2016 IEEE Conference on Computer Aided Control System Design (CACSD)
Part of 2016 IEEE Multi-Conference on Systems and Control
September 19-22, 2016. Buenos Aires, Argentina

Lyapunov Function Synthesis - Algorithm and Software

Tobias Leth[1], Christoffer Sloth[1], Rafał Wisniewski[1]
{tol, ces, raf}@es.aau.dk

***Abstract*— In this paper we introduce an algorithm for the synthesis of polynomial Lyapunov functions for polynomial vector fields. The Lyapunov function is a continuous piecewise-polynomial defined on simplices, which compose a collection of simplices. The algorithm is elaborated and crucial features are explained in detail. The strengths and weaknesses of the algorithm are exemplified and a new way of sub-dividing the simplices is presented.**

I. Introduction

Stability analysis plays a major role in the study of dynamic systems. Often the methods by Lyapunov are applied, and for linear systems they offer an easy way of certifying stability of equilibria. For non-linear systems, finding a Lyapunov function is a non-trivial task. For polynomial non-linearities the task of finding Lyapunov functions can be cast as a synthesis problem.

To this end some different toolboxes can be applied. Most notable are GloptiPoly [1] and SOSTOOLS [2]. They are two sides of the same coin and can be used for much more than Lyapunov function synthesis. Common for them is that when a synthesis problem is formulated they rely on sum-of-square (SOS) conditions for the positive definiteness. This translates to linear matrix inequalities (LMIs), and semi-definite programming (SDP) is needed to solve the problem. This makes scalability a practical limitation.

One way to improve the scalability is to solve the SDP by a sequence of increasingly accurate approximation using diagonally dominant SOS (DSOS) and scaled DSOS (SDSOS) polynomials [3]. This allows for linear programs (LP) and second order cone programs (SOCP) which are easier to solve than the SDP counterpart.

Another approach is to employ polynomials described in the Bernstein basis. The Bernstein basis has been wildly used for certifying positivity [4], for optimization [5] [6], and for finding Lyapunov functions [7] when dealing with polynomial non-linearities. Bernstein's Theorem [8] is vital when working on polynomials described in the Bernstein basis. It characterizes positive polynomials and is used for both positivity certification and Lyapunov function synthesis. When cast in the Bernstein basis, the problem of finding a Lyapunov function becomes an LP and the conditions for positive definiteness does not require any restriction on the class of polynomial.

This work is supported by the Danish Council for Independent Research under grant number DFF - 4005-00452 in the project CodeMe.

[1]Department of Electrical Engineering, section of Automation & Control, University of Aalborg, 9220 Aalborg East, Denmark

Using the Bernstein Theorem, [8] translates the Lyapunov criteria to polynomials in the simplicial Bernstein basis. We present an algorithm solving the synthesis problem by means of a linear feasibility problem (LFP). It searches for a continuous piecewise-polynomial Lyapunov function for a polynomial vector field defined on intervals. The interval is triangulated into a collection of simplices and the Lyaounov criteria are cast as an LFP in the unknown coefficients of the Lyapunov function. When infeasibility occurs we argue the use of sub-division of the simplicies favourable to raising the degree of the Lyapunov function. We present a new way to obtain a feasible problem by means of intelligent sub-division.

The paper is structured as follows: After introducing notation in section II, section III covers the translation of the Lyapunov criteria to polynomials defined in the Bernstein basis. Section IV derives the LFP and the algorithm for synthesizing polynomial Lyapunov functions. Examples of the algorithm are given in section V and a conclusion is drawn in section VI.

II. Bernstein Basis - Notation and Motivation

Based on [9] and [10] this section introduces simplices, the simplicial Bernstein Basis and some additional notation.

A nondegenerate n-simplex (a simplex of dimension n) $\sigma \equiv [\sigma_0, \ldots, \sigma_n]$, with ordering $<$ such that $\sigma_i < \sigma_j$ if $i < j$, is the convex hull created by $n+1$ affinely independent vertices $\sigma_i \in \mathbb{R}^n$ such that

$$[\sigma_0, \ldots, \sigma_n] = \left\{ \sum_{i=0}^{n} \lambda_i \sigma_i | \sum_{i=0}^{n} \lambda_i = 1 \wedge \lambda_i \geq 0 \right\}, \quad (1)$$

where λ_i are the barycentric coordinates associated with σ. An l-face of a n-simplex $\sigma^1 = [\sigma_0^1, \ldots, \sigma_n^1]$, with $l \leq n$, is any l-simplex $\sigma^2 = [\sigma_0^2, \ldots, \sigma_l^2]$ with ordering $<$ such that

$$\{\sigma_0^2, \ldots, \sigma_l^2\} \subseteq \{\sigma_0^1, \ldots, \sigma_n^1\}. \quad (2)$$

$(n-1)$-faces of an n-simplex are called facets and 0-faces are simply the vertices.

Defining a face operator as

$$\partial_i^{n+1} \circ \sigma = [\sigma_0, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n] \quad (3)$$

and writing $\partial_i^{n+1}\sigma = \partial_i^{n+1} \circ \sigma$ the facets of σ can be identified. Consecutive application of ∂ will identify the lower dimensional faces and, if applied n times, the vertices. Fig. 1 shows the 2 dimensional case where e.g. $\partial_1^n \partial_2^{n+1} \sigma^1 = \sigma_0^1$, $\partial_1^n \partial_0^{n+1} \sigma^2 = \sigma_1^2$, and so on. Note that the numbering of the vertices is arbitrary. In the following the superscript of

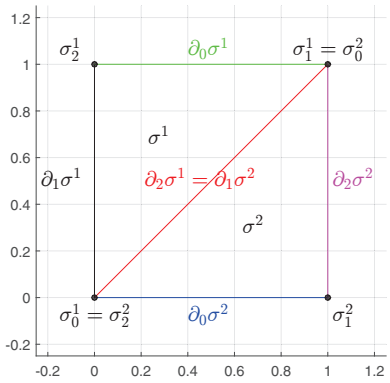
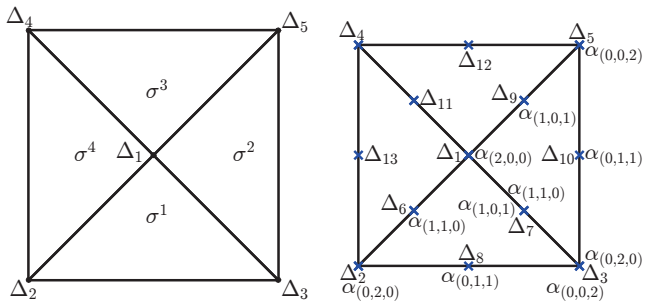


Fig. 1. Triangulation of the 2D unit cube into the collection of simplices K with simplices σ^i with facets $\partial_j\sigma^i$ and vertices σ^i_j.



(a) Triangulation of an arbitrary 2D box with the origin in its interior.

(b) Collection of simplicies and the associated grid points.

Fig. 2. Polynomial defined on collection of simplices. (a) shows the ordering of simplices and vertices. (b) expands Δ to include all grid points according to the order $<$ on the simplices and Eq. (10).

∂_i^{n+1} is dropped and the dimension, of the simplex on which ∂ operates, is assumed obvious from context.

The notation of a simplex is expanded to collections of simplices. Let $K = \{\sigma^1, \cdots, \sigma^m\}$ be a finite set of m non-overlapping (except for faces) n-simplices, and let them be ordered by $<$. Formally, one could refer to K as a simplicial complex, but since we will not be using the simplices of dimension lower than n, we will refer to K as a collection of simplices. See [11] for details. In relation to Fig. 1, the collection is $K = \{\sigma^1, \sigma^2\}$ and their intersection is $\partial_2\sigma^1 = \partial_1\sigma^2$ which is a face (in fact a facet) of both simplices.

Denoting $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$ we employ the following shorthand notation. For $\alpha \in \mathbb{N}_0^{n+1}$ we write

$$|\alpha| = \sum_{i=0}^{n} \alpha_i, \qquad \lambda^\alpha = \prod_{i=0}^{n} \lambda_i^{\alpha_i}, \tag{4}$$

and

$$\mathcal{B}_\alpha^D = \binom{D}{\alpha}\lambda^\alpha, \qquad \binom{D}{\alpha} = \frac{D!}{\alpha_0!\alpha_1!\cdots\alpha_n!}, \tag{5}$$

where $\mathcal{B}^D$ are the Bernstein basis polynomials of degree D.

With this, any polynomial p of degree d in n variables can be described in the Bernstein basis of degree $D \geq d$ on a simplex σ as

$$p = \sum_{|\alpha|=D} b_\alpha(p, D, \sigma)\mathcal{B}_\alpha^D = b(p, D, \sigma)\mathcal{B}^D. \tag{6}$$

The vector $b(p, D, \sigma)$ contains the coefficients which uniquely describe p. For each $b_\alpha(f, D, \sigma)$ a corresponding grid point is defined as [9]

$$\beta(D, \sigma) = \frac{\alpha_0\sigma_0 + \cdots + \alpha_n\sigma_n}{D}, \tag{7}$$

where $\alpha \in \mathbb{N}_0^{n+1}$ and $|\alpha| = D$. That is, every coefficient is located somewhere in the simplex. The pair $(\beta(D, \sigma), b_\alpha(f, D, \sigma))$ is the control point associated to α. Throughout the paper we shall makes use of Bernstein's Theorem, which we state as a corollary based on Thm. 5.7 in [9].

Corollary 1 *(Bernstein's Theorem) [9]. If a polynomial p of degree d is positive on a simplex σ, then there exists a sub-division of σ into a collection $K = \{\sigma^1, \cdots, \sigma^m\}$ of finitely many simplices such that*

$$b(p, d, \sigma^i) > 0 \quad \forall i \in \{1, \cdots, m\}. \tag{8}$$

The theorem characterizes all positive polynomials on σ of degree d and it is part of the motivation behind working in the Bernstein basis. Also, the Bernstein basis has two properties which are significant. The end-point value property which states that the coefficients at the vertices are equal to the polynomial when evaluated at the vertices [12]. The other property is that a continuous piecewise-polynomial described on two adjacent simplices has the same coefficients on the common facet.

A. Polynomials Defined on Collections of Simplicies

In the Bernstein basis, polynomials are defined uniquely on a simplex by their coefficient vector. Depending on the number of variables, n, and the degree of the representation, d, the number of coefficients is

$$N_d = \binom{n+d}{n}. \tag{9}$$

When the polynomial is defined on a collection of m simplicies the number of coefficients grows, but there are fewer than $N_d m$ coefficients. This is because coefficients located on common faces only appear once even though they define two (or more) polynomials on different simplices. In the following we take special care of keeping track of indices to make implementation easy and to make the algorithm work in arbitrary dimension.

Let Δ be a list of grid points, and initially let it consist of the 5 vertices shown in Fig. 2(a), where an arbitrary 2D box, which contains the origin in its interior, is triangulated. Except for Δ_1 which is placed in the origin, the vertices and simplices are arbitrarily numbered, but uniquely ordered by $<$. This makes e.g. $\sigma^2 = [\sigma_0^2, \sigma_1^2, \sigma_2^2] = [\Delta_1, \Delta_3, \Delta_5]$. Note that by placing Δ_1 in the origin we get $\sigma_0^i = \Delta_1$.

N_d is the number of possible combinations for $|\alpha| = d$ and we order them using Lexicographic order in the absolute numbering defined by Δ. For $n = d = 2$, $N_d = 6$ and the

possible α's are

$$\begin{array}{c} \\ \Delta_x \\ \Delta_y \\ \Delta_z \end{array} \begin{array}{c} \begin{array}{cccccc} \alpha_{(2,0,0)} & \alpha_{(1,1,0)} & \alpha_{(1,0,1)} & \alpha_{(0,2,0)} & \alpha_{(0,1,1)} & \alpha_{(0,0,2)} \end{array} \\ \left[\begin{array}{cccccc} 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 2 \end{array}\right] \end{array}, \tag{10}$$

with vertices $\Delta_x < \Delta_y < \Delta_z$ such that x, y, z refers to the vertices of one simplex. Using this order the initial list of grid points Δ, consisting of the 5 vertices, is expanded as follows: Starting in σ^1, Δ_6 is defined by $\alpha_{(1,1,0)}$, Δ_7 by $\alpha_{(1,0,1)}$ and Δ_8 by $\alpha_{(0,1,1)}$. Moving on to σ^2, Δ_9 is defined by $\alpha_{(1,0,1)}$ in σ^2 and Δ_{10} by $\alpha_{(0,1,1)}$. Note here, that the grid point at $\alpha_{(1,1,0)}$ in σ^2 is already defined as Δ_7 from $\alpha_{(1,0,1)}$ in σ^1. Continuing in this fashion we obtain Fig. 2(b) where the α's on σ^3 and σ^4 and $\alpha_{(2,0,0)}$ on σ^1 are omitted for clarity. As seen, even though there are 6 coefficients on either simplex, there are only 13 coefficients in the collection and $\Delta = [1, \cdots, 13]$. Denoting the coefficients on σ^i by Δ^i we get

$$\Delta^1 = [1, 6, 7, 2, 8, 3] \qquad \Delta^2 = [1, 7, 9, 3, 10, 5] \tag{11}$$

$$\Delta^3 = [1, 11, 9, 4, 12, 5] \quad \Delta^4 = [1, 6, 11, 2, 13, 4], \tag{12}$$

where the ordering from Eq. (10) is enforced.

Note that Fig. 2(b) was for $n = d = 2$. In general we get the following definition.

Definition 2 *(Continuous Piecewise-Polynomial on collection of simplices). Let K be a collection of m simplices and let V be an polynomial of degree d in n variables. Then*

$$V_i = \sum_{|\alpha|=d} CV_\alpha(\Delta^i)\mathcal{B}^d_\alpha = CV(\Delta^i)\mathcal{B}^d, \quad \forall i \in \{1, \cdots, m\}, \tag{13}$$

where CV are the coefficients and Δ^i is defined by the arbitrary ordering of the simplices and vertices, and the ordering of the grid points defined by Eq. (10)*.*

III. Lyapunov Criteria in Bernstein Basis

In [8] the Lyapunov criteria are translated to polynomials in the Bernstein basis. We first recall the stability result from Lyapunov and then show how the translation looks with the notation above.

Theorem 3 *(Local Stability) [13]. Let $\dot{x} = f(x)$ be an autonomous system with $f : U \to \mathbb{R}^n$ being a polynomial vector field, where $U \subseteq \mathbb{R}^n$ is open. If there exists a differentiable polynomial $V : U \to \mathbb{R}$ such that*

$$V(x) > 0 \quad \forall x \neq 0 \wedge V(0) = 0 \tag{14}$$

$$-\dot{V}(x) > 0 \quad \forall x \neq 0 \wedge \dot{V}(0) = 0, \tag{15}$$

with $\dot{V}(x) = \frac{\partial V(x)}{\partial x} f(x)$, then $x = 0$ is a local asymptotically stable equilibrium point of $f(x)$.

Since V is to be designed as a continuous piecewise-polynomial defined on simplices, which are closed, it has well defined differential quotients in the interior of the simplices, but it is (possibly) non-differential on the boundaries, i.e. on the faces. To address this we lean upon the Dini derivative and the unfamiliar reader is refereed to [14] and Thm. 8.2 therein.

Denoting, e.g., $\alpha_{(2,0,0)}$ by α_{de_0} where $d = 2$ and e_j is the j^{th} canonical unit vector, the vertices in a simplex are determined by α_{de_j} for $j \in \{0, 1, \cdots, n\}$. With this we obtain the following.

Lemma 4 *(Local Stability in Bernstein Basis) [8]. Let V be defined as in Definition 2 and let the Lie derivative $\dot{V}$ be defined by CL. Let*

$$CV \geq 0 \tag{16a}$$

$$CV_{\alpha_{de_0}}(\Delta^1) = 0 \tag{16b}$$

$$CV_{\alpha_{de_j}}(\Delta^i) > 0 \forall i \in \{1, \cdots, m\}, \forall j \in \{1, \cdots, n\} \tag{16c}$$

$$CL \leq 0 \tag{16d}$$

$$CL_{\hat{\alpha}_{\hat{d}e_0}}(\hat{\Delta}^1) = 0 \tag{16e}$$

$$CL_{\hat{\alpha}_{\hat{d}e_j}}(\hat{\Delta}^i) < 0 \forall i \in \{1, \cdots, m\}, \forall j \in \{1, \cdots, n\}, \tag{16f}$$

where we without loss of generality assumed the origin to be a vertex of σ^1, m is the number of simplices and $\hat{\alpha}$, $\hat{d}$, and $\hat{\Delta}$ are the multi index, the degree, and the grid points related to the Lie derivative. If such a V exists, then $x = 0$ is a local asymptotically stable equilibrium point of $f(x)$.

Regarding the Dini derivative, for practical proposes it simply amounts to having distinct coefficients on the common faces from all simplicies defined on the face.

The importance of this reformulation comes from the fact that the coefficient vector for the Lie derivative is linear in the unknown coefficients from the Lyapunov function, such that

$$CL = A\ CV, \tag{17}$$

where A is a matrix.

IV. Designing Positivity

It is, as noted in [8], possible to cast the Lyaounov synthesis problem as a linear feasibility program (LFP) if the vector field and the Lyapounov function are polynomials described in the Bernstein basis. In order to formulate the constraints we need to obtain two matrices which operate as a "differentiator" and a "multiplier". Multiplying these two matrices we obtain a matrix which, when multiplied by the vector of unknown Lyapunov coefficients, represents the operation of differentiation along trajectories. Combining this with Eq. (16) an LFP i obtained as

$$A\ CV \leq b_1 \qquad\qquad CV \geq b_2 \tag{18}$$

where CV is the vector of decision variables. The entries of the vector b_1 are either 0 or 1 depending on whether the corresponding row of A relates to Eq. (16d) or to Eq.

(16f). This is justified by the fact that Lyapunov functions are not unique with respect to multiplication with a positive constant. That is, if V is a Lyapunov function, then so is γV, $\gamma > 0$. The entries of the vector b_2 are determined likewise according to Eq. (16a) and (16c). Equation (16b) constitutes an equality constraint which is also to be included in the LFP. Equation (16e) is trivially fulfilled since $x = 0$ is the equilibrium under consideration. In the following we derive how to obtain the matrix A.

A. Converting Between Bases

Remembering Eq. (4), (5), and (6) any polynomial p is uniquely defined by its coefficient vector $b(p, D, \sigma)$. The same polynomial can also be described in the familiar monomial basis as

$$p = \sum_{k=1}^{K} C_k x^{\gamma_k} \tag{19}$$

where $C = [C_1 \ C_2 \ \cdots \ C_K]$ is the coefficient vector, $x \in \mathbb{R}^n$, and

$$\gamma = [\gamma_1 \ \gamma_2 \ \cdots \ \gamma_K], \ \gamma_i \in \mathbb{N}_0^n \tag{20}$$

$$x^{\gamma_k} = x_1^{\gamma_k(1)} x_2^{\gamma_k(2)} \cdots x_n^{\gamma_k(n)}. \tag{21}$$

The relationship between C and $b(p, D, \sigma)$ can be identified by expanding and equating terms. This is not treated further and the reader is referred to [15] for detail.

B. Partial Differentiation of an Unknown Polynomial

With the Lyapunov function and its partial derivative given by

$$V = \sum_{|\alpha_V| = d_V} CV_{\alpha_V} \mathcal{B}^{d_V}_{\alpha_V} = CV\mathcal{B}^{d_V} \tag{22}$$

$$\begin{aligned} \frac{\partial V}{\partial x} &= \sum_{|\tilde{\alpha}| = \tilde{d}} [Cd_1V_{\tilde{\alpha}}, \cdots, Cd_nV_{\tilde{\alpha}}] \mathcal{B}^{\tilde{d}}_{\tilde{\alpha}} \\ &= [Cd_1V, \cdots, Cd_nV] \mathcal{B}^{\tilde{d}}, \end{aligned} \tag{23}$$

with $\tilde{d} = d_V - 1$ and Cd_iV the coefficients of the partial derivative with respect to x_i, we want to find the matrices $B_{1,i}$ defined by

$$Cd_iV = B_{1,i} \ CV. \tag{24}$$

In [8] the partial derivative of the Bernstein basis polynomials is given. Using that and Eq. (22) the following is obtained:

$$\begin{aligned} Cd_iV_{\tilde{\alpha}} = [CV_{\tilde{\alpha}+e_0} \cdots CV_{\tilde{\alpha}+e_n}] \\ \begin{bmatrix} \sigma_0(1) & \sigma_1(1) & \cdots & \sigma_n(1) \\ \sigma_0(2) & \sigma_1(2) & \cdots & \sigma_n(2) \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_0(n) & \sigma_1(n) & \cdots & \sigma_n(n) \\ 1 & 1 & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} dI_{n\times n} \\ 0_{1\times n} \end{bmatrix}, \ \forall \tilde{\alpha}, \end{aligned} \tag{25}$$

where σ_i is the i^{th} vertex of the simplex σ. The vector of unknown Lyapunov coefficients is read in the following way. For a given $\tilde{\alpha}$, the $\tilde{\alpha} + e_j$ each corresponds to exactly one α_V. Because of this, each $Cd_iV_{\tilde{\alpha}}$ is a linear combination of $n+1$ of the unknown coefficients. Keeping this in mind and arranging the entries in $B_{1,i}$ accordingly, Eq. (24) is obtained. Doing this for all n variables identifies the n matrices corresponding to partial differentiation such that

$$[Cd_1V, \cdots, Cd_nV] = [B_{1,1}, \cdots, B_{1,n}]CV, \tag{26}$$

where $B_1 \in \mathbb{R}^{N_{\tilde{d}} \times N_{d_V} \times n}$.

C. Multiplication of a Known and an Unknown Polynomial

The way of describing a vector field in the Bernstein basis is straight forward. Using the description in Eq. (6) we get

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \cdots \\ x_n \end{bmatrix} &= \begin{bmatrix} f_1(x_1, \cdots, x_n) \\ \cdots \\ f_n(x_1, \cdots, x_n) \end{bmatrix} \\ &= \begin{bmatrix} \sum_{k=1}^{K} C_{1,k} x^{\gamma_k} \\ \cdots \\ \sum_{k=1}^{K} C_{n,k} x^{\gamma_k} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{|\alpha|=D} b_\alpha(f_1, D, \sigma) \mathcal{B}^D_\alpha \\ \cdots \\ \sum_{|\alpha|=D} b_\alpha(f_n, D, \sigma) \mathcal{B}^D_\alpha \end{bmatrix}. \end{aligned} \tag{27}$$

Note that in the description in the Bernstein basis, the degree is the same for all polynomials, even though the actual degree of the f_i's may differ.

Having the vector field described as in Eq. (27), and the partial derivative of the Lyapunov function as Eq. (23) we want to describe the Lie derivative

$$\dot{V}(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i \tag{28}$$

such that

$$\dot{V}(x) = \sum_{|\hat{\alpha}| = \hat{d}} CL_{\hat{\alpha}}(\dot{V}, \hat{d}, \sigma) \mathcal{B}^{\hat{d}}_{\hat{\alpha}} = CL\mathcal{B}^{\hat{d}} \tag{29}$$

with $\hat{d} = \tilde{d} + D$, and

$$CL = A_1 \ CV. \tag{30}$$

Since the Lie derivative in Eq. (28) is a sum over multiplications of two polynomials, we define an intermediate polynomial as $h_i = \frac{\partial V}{\partial x_i} f_i$. Note the following about h_i

$$d_{h_i} = \hat{d}, \qquad \alpha = \hat{\alpha} - \tilde{\alpha}, \tag{31}$$

$$|\hat{\alpha}| = |\tilde{\alpha} + \alpha|, \qquad |\alpha| = |\hat{\alpha} - \tilde{\alpha}|. \tag{32}$$

Using this, we want a matrix such that

$$Ch_i = C_{1,i} \ Cd_iV, \tag{33}$$

where Ch_i are the coefficients of h_i and $C_{1,i} \in \mathbb{R}^{N_{\hat{d}} \times N_{\tilde{d}}}$ is a matrix corresponding to multiplication of two polynomials. Writing out $\frac{\partial V}{\partial x_i} f_i$ and identifying the Bernstein basis

polynomials of the new degree, $\hat{d}$, the coefficients of h_i can be identified as

$$h_i = \sum_{|\tilde{\alpha}|=\tilde{d}} \sum_{|\alpha|=D} Cd_i V_{\tilde{\alpha}} b_\alpha(f_i, D, \sigma) \mathcal{B}^{\tilde{d}}_{\tilde{\alpha}} \mathcal{B}^{D}_{\alpha} \tag{34}$$

$$= \sum_{|\tilde{\alpha}|=\tilde{d}} \sum_{|\alpha|=D} Cd_i V_{\tilde{\alpha}} b_\alpha(f_i, D, \sigma) \frac{\binom{\tilde{d}}{\tilde{\alpha}}\binom{D}{\alpha}}{\binom{\hat{d}}{\hat{\alpha}}} \underbrace{\binom{\hat{d}}{\hat{\alpha}} \lambda^{\tilde{\alpha}+\alpha}}_{\mathcal{B}^{\hat{d}}_{\hat{\alpha}}}$$

$$= \sum_{|\hat{\alpha}|=\hat{d}} Ch_{i,\hat{\alpha}} \mathcal{B}^{\hat{d}}_{\hat{\alpha}} = Ch_i \mathcal{B}^{\hat{d}} \tag{36}$$

with

$$Ch_{i,\hat{\alpha}} = \sum_{\substack{|\hat{\alpha}-\tilde{\alpha}|=D \\ \hat{\alpha}-\tilde{\alpha}\geq 0}} Cd_i V_{\tilde{\alpha}} b_{\hat{\alpha}-\tilde{\alpha}}(f_i, D, \sigma) \frac{\binom{\tilde{d}}{\tilde{\alpha}}\binom{D}{\hat{\alpha}-\tilde{\alpha}}}{\binom{\hat{d}}{\hat{\alpha}}}. \tag{37}$$

The summation is read as follows. For each $\hat{\alpha}$ the summation variable is $\tilde{\alpha}$ and each $Ch_{i,\hat{\alpha}}$ is a linear combination of the unknown $Cd_i V_{\tilde{\alpha}}$ for which $|\hat{\alpha}-\tilde{\alpha}| = D$ and $\hat{\alpha}-\tilde{\alpha} \geq 0$ are fulfilled. Keeping this in mind and distributing the scaled coefficients $b(f_i, D, \sigma)$ in the entries of $C_{1,i}$ accordingly, Eq. (33) is obtained.

D. Combining the Results

Using the above the Lie derivative can be expressed as

$$\dot{V}(x) = (Ch_1 + \cdots + Ch_n)\mathcal{B}^{\hat{d}} \tag{38}$$

$$= (C_{1,1}B_{1,1} + \cdots + C_{1,n}B_{1,n})CV\mathcal{B}^{\hat{d}}. \tag{39}$$

Note here, that the summation makes sense since all matrices are of the same dimensions because all polynomials in the vector field were represented in the the same degree, D, regardless of the possibility of them not being of the same degree. Comparing Eq. (29) and (39) we get

$$CL = A_1\ CV \tag{40}$$

as wanted. This makes $A_1 \in \mathbb{R}^{N_{\hat{d}} \times N_{d_V}}$ a matrix which corresponds to taking the Lie derivative of f on σ. For the Lie derivative on the collection of simplices K, we simply stack the matrices like

$$A = \begin{bmatrix} A_1 \\ \cdots \\ A_m \end{bmatrix}, \tag{41}$$

such that A_1 relates to σ^1, A_2 relates to σ^2 and so on.

E. Triangulation

Before stating the algorithm for solving the synthesis problem we need to obtain the collection of simplices. This is done by triangulation. With $B_n \subset \mathbb{R}^n$ being an arbitrary n-dimensional box with the origin in its interior, we want a collection of simplices K, where the corners of B_n and the origin are the vertices of the simplices, such that

$$|K| = \bigcup_{\sigma \in K} \sigma = B_n, \tag{42}$$

where $|K|$ is the realisation of K [11]. This is obtained by performing Kuhn's triangulation [9] on the $2n$ facets of B_n and obtaining $2n!$ $(n-1)$-simplices. Adding the origin to each of them results in $2n!$ n-simplices and the collection fulfils Eq. (42).

F. BBAlgorithm

Utilizing the above, this section states the algorithm for solving the synthesis problem.

Algorithm 5 *(BBAlgorithm)*
Input: *C and γ defining the dynamical system, domain of definition B, degree of the Lyapunov function d_V, and maximal number of iterations $k_{\max}$.*
Output: *Triangulation of B and Lyapunov function V defined on the resulting simplices.*
Procedure:

0) *Set iteration counter $k = 1$ and get initial $K^{\{k\}}$.*
1) *Define V on $K^{\{k\}}$ and calculate A, b_1, and b_2 in Eq. (18).*
2) *If possible, solve the LFP. Terminate if feasible.*
3) *If $k < k_{\max}$ refine the triangulation by sub-division of (some of) the simplices. Set $k = k+1$ and return to 1.*

How to sub-divide (intelligently) when infeasibility occurs is an open question. In the next section three examples are given on how the BBAlgorithm works and we present a new way to sub-divide by means of an example.

The BBAlgorithm has been implemented in software for MATLAB. It can be freely downloaded from the CodeMe webpage: http://kom.aau.dk/project/CodeMe/index.html.

V. Synthesizing Examples

In this section three examples are considered. First a simple example to get familiar with the BBAlgorithm. Next, a comprehensive example shows how quickly the resulting LFP grows. The section ends with a problematic example, which details some drawbacks of the BBAlgorithm and presents a new way to overcome some of them.

A. Simple Example

We consider the reverse-time Van der Pol oscillator given in monomial basis as

$$C = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 1 & -1 \end{bmatrix} \qquad \gamma = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{43}$$

according to Eq. (19) and (27). We want to show that the system is locally stable by considering the box $B = [-1, 1]^2$, with the origin as the equilibrium point. To this end we search for an unknown Lyapunov function of degree $d_V = 3$. These 4 inputs are fed to the BBAlgorithm, and it returns that the LFP is feasible in one iteration. Figure 3 shows the Lyapunov function. The solid lines are the facets of the simplices. The crosses are the control points, i.e. the grid point and the coefficient at that point. A blue cross indicates that the coefficient is equal to zero, a green cross indicates a coefficient being positive. Figure 4 shows the Lie derivative. The Dini derivative is discontinuous on the facets

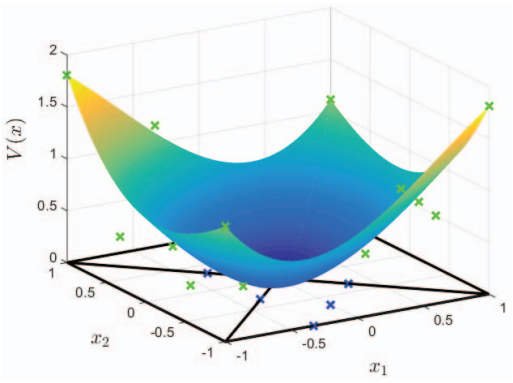


Fig. 3. Lyapunov function synthesised for the Van der Pole oscillator.

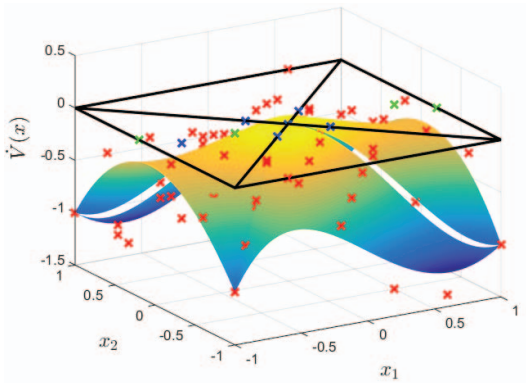


Fig. 4. Lie derivative of the Lyapunov function in the simple example.

even though the Lyapunov function is continuous across the facets. However, since all coefficients are negative the Lie derivative is in accordance with Eq. (16d). A red cross indicates a negative coefficient. Note that three coefficients are displayed as green, i.e. positive. Their values, however, are in the magnitude of 10^{-16} and depend on the accuracy of the solver.

B. Comprehensive Example

In [7] they design non-trivial stable polynomials vector fields of increasing difficulty. They have three different realizations of their approach, each with increasing complexity. When applied on their Benchmark #13, the two first experience numerical problems, whereas the third one is terminated after 20 minutes without reaching a conclusion.

Benchmark #13 is in 4 variables with a maximal monomial degree of 6. Across the four polynomials the vector field has a total of 161 different monomial combinations. When trying to solve the synthesis problem for this system using the BBAlgorithm on an everyday laptop, MATLAB returns an *out of memory* message after some time. In order to test the capability of the BBAlgorithm the problem was fed to a somewhat more capable setup. Using an average of 45 GB RAM the constraints were calculated in 38 minutes and the LFP was solved in one iteration after 80 hours of runtime. It had 3,697 variables and 65,520 constraints. The synthesised Lyapunov function was of degree 6 and defined on $2{\cdot}4! = 48$ simplices.

Off cause, such a brute force approach is in general not applicable since one does not know if the system is in fact stable. However, we wanted to test the BBAlgorithm.

C. Problematic Example

This example is from [16] and given as

$$C = \begin{bmatrix} -1 & 2 & -1 & 4 & -8 & 4 & \\ 0 & 0 & 0 & 0 & -9 & 10 & \\ -1 & 4 & 0 & -4 & 0 & 10 & 0 \\ 0 & 2 & -8 & -4 & -1 & 4 & -4 \end{bmatrix} \tag{44}$$

$$\gamma = \begin{bmatrix} 3 & 3 & 3 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 2 & 1 & 0 & 4 & 3 & 2 & 0 & 3 & 2 & 1 \end{bmatrix}.$$

We want to test if the system is stable on the box $B = [-1.75, 1.75]^2$. Using a Lyapunov function of degree $d_V = 2$ the LFP in the first iteration is returned as infeasible. This means one of two things. Either the system is in fact stable, but the current triangulation and degree is not sufficient to render the LFP solvable. Or the system is in fact not stable, and no choice of triangulation and degree will ever result in a solvable LFP. If one believes the system to be stable, two methods of modifying the LFP can be employed. Raising the degree of the Lyapunov function will increase the number of decision variables in the LFP as the number of coefficients in each simplex increases according to Eq. (9). Using subdivision of (some of) the simplices also increases the number of decision variables. It is desired to obtain a solvable LFP containing as few decision variables as possible.

It is important to realise a particular feature of the LFP obtained in this work. The decision variables and the constraints have geometrical meaning since they are all related to unique points in space, according to Eq. (7). Increasing the number of decision variables will increase the number of constraints and refine the location of these points. In light of this, raising the degree or increasing the number of simplices differ on a crucial thing. Raising the degree will result in a refinement of the location of decision variables and constraints on the entire domain. Sub-dividing offers the option of only refining the location of decision variables and constraints on part of the domain. For this reason, sub-dividing offers an intuitive advantage. It can zoom in on the problematic parts of the domain, so to say.

In order to do so we propose to add a slack variable to each simplex and allow for constraint violation on each simplex. By minimizing the sum of the slack variables, a linear minimization problem (LMP) is obtained as

$$\begin{aligned} \min_{CV, s_i} \quad & \sum_{i=1}^{m} s_i \\ \text{s.t.} \quad & A\,CV - \begin{bmatrix} \mathbf{1} & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & \mathbf{1} \end{bmatrix} \begin{bmatrix} s_1 \\ \cdots \\ s_m \end{bmatrix} \leq b_1 \\ & CV \geq b_2 \\ & s_i \geq 0, \end{aligned} \tag{45}$$

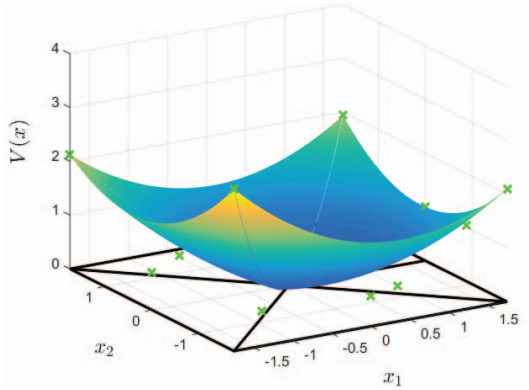


Fig. 5. Lyapunov function synthesised for the problematic example.

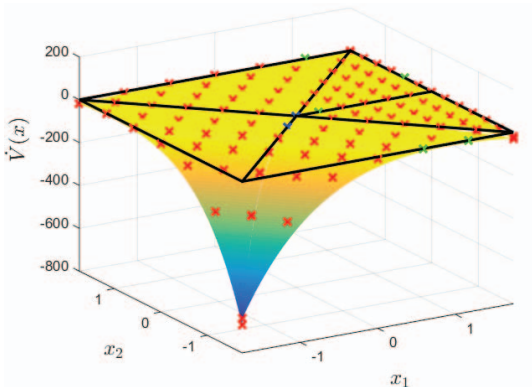


Fig. 6. Lie derivative of the Lyapunov function in the problematic example.

where s_i are the slack variables, m is the number of simplices, and $\mathbf{1}$ is an $N_{\hat{d}} \times 1$ vector of ones. This enables an identification of the simplices which need to be sub-divided. If all the slack variables are equal to zero the original LFP is solvable, and a Lyapunov function is found.

Using this the slack variables becomes $s_1 = s_3 = s_4 = 0$ and $s_2 > 0$. Four constraints are violated by s_2. They are located on the facets of σ^2 such that two of them are located on $\delta_0\sigma^2$, one on $\delta_1\sigma^2$, and one on $\delta_2\sigma^2$. This identifies the need for sub-division in simplex σ^2.

Applying binary splitting [9] to sub-divide σ^2 the new collection results in a solvable LFP. The synthesised Lyapunov function and the Lie derivative are shown in Fig. 5 and 6. In particular, note on Fig. 6 how the control points in the two small simplices are more dense than in the three big.

In total, there are 15 decision variables in the LFP in the second iteration. If instead, had we used a Lyapunov function of degree $d_V = 3$ the LFP would also be solvable, but with 24 decision variables. Thus zooming in did indeed save on the number of variables.

The synthesis problem can also be cast to search for a continuous differentiable Lyapunov function. In this case there are 14 variables in the LFP, but the degree of the Lyapunov function needs to be $d_V = 4$, see [16] for a discussion on this. On the contrary, if the Lyapunov criteria are embedded as SOS conditions using Putinar's Positivstellensatz, the resulting SDP has 21 decision variables. Thus, for this specific problem, the proposed algorithm has fewer decision variables than the SOS approach, and it results in an LFP rather than an SDP. Code behind this can be found on the CodeMe webpage: http://kom.aau.dk/project/CodeMe/index.html.

VI. Conclusion

In this paper an algorithm for the synthesis of Lyapunov functions was outlined. Given an arbitrary polynomial vector field, the BBAlgorithm searches for a continuous piecewise-polynomial Lyapunov function. The problem boils down to a linear feasibility problem which can then be solved by standard LP solvers.

When used in examples the BBAlgorithm showed its usefulness, although the number of decision variables and constraints grow rapidly with increase in degree and dimension of the vector field. When infeasibility occurs, a new way of modifying the problem was presented. Improving this technique, and further studying the advantages and disadvantages of searching for either continuous piecewise-polynomial or continuous differentiable polynomial Lyapunov functions is the focus of current work.

References

[1] D. Henrion, J.-B. Lasserre, and J. Löfberg, "GloptiPoly 3: Moments, Optimization and Semidefinite Programming." *Optimization Methods and Software*, vol. 24, pp. 761–779, 2009.

[2] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, `http://arxiv.org/abs/1310.4716`, 2013, available from `http://www.eng.ox.ac.uk/control/sostools`, `http://www.cds.caltech.edu/sostools` and `http://www.mit.edu/~parrilo/sostools`.

[3] A. A. Ahmadi and G. Hall, "Sum of Squares Basis Pursuit with Linear and Second Order Cone Programming," *Contemporary Mathematics*, pp. 1–26, 2015.

[4] F. Caruso, M.-F. Roy, and F. Boudaoud, "Certificates of Positivity in the Bernstein Basis," *Springer*, vol. 39, 2007.

[5] C. Muñoz and A. Narkawicz, "Formalization of Bernstein Polynomials and Applications to Global Optimization," *Journal of Automated Reasoning*, vol. 51, no. 2, pp. 151–196, 2013.

[6] P. S. V. Nataraj and M. Arounassalame, "A New Subdivision Algorithm for the Bernstein Polynomial Approach to Global Optimization," *International Journal of Automation and Compting*, vol. 4, no. 4, pp. 342–352, 2007.

[7] M. A. B. Sassi and S. Sankaranarayanan, "Linear relaxations of polynomial positivity for polynomial Lyapunov function synthesis," *Journal of Mathematical Control and Information*, pp. 1–34, 2015.

[8] C. Sloth and R. Wisniewski, "Robust stability of switched systems," *CDC*, 2014.

[9] R. Leroy, "Certificates of positivity in the simplicial Bernstein basis," *hal-00589945*, 2011.

[10] C. de Boor, "B-form basics," *Geometric Modeling*, pp. 131–148, 1987.

[11] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in Real Algebraic Grometry*. Springer, 2006, vol. 10.

[12] R. T. Farouki, "The bernstein polynomial basis: A centennial retrospective," Department of Mechanical and Aerospace Engineering, University of California, Tech. Rep., 2012.

[13] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 1996, vol. 3.

[14] G. V. Smirnov, *Introduction to the Theory of Differential Inclusions*, ser. Graduate Studies in Mathematics. American Matgematical Society, 2002, vol. 41.

[15] J. Berchtold and A. Bowyer, "Robust artihmetic for multivariate Bernstein-form polynomials," *Computer-Aided Design*, no. 32, pp. 681–689, 2000.

[16] A. A. Ahmadi and P. A. Parrilo, "Converse results on existence of sum of squares lyapunov functions," *Proceedings of the Conference on Decision and Control*, pp. 6516–6521, 2011.