A SEMI-ALGEBRAIC APPROACH FOR THE COMPUTATION OF LYAPUNOV FUNCTIONS

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

Bican Xia and Rong Xiao
School of Mathematical Sciences, Peking University
Beijing, China
email: xbc@math.pku.edu.cn, rxiao@pku.edu.cn

ABSTRACT

In this paper we deal with the problem of computing Lyapunov functions for stability verification of differential systems. We concern on symbolic methods and start the discussion with a classical quantifier elimination model for computing Lyapunov functions in a given polynomial form, especially in quadratic forms. Then we propose a new semi-algebraic method by making advantage of the local property of the Lyapunov function as well as its derivative. This is done by first using real solution classification to construct a semi-algebraic system and then solving this semi-algebraic system. Our semi-algebraic approach is more efficient in practice, especially for low-order systems. This efficiency will be evaluated empirically.

KEY WORDS

autonomous systems, Lyapunov functions, asymptotic stability, semi-algebraic system, partial-CAD.

1 Introduction

The computation of Lyapunov functions for dynamical systems plays a very important role in control system analysis and design. On one hand, Lyapunov functions can be used for verifying the stability. On the other hand, making use of Lyapunov functions, one can further compute attraction regions which give more details about the stability. In this paper, we deal with the problem of computing Lyapunov functions for verifying the asymptotic stability of polynomial differential systems.

For a polynomial differential systems, the problem of computing a Lyapunov function in a given polynomial form can be naturally transformed into a quantifier elimination (QE) problem, which is decidable [22, 4]. However, this method has some well known disadvantages and is hard to get an efficient implementation.

We propose a new method by making advantage of the local advantage of the local properties of a Lyapunov function in quadratic form as well as its derivative. That is, we first use real solution classification to construct a semi-algebraic system and then solve this resulting semi-algebraic system. When doing this, we may ignore the neighborhood of the origin and reduce the number of total variables. Further, our method can be extended to some more general cases, and we will use a simple example to explain the idea of this extension which will come out later in our extended paper in details.

We implemented our algorithm based on the MAPLE package DISCOVERER [27] and tested our implementation on some examples in literature.

The structure of the paper is as follows. In Section 2 we formalize our problem of computing Lyapunov functions for verifying asymptotic stability. Section 3 describes a classical quantifier elimination method for computing Lyapunov functions and we propose a new method in Section 4 by using a semi-algebraic system based approach. In Section 5 some examples are shown with computation results and timings by our algorithm. Section 6 devotes to some related work and we conclude the paper in Section 7.

2 The Problem

Consider an autonomous polynomial system of differential equations $\dot{\vec{x}} = f(\vec{x})$, where $f(\vec{x})$ is a polynomial from $\mathbb{R}^n$ to $\mathbb{R}^n$. We denote such a system by PSf.

For a given system PSf, a point $\vec{x}^*$ is called an *equilibrium* of this given system if $f(\vec{x}^*) = \vec{0}$. Without loss of generality, we suppose the origin to be an equilibrium of this given system if its equilibria exist.

From now on, if not specified, a differential system means an autonomous polynomial system of differential equations with an equilibrium at the origin. Moreover, for a given differential system, the asymptotic stability means the asymptotic stability of the origin in the Lyapunov sense [11].

A necessary and sufficient condition for verifying asymptotic stability [11] is the existence of a Lyapunov function (LF), which is defined as follows.

Definition 1 *Given a differential system PSf and a neighborhood $\mathbf{U}$ of the origin, a* Lyapunov function *is a differentiable function $V : \mathbf{U} \mapsto \mathbb{R}$ such that*

- $V(\vec{0}) = 0$ *and* $V(\vec{x}) > 0$ *whenever* $\vec{x} \neq \vec{0}$*;*
- $\frac{d}{dt}V(\vec{0}) = 0$ *and* $\frac{d}{dt}V(\vec{x}) < 0$ *whenever* $\vec{x} \neq \vec{0}$*.*

In this paper, we would like to have an algorithm that, for a given differential system PSf, computes a Lyapunov function (LF) for the verification of asymptotic stability. In general, this is an undecidable problem [12]. So we aim at an efficient algorithm for computing a polynomial Lyapunov function such that its terms are all of degree 2, that is, a polynomial in quadratic form, if such a polynomial Lyapunov function exists. It seems that polynomial Lyapunov functions in quadratic forms play important roles in the literature of the verification of hybrid systems [16, 2, 6, 3].

3 The Quantifier Elimination Method

In this section, we will describe a classical method for computing a Lyapunov function (LF) in quadratic (or any polynomial) form for the verification of asymptotic stability.

For a given differential system PSf and a neighborhood $\mathbf{U}$ of the origin. Let V be a polynomial in quadratic form with parametric coefficients, and $\frac{d}{dt}V$ 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. Set $Cond(V)$ to be

$$\left[\vec{x} = \vec{0} \Leftrightarrow V(\vec{x}) = 0\right] \wedge V(\vec{x}) \geq 0 \\ \wedge \left[\vec{x} = \vec{0} \Leftrightarrow \frac{d}{dt}V(\vec{x}) = 0\right] \wedge \frac{d}{dt}V(\vec{x}) \leq 0.$$

If we can find a solution to

$$\forall \vec{x} \in \mathbf{U}\ [Cond(V)]$$

for the parameters that form the coefficients of V,then the V formed by this solution is a Lyapunov function.

In general, $\mathbf{U}$ is an arbitrary neighborhood of the origin, and does not have an explicit algebraic expression. However, we can view $\mathbf{U}$ as a ball $\{\vec{x} \in \mathbb{R} : \sum_{i=1}^{n} x_i^2 < r^2\}$ or a block $\{\vec{x} \in \mathbb{R} : |x_i| < r, 1 \leq i \leq n\}$. If we view $\mathbf{U}$ as a block, we get the following constraint:

$$\exists r > 0 \forall \vec{x} \in \mathbb{R}^n \left[[\wedge_{1 \leq i \leq n} |x_i| < r] \Rightarrow Cond(V)\right]. \quad (1)$$

The constraint (1) is a formula in the first-order predicate language over the real numbers. Due to decidability of the theory over real-closed fields [22], one can always check whether for a given polynomial with parametric coefficients, there are instantiations of these parameters resulting in a Lyapunov function by applying the quantifier elimination (QE) method. As a QE procedure, it eliminates the quantified variables, gives some equivalent quantifier free formula for the parameters and then samples values of the parameters.

Here, we will employ a cylindrical algebraic decomposition (CAD) based QE method [4] — the details for CAD will come out later — and use an example to explain how it works by executing **QEPCAD B** [5]—a CAD based QE tool.

Example 1 *Consider* $(\dot{x}, \dot{y}) = (y, -x - y + x^2)$. *Let* $V(x, y) = x^2 + axy + y^2$. *The input to* **QEPCAD B** *is* $(Er)(Ax)(Ay)[r > 0 \wedge [x > r \vee y > r \vee x + r < 0 \vee y + r < 0 \vee x = 0 \vee [a > -2 \wedge a < 2 \wedge y(2x + ay) + (-x - y + x^2)(ax + 2y) < 0]]]$, *which is equivalent to the Constraint (1). After a computation of about* 2000 *seconds, we got an equivalent quantifier free formula* $0 < a < \frac{8}{5}$. *And this implies that* $\forall a\ 0 < a < \frac{8}{5}$, $V = x^2 + axy + y^2$ *form a LF.*

Clearly, the QE method functions in the same way for computing polynomial Lyapunov functions in any given forms. However, it is well known that the QE method is of low efficiency in practice, and can hardly produce a result when the number of total variables is greater than 5.

4 A Semi-Algebraic Systems based Remedy

There are two shortages of the QE method for computing LFs introduced in the previous section: one is that we need to compute Lyapunov functions as well as neighborhoods; the other is that there are still too many total variables to solve even though we only consider quadratic forms. In this section, we will try to find a remedy for these problems. The idea is to compute a Lyapunov function in quadratic form using its local property to first construct a semi-algebraic system (SAS) — the formal definition will appear in Subsection 4.3 — and then find a solution to this resulting system.

4.1 Semi-Algebraic Systems based Computation of Lyapunov Functions

In this subsection, we will compute a Lyapunov function in quadratic form by making advantage of the local properties of a Lyapunov function as well as its derivative, that is, they are locally either positive or negative. Specifically, we will concentrate on the case that the Hessian Matrix of a Lypunov function is positive definite and its derivative's Hessian Matrix is negative definite. Following this way, we first use real solution classification to construct a semi-algebraic system and then solve this system.

The Hessian matrix at the origin for a differential function V is defined as follows:

$$Hess(V)|_{\vec{x}=\vec{0}} = \left.\begin{bmatrix} \frac{\partial^2 V}{\partial x_1^2} & \frac{\partial^2 V}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 V}{\partial x_1 \partial x_n} \\ \frac{\partial^2 V}{\partial x_2 \partial x_1} & \frac{\partial^2 V}{\partial x_2^2} & \cdots & \frac{\partial^2 V}{\partial x_2 \partial x_n} \\ & \cdots & \cdots & \\ \frac{\partial^2 V}{\partial x_n \partial x_1} & \frac{\partial^2 V}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 V}{\partial x_n^2} \end{bmatrix}\right|_{\vec{x}=\vec{0}}$$

Now, based on some local properties, we state a sufficient condition for the existence of a Lyapunov function:

Theorem 1 *For a given differential system* PSf, *and a polynomial* $V(x)$ *in quadratic form such that* $Hess(V)|_{\vec{x}=\vec{0}}$ *is positive-definite and* $Hess(\frac{d}{dt}V)|_{\vec{x}=\vec{0}}$ *is negative-definite, then* V *is a Lyapunov function.*

Proof. Obviously, $V = \frac{1}{2}\vec{x} Hess(V)|_{\vec{x}=\vec{0}} \vec{x}^T$. It is sufficient to prove that there is a neighborhood $\mathbf{U}$ of the origin such that $\frac{d}{dt}V(\vec{x}) < 0$ for all $\vec{x} \in \mathbf{U} \setminus \{\vec{0}\}$.

Since $(\frac{\partial}{\partial x_1}(\frac{d}{dt}V), \ldots, \frac{\partial}{\partial x_n}(\frac{d}{dt}V))|_{\vec{x}=\vec{0}} = \vec{0}$ and $Hess(\frac{d}{dt}V)|_{\vec{x}=\vec{0}}$ is negative definite, due to the extremum theory, there is a neighborhood $\mathbf{U}$ of the origin such that $\frac{d}{dt}V^*(\vec{x}) < \frac{d}{dt}V^*(\vec{0}) = 0$ for all $\vec{x} \in \mathbf{U} \setminus \{\vec{0}\}$. ■

Due to Theorem 1, we try to find a Lyapunov function in quadratic form such that the conditions described in Theorem 1 hold. Following this idea, we will first use the real solution classification to construct a semi-algebraic system and then solve it in a symbolic way by finding a solution to this resulting semi-algebraic system.

Observing that the two Hessian matrices are both symmetric, it is easy to know that their eigenvalues are all real. On the other hand, a necessary and sufficient condition for a matrix to be positive-definite (negative definite) is that its eigenvalues are all positive (negative). This leads us to study its characteristic polynomial.

Definition 2 *A univariate polynomial is* totally real *if its roots are all real. A totally real polynomial is* positive rooting *if its roots are all positive.*

For a given symmetric matrix H with parametric entries, let $h(\lambda) = \lambda^n + c_{n-1}(\vec{a})\lambda^{n-1} + \cdots + c_0(\vec{a})$ be its characteristic polynomial, where the components of $\vec{a}$ are the parameters that form H and $c_i(\vec{a})$'s are polynomials in $\vec{a}$. Obviously, h is totally real. Moreover, H is positive-definite if and only if h is positive rooting.

The following theorem — which can be deduced from the Descartes rule of signs — gives the necessary and sufficient condition for checking whether a totally real polynomial is positive rooting or not.

Theorem 2 *[23] For a totally real polynomial $h(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_0$, it is positive rooting if and only if for all $1 \le i \le n$, $(-1)^i c_{n-i} > 0$.*

Due to Theorem 1, for a given differential system PSf, if we can find a solution for the parameters that form a symmetric Matrix H such that H is positive definite and $H^* = Hess(\frac{d}{dt}(\frac{1}{2}\vec{x}H\vec{x}^T))|_{\vec{x}=\vec{0}}$ is negative definite, then $\frac{1}{2}\vec{x}H\vec{x}^T$ is a Lyapunov function.

From Theorem 2, the symmetric matrix H is positive-definite if and only if $c_{n-1}(\vec{a}) > 0, c_{n-2}(\vec{a}) < 0, \cdots, (-1)^n c_0(\vec{a}) > 0$, where $c_i(\vec{a})$'s — which are polynomials in parameters $\vec{a}$ — are the coefficients of the characteristic polynomial of H. Thus, to find a solution for the parameters such that H is positive definite is equivalent to find a solution to the semi-algebraic system $\{c_{n-1}(\vec{a}) > 0, c_{n-2}(\vec{a}) < 0, \cdots, (-1)^n c_0(\vec{a}) > 0\}$. Similarly, to find a solution for the parameters such that $-H^*$ is positive definite is equivalent to find a solution to the semi-algebraic system $\{c^*_{n-1}(\vec{a}) > 0, c^*_{n-2}(\vec{a}) < 0, \cdots, (-1)^n c^*_0(\vec{a}) > 0\}$, where $c^*_i(\vec{a})$'s are the coefficients of the characteristic polynomial of $-H^*$.

Combining the above two discussions, to find a solution for the parameters such that H and $-H^*$ are both positive definite, is equivalent to find a solution to the semi-algebraic system $\{c_{n-1}(\vec{a}) > 0, \cdots, (-1)^n c_0(\vec{a}) > 0, c^*_{n-1}(\vec{a}) > 0, \cdots, (-1)^n c^*_0(\vec{a}) > 0\}$. We denote this resulting semi-algebraic system by $SAS(H, -H^*)$. This results in Algorithm 1, whose core algorithm is **SASolver**. Note that for a given semi-algorithm system, **SASolver** either returns a solution to this system or returns an empty set when there is no solution to this system. The details for **SASolver** will come out in Subsection 4.3.

Algorithm 1 Computing Lyapunov functions

Input: A given differential system PSf.
Output: A Lyapunov function or UNKNOWN.

choose a polynomial in quadratic form with parametric coefficients.
Compute $H = Hess(V)|_{\vec{x}=\vec{0}}$ and $H^* = Hess(\frac{d}{dt}V)|_{\vec{x}=\vec{0}}$.
compute the characteristic polynomials of H and $-H^*$, respectively.
compute $SAS(H, -H^*)$.
apply **SASolver** to $SAS(H, -H^*)$.
if SASolver returns a solution **then**
  set this solution to the parameters in V and return V.
else
  return UNKNOWN.
end if

We use the following example to explain Algorithm 1 with comparison to the QE method.

Example 2 *This is an example from [15] whose Lyapunov function has been constructed by the sum of squares decomposition.*

$$\begin{cases} \dot{x} = -x + y + xy \\ \dot{y} = -x - x^2 \end{cases}$$

Let $V = ax^2 + bxy + cy^2$, then $\frac{d}{dt}V = (2ax + by)(-x + y + xy) + (bx + 2cy)(-x - x^2)$. And the two required characteristic polynomials are $C(t) = t^2 - 2at - 2ct + 4ac - b^2$ and $D(t) = t^2 - 4at - 4ab - 5b^2 - 4a^2 + 8ac - 4bc - 4c^2$. Applying Algorithm 1, **SASolver** *found a solution $(a, b, c) = (1, -1, 1)$ within one second. Thus $V = x^2 - xy + y^2$ is a Lyapunov function.*

When using the QE method, we let $V = x^2 + axy + y^2$ for simplicity. Then $\frac{d}{dt}V = (2x + ay)(-x + y + xy) + (ax + 2by)(-x - x^2)$. After applying **QEPCAD B** *to the Constraint (1), the program terminates abnormally after a computation of about 4868 seconds.*

Note that our semi-algebraic system based approach is equivalent to first linearize the system and then apply linear matrix inequalities (LMIs) based approach. That is, each positive definite matrix found by LMIs based approach forms a solution to our semi-algebraic system and each solution to our semi-algebraic system forms a positive definite matrix which satisfies the LMIs. The proof for this equivalence will be a very interesting issue for readers.

4.2 Discussion

In Subsection 4.1, we discussed how to ignore the neighborhood of the origin and compute a Lyapunov function in quadratic form by first constructing a semi-algebraic system and then solving this system. In this way, we root out the first shortage that occurs when we use the quantifier elimination method. Moreover, we ease the second shortage since our semi-algebraic approach requires less variables. This is easily derived from the fact that, if using the general elimination method, we need to use at most $1 + n + \frac{(n+1)n}{2}$ total variables; if using the semi-algebraic approach, we only use at most $\frac{(n+1)n}{2}$ total variables. Thus, due to the double-exponential complexity of these two methods, our semi-algebraic approach is more efficient in practice, especially for the low-order (second and third) systems. Note that the complexity analysis in details will come out later in our extended paper. This efficiency will also be evaluated empirically in Section 5.

Although our semi-algebraic approach focuses on quadratic forms, it can be extended to any differentiable forms. The following theorem gives a sufficient condition on local properties of a function for it to be a LF.

Theorem 3 *For a given differential system PSf, if there exists a differentiable function $V(\vec{x})$ such that*

- $V(\vec{0}) = 0$, $(\frac{\partial V}{\partial x_1}, \dots, \frac{\partial V}{\partial x_n})|_{\vec{x}=\vec{0}} = \vec{0}$ *and* $Hess(V)|_{\vec{x}=\vec{0}}$ *is positive-definite;*
- $Hess(\frac{d}{dt}V)|_{\vec{x}=\vec{0}}$ *is negative-definite,*

then V is a Lyapunov function.

Proof. Since $(\frac{\partial V}{\partial x_1}, \dots, \frac{\partial V}{\partial x_n}) = \vec{0}$ and $Hess(V)|_{\vec{x}=\vec{0}}$ is positive definite, due to the extremum theory, there is a neighborhood $\mathbf{U}_1$ of the origin such that $V(\vec{x}) > V(\vec{0}) = 0$ for all $\vec{x} \in \mathbf{U}_1 \setminus \{\vec{0}\}$.

On the other hand, $\frac{d}{dt}V(\vec{0}) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \cdot f_i|_{\vec{x}=\vec{0}} = 0$. And for any arbitrary but fixed j, $1 \le j \le n$, $\frac{\partial(\frac{d}{dt}V)}{\partial x_j} = \sum_{i=1}^{n}(\frac{\partial^2 V}{\partial x_j \partial x_i} \cdot f_i + \frac{\partial V}{\partial x_i} \cdot \frac{\partial f_i}{\partial x_j})|_{\vec{x}=\vec{0}} = 0$. Thus, $(\frac{\partial(\frac{d}{dt}V)}{\partial x_1}, \dots, \frac{\partial(\frac{d}{dt}V)}{\partial x_n})|_{\vec{x}=\vec{0}} = \vec{0}$. Since $Hess(\frac{d}{dt}V)|_{\vec{x}=\vec{0}}$ is negative-definite, due to the extremum theory, there is a neighborhood $\mathbf{U}_2$ of the origin such that $\frac{d}{dt}V(\vec{x}) < \frac{d}{dt}V(\vec{0}) = 0$ for all $\vec{x} \in \mathbf{U}_2 \setminus \{\vec{0}\}$.

Set $\mathbf{U} = \mathbf{U}_1 \cap \mathbf{U}_2$. Obviously, V restricted to $\mathbf{U}$ is a Lyapunov function. ■

According to Theorem 3, if $V(\vec{x})$ is a Lyapunov function obtained from Algorithm 1, then $V^*(\vec{x}) = V(\vec{x}) + P_{\ge 3}(\vec{x})$ is also a Lyapunov function, where $P_{\ge 3}(\vec{x})$ is a polynomial such that its terms are all of degree ≥ 3.

Moreover, we can extend Theorem 2 and our algorithm to compute a Lyapunov-like function, that is, a polynomial such that its Hessian matrix at the origin is positive definite and its derivative's Hessian matrix at the origin is negative semi-definite. Then, the QE method is used to check whether this resulting Lyapunov-like function is a LF or not. The detail will come out in our extended paper. Here, we simply use an example to explain this idea.

Example 3 *Consider $(\dot{x}_1, \dot{x}_2) = (-x_2 - x_1^3, x_1 - x_2^3)$. Assume that $V(x_1, x_2) = x_1^2 + ax_2^2$. Then, $a = 1$ is a solution such that $Hess(V)|_{\vec{x}=\vec{0}}$ is positive definite and $Hess(\frac{d}{dt}V)|_{\vec{x}=\vec{0}}$ is negative semi-definite. Applying the quantifier elimination, it is easy to check that $V(x_1, x_2) = x_1^2 + x_2^2$ is a Lyapunov function.*

4.3 SASolver

In this subsection we come to semi-algebraic systems and study how to solve them.

A *semi-algebraic system* is a set of polynomial equations and inequalities over the real number field. A *semi-algebraic set* is the solution set of a semi-algebraic system in $\mathbb{R}^n$. For simplicity, we denote them both by SAS.

To solve a SAS has diverse meanings due to different applications: sometimes we want to have real solution classification based on the parameters; sometimes we want to find solutions of the SAS. The core of Algorithm 1 is **SASolver**, which can find one solution of a given semi-algebraic system (i.e., a sample point in the semi-algebraic set) or return $\emptyset$ if no solution exists. Note that the current version of **SASolver** is based on a Partial-CAD process.

To arrive at Partial-CAD, we start with CAD [4, 1]. CAD plays a very important role in both the QE method and our **SASolver**. It is a data structure to express the SAS in $\mathbb{R}^n$. Every CAD relates to a polynomial F, which is the product of all elements in a set of polynomials $\mathcal{F}$, and divides the $\mathbb{R}^n$ space into finitely many ordered connected sets such that F is ordered and sign-invariant in such sets. Each connected set in a CAD is called a *cell* and a point in the cell is called a *sample point* of the cell. Moreover, the CAD is a recursive structure, that is, we first define a cell in $\mathbb{R}^1$ as an open intervals or a point; and a cell in R^{k+1} then has the form $\{(x, y) : x \in C, f(x) < y < g(x)\}$ or $\{(x, y) : x \in C, y = f(x)\}$, where C is a cell in $\mathbb{R}^k$, and f and g are both continuous function on C such that for some polynomials F and G, $F(x, f(x)) = 0$ and $G(x, g(x)) = 0$, or $\pm\infty$ and $f(x) < g(x)$ for all $x \in C$.

Due to the recursive structure, a CAD algorithm [4, 1] generally includes three basic processes: *projecting*, *lifting* and *definition formula constructing*. However, since we are only interested in finding a sample point in the SAS, we can simplify these processes by using *Partial-CAD*, which origins from [5].

The Partial-CAD we process [26, 27] here is to first consider the open cells of the highest dimension, leaving all the lower dimensional ones, which are defined as boundaries. If we cannot find a solution in such cells, we add the boundaries as equality constraints, and process triangular decomposition with these equality constraints to get a regular triangular system. If no solution found, we proceed in the same way for lower dimensional cells until we can get one solution or we can assert there is no solution. And it's even lucky that we can ignore the boundaries all the time safely in **SASolver** here since the SAS we got includes only strict inequalities.

5 Examples

In this section, we will use five examples to show the efficiency of our Algorithm 1. Note that we used a computer with an Intel Pentium 2.60 GHz CPU with 1024 Mbytes of main memory running $Maple10$.

Example 4 *An example of the simplified model of a chemical oscillator in [14]. The original system is:*

$$\begin{cases} \dot{u} = a - u + u^2 v \\ \dot{v} = b - u^2 v \end{cases}$$

Let $(a, b) = (0.5, 0.5)$*, then the equilibrium is* $(1, 0.5)$*. Substituting* u *and* v *with* $u + 1$ *and* $v + 0.5$ *respectively, we got a new system* $(\dot{u}, \dot{v}) = (v + \frac{1}{2}u^2 + 2uv + u^2 v, -u - v - \frac{1}{2}u^2 - 2uv - u^2 v)$*. Letting* $V = au^2 + buv + cv^2$*, we got a solution* $(a, b, c) = (1, 1, 1)$ *within 2 seconds which implies that* $V = u^2 + uv + v^2$ *is a Lyapunov function of the new system.*

Example 5 *An example in [10]:*

$$\begin{cases} \dot{x}_1 = -2x_1 + x_2 + x_1^3 + x_2^5 \\ \dot{x}_2 = -x_1 - x_2 + x_1^2 x_2^3 \end{cases}$$

We assume that $V(x_1, x_2) = ax_1^2 + bx_1x_2 + cx_2^2$*. Algorithm 1 returned* $V(x_1, x_2) = x_1^2 - x_1x_2 + x_2^2$ *as a Lyapunov function within* 0.1 *seconds.*

Example 6 *An example whose Lyapunov function has been computed by Gröbner basis [8]:*

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

Letting $V(x_1, x_2, x_3) = x_1^2 + a_{12}x_1x_2 + a_{13}x_1x_3 + a_{22}x_2^2 + a_{23}x_2x_3 + a_{33}x_3^2$*, we find* 20 *solutions for* $(a_{12}, a_{13}, a_{22}, a_{23}, a_{33})$*. Taking* $(a_{12}, a_{13}, a_{22}, a_{23}, a_{33}) = (3, 1, 3, 0, 2)$ *among them, then* $V(x_1, x_2, x_3) = x_1^2 + 3x_1x_2 + x_1x_3 + 3x_2^2 + 2x_3^2$ *is a Lyapunov function.*

Example 7 *An example from a classical ODE's textbook:*

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

Assume that $V(x, y, z) = x^2 + axy + xz + cy^2 + dyz + ez^2$*. Within about* 1900 *seconds, we got* 500 *solutions for the parameters that form the coefficients of* V*. We take* $(a, c, d, e) = (2, 2, 2, 2)$ *among them and then* $V(x, y, z) = x^2 + 2xy + xz + 2y^2 + 2yz + 2z^2$ *is a Lyapunov function.*

Example 8 *Another example from an ODE's textbook:*

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

Let $V(x, y, z) = x^2 + bxz + cy^2 + dyz + ez^2$*. Running about* 840 *seconds, we got* 250 *solutions for the parameters. Taking* $(b, c, d, e) = (1, 1, -1, 1)$ *from them, then* $V = x^2 + xz + y^2 - yz + z^2$ *is a Lyapunov function.*

Note that, when using QE method, **QEPCAD B** failed to find Lyapunov functions for all the above examples because of the extreme time cost, memory cost, etc.

6 Related Work

The idea of using Lyapunov functions to verify asymptotic stability is not new. The difference is which approach to use for efficiently computing a Lyapunov function.

To our knowledge, there are two methods in the literature that can directly compute non-linear Lyapunov functions in an automatic way. One is a method based on sum of squares decomposition which can be efficiently computed by using semi-definite programming [18, 14, 15]. The other method is to use Gröbner bases to choose the parameters in Lyapunov functions in an optimal way [7, 8]. This requires the computation of a Gröbner basis for an ideal with a large number of variables, and requires some manual intervention to distinguish critical points from optima.

For verifying the asymptotic stability, another choice is to first linearize the original system and then study the linearization system. For doing this, there are two well-known methods. One method is to check the real parts of the eigenvalues of the Jacobian matrix at the origin by using Routh-Hurwitz's criterion [24]. However, we cannot verify the asymptotic stability if there are purely imaginary eigenvalues. Thus, for such cases, we still have to compute Lyapunov functions. The other method is to compute a Lyapuov function in quadratic form which results in linear matrix inequalities [16, 2, 6, 3].

7 Conclusion

In this paper we focus on computing Lyapunov functions in quadratic forms by a semi-algebraic system based approach for the verification of asymptotic stability of a PSf. This approach can be easily implemented and is more efficient in practice. And it can be applied to non-quadratic forms or extended to compute some Lyapunov-like functions.

An interesting problem is to compute attraction regions [21, 9, 25]. A further interesting problem is to partition the state space into finite many regions and compute piecewise Lyapunov-like functions over these regions [16].

Our long term goal is to verify the stability of switched and hybrid systems [6] in the Lyapunov sense by computing multiple Lyapunov functions [2], or in the practical sense [13] by computing multiple Lyapunov-like functions [20] or by computing transition systems [19, 17].

Acknowledgements. This work was partly supported by the German Research Foundation (SFB/TR 14 AVACS), Chinese-SALSA INRIA project, Chinese NKBRPC-2004CB318003, NKBRPC-2005CB321902 and NSFC-60573007.

References

[1] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal of Computing*, 13(4):865–877, 1984.

[2] M. S. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. on Automatic Control*, 43(4):475–482, 1998.

[3] 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.

[4] G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *LNCS* 33, 134–183, 1975.

[5] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12:299–328, 1991.

[6] R. A. Decarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7), 2000.

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

[8] K. Forsman. Optimization, stability and cylindrical decomposition. Technical report, Automatic Control Group in Linköping, 1993.

[9] 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.

[10] G. Ghesi, A. Tesi, and A. Vicino. On the optimal quadratic lyapunov functions for polynomial systems. In *15 th Int. Symp. on Mathematical Theory of Networks and Systems*, 2002.

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

[12] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata. *Journal of Computer and System Sciences*, 57:94–124, 1998.

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

[14] 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.

[15] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming Ser. B*, 96(2):293–320, 2003.

[16] S. Pettersson and B. Lennartson. An lmi approach for stability analysis of nonlinear systems. In *Proc. of the 4th European Control Conference*, 1997.

[17] 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*, *LNCS* 3927, Springer, 2006.

[18] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2002.

[19] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, *LNCS* 3414, pages 573–589, Springer, 2005.

[20] S. Ratschan and Z. She. Providing a basin of attraction to a target region by computation of Lyapunov-like functions. In *Proc. of the* 4*th IEEE Int. Conf. on Computational Cybernetics*, pages 245-249, 2006.

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

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

[23] D. Wang and B. Xia. *Computer Algebra*. Tsinghua Univ. Press, Beijing, 2004.

[24] D. Wang and B. Xia. Stability analysis of biological systems with real solution classification. In *ISSAC*, pages 354–361. ACM Press, New York, 2005.

[25] T.-C. Wang, S. Lall, and M. West. Polynomial level-set methods for nonlinear dynamical systems analysis. In *Proceedings of the Allerton Conference on Communication, Control and Computing*, 2005.

[26] L. Yang and B. Xia. An algorithm for isolating the real solutions of semi-algebraic systems. *Journal of Symbolic Computation*, 34: 461–477, 2002.

[27] L. Yang and B. Xia. Automated deduction in real geometry. In: *Geometric Computation*, World Scientific Press, 248–298, 2004.