that the set $BA_B(V)$ is never left.

Theorem 1: The existence of a relaxed Lyapunov function $V(x)$ wrt. a closed set B and an open set TR guarantees that every trajectory starting in $BA_B(V)$ will always stay in this set , provided that the closure of TR is a strict subset of $BA_B(V)$.

Proof: Let $r = \min_{x \in \partial B} V(x)$. We want to prove that for all $x_0 \in BA_B(V)$, for all $t \in \mathbb{R}_{\geq 0}$, $x(t, x_0) \in BA_B(V)$. Let $x_0 \in BA_B(V)$, $t \in \mathbb{R}_{\geq 0}$ be arbitrary but fixed. We assume that $x(t, x_0) \notin BA_B(V)$ and try to derive a contradiction. Our assumption implies that either $x(t, x_0) \notin B$, or $x(t, x_0) \in B$ and $V(x(t, x_0)) \geq r$.

- If $x(t, x_0) \notin B$, according to the continuity of $x(\cdot, x_0)$, there exists a t_2 such that $x(t_2, x_0) \in \partial B$. Let $t_2' = \inf\{t \in [0, t_2] : x(t, x_0) \in \partial B\}$. Then, due to our choice of r, $V(x(t_2', x_0)) \geq r$. Moreover, due to our choice of t_2', for all $t' \in [0, t_2']$, $x(t', x_0) \in B$.
  Let $t_3 = 0$, if $x(\cdot, x_0)$ does not enter TR in $[0, t_2']$ and $\sup\{t \in [0, t_2'] : x(t, x_0) \in TR\}$, otherwise.
  Since $x(t_3, x_0) \in BA_B(V)$, $V(x(t_3, x_0)) < r$. Moreover, for all $t_4 \in [t_3, t_2']$, $x(t_4, x_0) \in B \setminus TR$. Hence, due to the continuity of $V(x(\cdot, x_0))$ and the mean value theorem there has to be a point $t_5 \in [t_3, t_2']$ such that $\dot{V}(x(t_5, x_0)) > 0$, which is a contradiction to the fact that V is a relaxed Lyapunov function wrt. B and TR.
- If $x(t, x_0) \in B$ and $V(x(t, x_0)) \geq r$, we can derive a contradiction in a similar way.

■

Moreover, a relaxed Lyapunov function ensures that a target region is eventually reached:

Theorem 2: The existence of a relaxed Lyapunov function $V(x)$ wrt. a closed set B and an open set TR guarantees that every trajectory starting in $BA_B(V)$ will enter TR, provided that the closure of TR is a strict subset of $BA_B(V)$.

Proof: Since B is bounded and $B \setminus TR$ is closed, due to the continuity of $\dot{V}$, we know that $\dot{V}$ has a maximum ϵ in $B \setminus TR$. Obviously, $\epsilon < 0$.

Let x_0 be an arbitrary, but fixed point in $BA_B(V)$. It is sufficient to only consider the case $x_0 \in BA_B(V) \setminus TR$.

We assume that for all $t \in \mathbb{R}_{\geq 0}$, $x(t, x_0) \notin TR$, and derive a contradiction. From Theorem 1, for all $t \in \mathbb{R}_{\geq 0}$, $x(t, x_0) \in BA_B(V) \setminus TR$. Thus, for all $t \in \mathbb{R}_{\geq 0}$, $\dot{V} \leq \epsilon$. This implies that as t goes to infinity, $V(x(t, x_0))$ goes to minus infinity, contradicting the fact that V is bounded in $B \setminus TR$.

Thus, there exists a $t \in \mathbb{R}_{\geq 0}$ such that $x(t, x_0) \in TR$. ■

Hence stability can be proven by computation of Lyapunov functions:

Corollary 1: The existence of a relaxed Lyapunov function $V(x)$ wrt. a closed set B and an open set TR guarantees stability wrt. $BA_B(V)$ and the target region TR.

In theory, as for classical Lyapunov functions, one could compute relaxed Lyapunov functions using decision procedures for the theory of real-closed fields [24]. However, the current methods are by far not efficient enough to solve this problem in practice. Another approach would be to use an interval arithmetic based branch-and-bound or branch-and-prune scheme [15], [17]. However, one can do even better, as will be shown in the next section.

III. Algorithm

In this section, we present a method for finding a polynomial relaxed Lyapunov function V, provided that f is a polynomial. Let V be a polynomial with parametric coefficients. Let $\frac{d}{dt}V$ be the derivative of V along f, represented as a polynomial whose coefficients are linear combinations of the parameters that form the coefficients of V. We substitute the resulting $\frac{d}{dt}V$ into the Constraint 1 and then solve the resulting constraint Λ_V for the parameters that form the coefficients of V.

For solving the constraint Λ_V we use the convention that an interval occurring in a constraint represents a fresh, universally quantified variable ranging over that interval (e.g., $[0, 1]a \leq 1$ represents $\forall v \in [0, 1]\ va \leq 1$, where v is a new variable). Also, we assume that V is of the form $\sum_{j=1}^{m} a_j x^{\alpha_j}$ (the powers occurring in the tuple α_j are applied element-wise to the tuple of variables x). We view the resulting $\frac{d}{dt}V$ as having the form $\sum_{i=1}^{l} g_i(a_1, \ldots, a_m) x^{\gamma_i}$, where the $a_1, \ldots, a_m$ occur linearly in the $g_i(a_1, \ldots, a_m)$. Then we proceed as follows:

1) drop the constraint on the left-hand side of the implication sign of Λ_V
2) replace every monomial x^{γ_i} by an interval bounding its range over B (the result is a constraint of the form $\sum_{i=1}^{l} g_i(a_1, \ldots, a_m) I_i < 0$, where the I_i's are intervals),
3) rewrite $\sum_{i=1}^{l} g_i(a_1, \ldots, a_m) I_i$ to an expression of the form $\sum_{j=1}^{m} I_j' a_j$ by distributing each interval I_i over the linear combination $g_i(a_1, \ldots, a_m)$, collecting the coefficients of each a_j, and computing a single interval for each such coefficient.

The interval calculations of Steps 2 and 3 can be done using interval arithmetic. The resulting constraint over-approximates the original constraint Λ_V (resulting in a smaller set of $a_1, \ldots, a_m$ on which it holds). Moreover it is a linear interval inequality, which can be solved exactly (to be shown later). However, each of the above steps introduces some over-approximation. We will now try to reduce this over-approximation.

We will start with the over-approximation introduced by Step 2. Even if the bounds on the monomials are exact, this step looses the dependency between the different monomials. For reducing this problem, we rewrite the universal quantifier $\forall x \in B$ of Λ_V to a conjunction of the form $\forall x \in B_1 \sqcup \wedge \forall x \in B_2 \sqcup$, where $B_1 \cup B_2 = B$, and B_1 and B_2 non-overlapping. We can apply the above process to every branch of the resulting constraint, arriving at a system of linear interval inequalities, which also can be solved exactly. We continue with this branching process until a solution can be found.

Algorithm 1 Computing a relaxed Lyapunov function

Input: a polynomial differential system $\dot{x} = f(x)$, sets B and TR

Output: if the algorithm terminates, a relaxed Lyapunov function with respect to B and TR

1: choose a polynomial V with parametric coefficients
2: let $\phi \leftarrow \bigwedge \Lambda_V$
3: **while** relax(ϕ) does not have a solution **do**
4: branch a universal quantifier in ϕ
5: **end while**
6: return V with the solution of $relax(\phi)$ substituted for $a_1, \ldots, a_m$

For reducing the over-approximation introduced by Step 1, we observe that the above branching process results in smaller bounds for the universal quantifiers. This allows us, for some branches of the form $\forall x \in B' \left[x \notin TR \Rightarrow \frac{d}{dt}V(x) < 0\right]$ to prove $\forall x \in B'[x \in TR]$, which also proves the full branch. Hence we can drop the branch from the conjunction.

For a constraint ϕ, we denote the result of the relaxation process, as described until now by $relax(\phi)$, and arrive at the branch-and-relax Algorithm 1.

For reducing the over-approximation introduced by Step 3, instead of distributing the intervals over the linear combinations $g_i(a_1, \ldots, a_m)$, we introduce for each linear combination $g_i(a_1, \ldots, a_m)$ a new variable a_i', and relate it to the variables $a_1, \ldots, a_m$ by adding the equation $a_i' = g_i(a_1, \ldots, a_m)$. In fact, we rewrite this equation to two inequalities $a_i' \leq g_i(a_1, \ldots, a_m)$ and $-a_i' \leq -g_i(a_1, \ldots, a_m)$ to again arrive at linear interval inequalities. Note that the new constraints do not contain the states space variables, hence they only have to be added once—and not for every branch.

For a constraint ϕ, we denote the result of the relaxation process that uses the construction of the previous paragraph in Step 3 by $relax^{=}(\phi)$. It can be used in Algorithm 1 instead of $relax(\phi)$. This improves the algorithm, since we have:

Property 1: $relax^{=}(\phi)$ is tighter than $relax(\phi)$

The system of linear interval inequalities formed by $relax(\phi)$ or $relax^{=}(\phi)$ can be solved as follows: First replace each strict inequality < 0 by a non-strict inequality $\leq -\varepsilon$, with ε positive but small. This introduces some over-approximation, but this over-approximation can easily be made arbitrarily small.

Now we can proceed using a method due to Rohn [20]: we can replace each variable a we want to solve for, by a difference of two positive variables $a_1 - a_2$, and replace expressions of the form $I(a_1 - a_2)$ by $Ia_1 - Ia_2$. Since a_1 and a_2 are positive, and I represents a universally quantified variables in an inequality, we can replace the interval $I = [I^-, I^+]$ by its bounds to arrive at $I^+a_1 - I^-a_2$. The result is a system of linear inequalities that can be solved by linear programming. According to a proof by Rohn this procedure does not introduce over-approximation [20].

IV. Implementation

We implemented the method within the framework of our constraint solver RSolver [16], [15] that allows solving of quantified constraints using a branch-and-prune algorithm. This solver has a branching loop of the same form as Algorithm 1, but instead of using the relaxation technique described in this paper, it deduces information from the input constraints using a generalization of interval arithmetic called interval constraint propagation.

We simply added our relaxation technique to the branching loop of RSolver. As a result, we have an algorithm that can in some cases infer more information from the input than Algorithm 1 due to its use of interval constraint propagation.

We use the following heuristics for branching: Choose the widest box, and bisect it into two boxes along the variable along which this variable has not been split for the longest time.

V. Examples

In this section, eight examples will be presented, for which we computed relaxed Lyapunov functions using the method described in this paper. Here the target region TR is the set $\{x \in B : |x_i - \bar{x}_i| < \delta, 1 \leq i \leq n\}$, where B is a given box containing the equilibrium $\bar{x}$ and $\delta > 0$ is a arbitrarily given constant. The choice of the constant ε used for rewriting the strict to non-strict inequalities depend on the degree of the chosen parametric polynomial of each example.

Example 1: An example of the simplified model of a chemical oscillator in [10]. The original system is:

$$\begin{cases} \dot{x}_1 = a - x_1 + x_1^2 x_2 \\ \dot{x}_2 = b - x_1^2 x_2 \end{cases}$$

Let $(a, b) = (0.5, 0.5)$, then the equilibrium is $(1, 0.5)$. Let $V(x_1, x_2) = ax_1^2 + bx_1 + cx_1x_2 + dx_2 + ex_2^2 + f$, then $\dot{V}(u, v) = 2ax_1^3x_2 + (c - 2e)x_1^2x_2^2 - cx_1^3x_2 + bx_1^2x_2 - dx_1x_2^2 - 2ax_1^2 - cx_1x_2 + (a - b + 0.5c)x_1 + (0.5c + e)x_2 + 0.5b + 0.5d$.

Choosing $B = [0.8, 1.2] \times [0.3, 0.7]$, $\delta = 0.01$ and $\varepsilon = 0.0001$, we get a relaxed Lyapunov function $V = x_1^2 - 38.5021289757x_1 - 62.5573802031x_2 + 17.3662782081x_2^2$.

Example 2: This is the well-known Van-der-pol equation:

$$\begin{cases} \dot{x}_1 = -x_2 \\ \dot{x}_2 = x_1 - (1 - x_1^2)x_2 \end{cases}$$

Let $V(x_1, x_2) = ax_1^2 + bx_1x_2 + cx_2^2$, then $\dot{V}(x_1, x_2) = (-2a - b + 2c)x_1x_2 + bx_1^2 + (-b - 2c)x_2^2 + bx_1^3x_2 + 2cx_1^2x_2^2$.

Choosing $B = [-0.8, 0.8] \times [-0.8, 0.8]$, $\delta = 0.1$ and $\varepsilon = 0.0001$, we get a relaxed Lyapunov function $V(x_1, x_2) = x_1^2 - 0.596022616494x_1x_2 + 0.701988691753x_2^2$.

Example 3: An example from [5]:

$$\begin{cases} \dot{x}_1 = -x_1 + x_2 \\ \dot{x}_2 = 0.1x_1 - 2x_2 - x_1^2 - 0.1x_1^3 \end{cases}$$

Let $V(x_1, x_2) = ax_1^2 + bx_2^2$, then $\dot{V}(x_1, x_2) = -2ax_1^2 + (2a + 0.2b)x_1x_2 - 4bx_2^2 - 2bx_1^2x_2 - 0.2bx_1^3x_2$.

If we choose $B = [-0.8, 0.8] \times [-0.8, 0.8]$, $\delta = 0.1$ and $\varepsilon = 0.0001$, we get a relaxed Lyapunov function $V(x_1, x_2) = x_1^2 + 2.51614247032x_2^2$.

Example 4: An example from a Chinese textbook on ODEs:

$$\begin{cases} \dot{x} = -4x^3 + 6x^2 - 2x \\ \dot{y} = -2y \end{cases}$$

Let $V(x, y) = ax^4 + bx^3 + cx^2 + dy^2$, then $\dot{V}(x, y) = -16ax^6 + (24a - 12b)x^5 + (-8a + 18b - 8c)x^4 + (-6b + 12c)x^3 - 4cx^2 - 4dy^2$.

If we choose $B = [-0.8, 0.8] \times [-0.8, 0.8]$, $\delta = 0.1$ and $\varepsilon = 0.000001$, we get a relaxed Lyapunov function $V(x, y) = x^4 + 1.21822257384x^3 + 0.60911128692x^2 + 0.0250081223629y^2$.

Example 5: An example from [11] whose Lyapunov function has been constructed using the sum of squares decomposition:

$$\begin{cases} \dot{x} = -x + (1 + x)y \\ \dot{y} = -(1 + x)x \end{cases}$$

Let $V(x, y) = ax^2 + bxy + cy^2 + dy^3 + ex^4 + fx^2y^2 + gy^4$, then $\dot{V}(x, y) = (-2a - b)x^2 + (2a - b - 2c)xy + by^2 - bx^3 + (2a - 2c)x^2y + (b - 3d)xy^2 + (-4e)x^4 + (4e - 2f)x^3y + (-3d - 2f)x^2y^2 + (2f - 4g)xy^3 + (4e - 2f)x^4y + (2f - 4g)x^2y^3$.

If we choose $B = [-0.7, 0.9] \times [-0.7, 0.9]$, $\delta = 0.1$ and $\varepsilon = 0.0001$, we get a relaxed Lyapunov function $V(x, y) = x^2 - 0.70528691608xy + 1.34394364271y^2 - 0.235095638693y^3 + 0.17632172902x^4 + 0.35264345804x^2y^2 + 0.17632172902y^4$.

Example 6: A three-dimensional example from [22]:

$$\begin{cases} \dot{x}_1 = -x_2 \\ \dot{x}_2 = -x_3 \\ \dot{x}_3 = -x_1 - 2x_2 - x_3 + x_1^3 \end{cases}$$

Let $V(x_1, x_2, x_3) = ax_1^2 + bx_2^2 + cx_3^2 + dx_1x_2 + ex_1x_3 + fx_2x_3$, then $\dot{V}(x_1, x_2, x_3) = (-d + e)x_1^2 - 2fx_2^2 + (-2c - f)x_3^2 + (-2a - 2e - f)x_1x_2 + (-2c - d - e)x_1x_3 + (-2b - e - f)x_2x_3 + 2cx_1^3x_3 + ex_1^4 + fx_1^3x_2$.

If we choose $B = [-0.2, 0.2] \times [-0.2, 0.2] \times [-0.2, 0.2]$, $\delta = 0.1$ and $\varepsilon = 0.0001$, we get a relaxed Lyapunov function $V(x_1, x_2, x_3) = x_1^2 + 0.494353826851x_2^2 + 0.505646173149x_3^2 + -1.0112923463x_1x_3 + 0.0225846925972x_2x_3$.

Example 7: An example from a Chinese textbook on ODEs:

$$\begin{cases} \dot{x} = -x - 3y + 2z + yz \\ \dot{y} = 3x - y - z + xz \\ \dot{z} = -2x + y - z + xy \end{cases}$$

Let $V(x, y, z) = ax^2 + by^2 + cz^2$, then $\dot{V}(x, y, z) = -2ax^2 - 2by^2 - 2cz^2 + (-6a + 6b)xy + (4a - 4c)xz + (-2b + 2c)yz + (2a + 2b + 2c)xyz$

If we choose $B = [-0.4, 0.4] \times [-0.4, 0.4] \times [-0.4, 0.4]$, $\delta = 0.1$ and $\varepsilon = 0.0001$, we get a relaxed Lyapunov function $V(x, y, z) = x^2 + y^2 + z^2$.

TABLE I
$relax(\phi)$

Example	CPU time	Branching steps
1	2.21s	49
2	$>$ 4 hours	unknown
3	404.70s	1380
4	$>$ 4 hours	unknown
5	$>$ 4 hours	unknown
6	$>$ 4 hours	unknown
7	$>$ 4 hours	unknown
8	$>$ 4 hours	unknown

TABLE II
$relax^{=}(\phi)$

Example	CPU time	Branching steps
1	2.16s	49
2	79.63s	985
3	1340.32s	2739
4	27.11s	244
5	228.89s	523
6	98.79s	506
7	5861.59s	4044
8	$>$ 4 hours	unknown

Example 8: A six-dimensional system from [10]:

$$\begin{cases} \dot{x}_1 = -x_1^3 + 4x_2^3 - 6x_3x_4 \\ \dot{x}_2 = -x_1 - x_2 + x_5^3 \\ \dot{x}_3 = x_1x_4 - x_3 + x_4x_6 \\ \dot{x}_4 = x_1x_3 + x_3x_6 - x_4^3 \\ \dot{x}_5 = -2x_2^3 - x_5 + x_6 \\ \dot{x}_6 = -3x_3x_4 - x_5^3 - x_6 \end{cases}$$

Let $V(x_1, x_2, x_3, x_4, x_5, x_6) = ax_1^2 + bx_2^4 + cx_3^2 + dx_4^2 + ex_5^4 + fx_6^2$, then $\dot{V}(x_1, x_2, x_3, x_4, x_5, x_6) = -2ax_1^4 - 4bx_2^4 - 2cx_3^4 - 2dx_4^4 - 4ex_5^2 - 2fx_6^2 + (8a - 4b)x_1x_2^3 + (-12a + 2c + 2d)x_1x_3x_4 + (4b - 8e)x_2^3x_5^3 + (2c + 2d - 6f)x_3x_4x_6 + (4e - 2f)x_5^3x_6$.

Choosing $B = [-0.8, 0.8] \times \cdots \times [-0.8, 0.8]$, $\delta = 0.1$ and $\varepsilon = 0.0001$, computing by hand, we can easily get a relaxed Lyapunov function $V(x_1, x_2, x_3, x_4, x_5, x_6) = x_1^2 + 2x_2^4 + 3x_3^2 + 3x_4^4 + x_5^4 + 2x_6^2$.

Note that using our solver, we need a lot of branching for getting this result since this is a six-dimensional example.

The computations were performed on an IBM notebook of Pentium IV, 1.70 GHz with 1 GB RAM, and they were canceled in cases when computation did not terminate before 4 hours of computation time. The computing times and the number of branching steps are listed in Table I (algorithm with relaxation $relax(\phi)$), and Table II (algorithm with improved relaxation $relax^{=}(\phi)$).

Clearly, the improved relaxation scheme also improves the overall algorithm. The only example is Example 3: The main reason is, that in this cases the branching heuristics happen to be more lucky for $relax(\phi)$ than for $relax^{=}(relax)$. For the other examples, as expected, the results show that the Algorithm 1 is better than the basic algorithm.

VI. Related Work

We are only aware of two methods that can compute non-linear Lyapunov functions in a completely automatic way. One is a method based on sum of squares decomposition using relaxation to linear matrix inequalities [10], [11]. The other method is to use Gröbner bases to choose the parameters in Lyapunov functions in an optimal way [4]. This requires the computation of a Gröbner basis for an ideal with a large number variables, and requires some manual intervention to distinguish critical points from optima.

Many methods proposed for computing the region of attraction can be roughly divided into two classes: one class is to maximizes the size of a region of attraction over a certain given Lyapunov function [22]; another class generates a convergent sequence of regions, starting from a small initial region of attraction [5].

VII. Conclusion

In this paper we have provided a method for computing the basin of attraction to a target region. The method is based on computation of Lyapunov-like functions using a branch-and-relax constraint solving algorithm. It seems that similar constraints have to be solved in many other areas (e.g., proving the termination of term-write systems, computation of barrier certificates [13], invariant generation [21], [18], [9], [19], [14], and analysis of FEM [23]), and it is interesting work to apply our algorithms in these areas.

We will further increase the efficiency of our method, for example, by improving the used branching heuristics, and we will generalize our results to the stability analysis of hybrid systems [12], [2].

References

[1] R. Alur and G. J. Pappas, editors. *Hybrid Systems: Computation and Control*, number 2993 in LNCS. Springer, 2004.

[2] H. Burchardt, J. Oehlerking, and O. Theel. The role of state-space partitioning in automated verification of affine hybrid system stability. In *Proc. of the 3rd Intl. Conf. on Computing, Communications and Control Technologies*, volume 1, pages 187–192. International Institute of Informatics and Systemics, 2005.

[3] B. F. Caviness and J. R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer, Wien, 1998.

[4] K. Forsman. Construction of Lyapunov functions using Gröbner bases. In *Proc. of the 30th Conf. on Decision and Control*, pages 798–799, 1991.

[5] R. Genesio, M. Tartaglia, and A. Vicino. On the estimation of asympototic stability regions: state of the art and new proposals. *IEEE Trans. on Automatic Control*, 30(8):747–755, 1985.

[6] W. Hahn. *Stability of Motion*. Springer, 1967.

[7] V. Lakshmikantham, S. Leela, and A. Martynyuk. *Practical Stability of Nonlinear Systems*. World Scientific, 1990.

[8] M. Morari and L. Thiele, editors. *Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*. Springer, 2005.

[9] M. Müller-Olm and H. Seidl. Computing polynomial program invariants. *Inf. Process. Lett.*, 91(5):233–244, 2004.

[10] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *Proc. of the IEEE Conf. on Decision and Control*, 2002.

[11] P. Parrilo and S. Lall. Semidefinite programming relaxations and algebraic optimization in control. *European Journal of Control*, 9(2–3), 2003.

[12] A. Podelski and S. Wagner. Model checking of hybrid systems: From reachability towards stability. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, volume 3927 of *LNCS*. Springer, 2006.

[13] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In Alur and Pappas [1].

[14] S. Prajna and A. Rantzer. Primal-dual tests for safety and reachability. In Morari and Thiele [8].

[15] S. Ratschan. Continuous first-order constraint satisfaction. In J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, and V. Sorge, editors, *Artificial Intelligence, Automated Reasoning, and Symbolic Computation*, number 2385 in LNCS, pages 181–195. Springer, 2002.

[16] S. Ratschan. RSolver. `http://rsolver.sourceforge.net`, 2004. Software package.

[17] S. Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Transactions on Computational Logic*, 2005. To appear.

[18] E. Rodriguez-Carbonell and D. Kapur. Automatic generation of polynomial loop invariants: Algebraic foundations. In *Proc. Intl. Symp on Symbolic and Algebraic Computation, ISSAC-2004*, 2004.

[19] E. Rodriguez-Carbonell and A. Tiwari. Generating polynomial invariants for hybrid systems. In Morari and Thiele [8].

[20] J. Rohn and J. Kreslová. Linear interval inequalities. *Linear and Multilinear Algebra*, 38:79–82, 1994.

[21] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In Alur and Pappas [1].

[22] D. N. Shields and C. Storey. The behaviour of optimal lyapunov functions. *Int. J. Control*, 21(4):561–573, 1975.

[23] P. Šolín and T. Vejchodský. On a discrete maximum principle for one-dimensional *hp*-FEM. Submitted.

[24] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Univ. of California Press, Berkeley, 1951. Also in [3].