Closed-Form Expressions of Convex Combinations

Bastian Schürmann, Ahmed El-Guindy, and Matthias Althoff

Abstract— **In this paper, the computation of closed-form convex combinations is considered. In many control tasks, convex combinations play a crucial role, thus requiring an efficient computation. This is the case for online control of fast dynamical systems, in which the control algorithms rely on convex combinations, for example robust control of linear parameter-varying systems. On the other hand, for formal verification, it is necessary that the closed-loop behavior can be expressed in closed-form, which is not possible if the convex combination is expressed as constraints. In this paper, we provide closed-form expressions for any kind of polytope with finitely many extreme points. For special types of polytopes, such as simplices and parallelotopes, we provide especially efficient, analytical closed-form expressions. Numerical experiments show that the closed-form expressions are significantly faster in all randomly-generated cases and thus enable shorter sampling intervals of control schemes involving convex combinations.**

I. Introduction

Convex combinations are used in many different applications in which all points in a set must be expressed as combinations of extreme points. For example, convex combinations are very useful in the robust control of linear parameter varying (LPV) systems. If an LPV system acts in uncertain environments and is affected by external and internal disturbances, the control algorithm must take those uncertainties and disturbances into account. For example, reusable launch vehicles, upon reentry, as discussed in [1], have to deal with many disturbances due to wind and electromagnetic interferences, as well as aerodynamic and atmospheric parameter uncertainties.

For the aforementioned reasons, robust control methods like $\mathcal{H}_\infty$ or $\mathcal{H}_2$ control are introduced for LPV systems. For a linear system with fixed parameters, a set of linear matrix inequalities (LMIs) which depends on the system matrices is solved, and robust control can be achieved. An LPV system, however, does not have constant system matrices which can be used for the LMI approach. Instead, there are infinitely many matrices which depend on the parameters, which makes the robust control task for LPV systems challenging. However, there exists a promising gain-scheduling approach [2], [3] for LPV where the system matrices depend affinely on the parameter vector θ, where θ can be measured or estimated in real time. The basic assumption of this approach is that the parameters can vary in a given polytope described by its extreme points. Instead of trying to find a fixed robust controller for the whole parameter space, the authors in [2] propose computing robust $\mathcal{H}_\infty$ controllers for the extreme points of the parameter polytope only. These controllers for the extreme points are obtained by solving a set of LMIs along the lines of typical robust controller design. One of the novel parts of the approach is that the authors obtain the controller for any parameter θ by using a convex combination of the controllers of the extreme points.

This controller approach was proposed for various applications, including missile control [4], multi-propeller airships [5], and the previously mentioned reusable launch vehicles [1]. It was also proposed for several wind power applications, including wind energy conversion systems [6], wind turbines [7], and wind power generators [8]. Its applicability was proven by several implementations on real hardware setups, including electromagnetic actuators [9], robotic manipulators [10], and winding systems for elastic webs [11]. In [12] the applicability of the LPV approach was used for model predictive control, where the nonlinear model was linearized using LPV models. This control approach strongly depends on the ability to express a point inside a polytope as a convex combination of the extreme points. If this is done by solving a linear program, the computation is time-consuming and prevents the use of formal methods, as they rely on the closed-form expressions of the closed-loop dynamics. One can overcome these problems by expressing the convex combination in closed-form.

In the literature, closed-form expressions are only mentioned for special cases, for example for two-dimensional, axis-aligned boxes in [2], [10], [11], [7], and [12] or even axis-aligned boxes of arbitrary dimension in [1] and [8]. However, none of these papers provide proofs of the formulas. All of these sources, even [1] and [8], who provide a formula for $n-$dimensional, axis-aligned boxes, discuss their theory for general polytopes but are only able to provide formulas for special forms of boxes. Other papers, for example [13], circumvent the problem of obtaining closed-form expressions by simply assuming that the convex combinations can be obtained in real-time.

The purpose of this paper is to close this gap by providing closed-form expressions of convex combinations for general polytopes in order to reduce online computation times and therefore enable the use of the previously discussed control approaches for fast dynamical systems. At the same time, this provides the required tools for formal verification in order to prove safety and other properties before runtime by e.g. using reachability analysis [14], [15]. In order to apply reachability analysis to controllers using convex combinations, these techniques need closed-form expressions of the closed-loop dynamics, and therefore also of the convex combinations.

In this paper, we refer to an expression which can be

The authors are with the Department of Informatics, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany. bastian.schuermann@tum.de, ahmed.elguindy@tum.de, althoff@in.tum.de

written as an explicit mathematical formula, i.e., without any loops or iterations, as analytical closed-form expressions. For general closed-form expressions, we allow that there are finitely many case distinctions, but each case must only consist of an analytical closed-form expression. While it is not possible to provide an analytical closed-form expression for every type of set, we discuss how such an expression is possible for certain classes of sets, more precisely simplices and parallelotopes. Moreover, we show how non-analytical closed-form expressions can be obtained even for arbitrary polytopes with finitely many extreme points. The latter case involves solving point location problems, which makes it non-analytical.

The remainder of this paper is organized as follows: We begin with a motivating example in Sec. II. In Sec. III we provide a summary of the notation and give some definitions of important concepts. Sec. IV is the main section and contains the closed-form expressions for simplices, parallelotopes, and arbitrary polytopes. The usefulness of these expressions is validated by randomly-generated numerical examples in Sec. V, where our results are compared with solving the convex decomposition problem using linear programming. The paper concludes with a discussion of the results in Sec. VI.

II. Motivating Example

To motivate the usefulness of closed-form expressions of convex combinations, let us consider the robust, gain-scheduled controller approach of [2] for LPV systems of the form

$$\begin{aligned}\dot{x} &= A(\theta(t))x + B(\theta(t))u,\\ y &= C(\theta(t))x + D(\theta(t))u.\end{aligned}$$

The proposed controller synthesis algorithm works as follows: First, the robust controllers for the p extreme points $\hat{\theta}^{(i)}, i \in \{1,\dots,p\}$, of the parameter polytope are computed offline and denoted by $K^{(1)},\dots,K^{(p)}$. When applying the controller online, the current parameter θ is measured and the coefficients $\lambda_i(\theta)$ of the convex combination of the extreme points are computed such that

$$\sum_{i=1}^{p} \lambda_i(\theta)\hat{\theta}^{(i)} = \theta, \quad \lambda_i(\theta) \geq 0, \quad \sum_{i=1}^{p} \lambda_i(\theta) = 1. \qquad (1)$$

The same coefficients $\lambda_i(\theta)$ are then used to obtain the corresponding controller K by computing it as a convex combination of the extreme point controllers with the same coefficients as before, i.e., $K = \sum_{i=1}^{p} \lambda_i(\theta)K^{(i)}$.

Using this technique, robust controllers for any of the infinitely many parameters θ in the parameter polytope can be obtained, while the robust controller synthesis has to be performed for the finitely many extreme points only. Although this shows the usefulness of this control approach, it has one problem which limits its real-time application for fast systems. Since the controllers are not known explicitly, but obtained using a convex combination, a convex decomposition problem has to be solved online in every time step. This means that every time we want to find the combination $\lambda_i(\theta)$ for a given parameter θ, we have to solve the system of inequalities in (1). These are $n+1$ equalities and p inequalities for p unknowns. In particular for high-dimensional polytopes with many extreme points, this leads to a large system of inequalities, which is hard to solve. Moreover, with the exception of simplices, this system of inequalities might have non-unique solutions. If the convex combinations are only written as (1), we do not have a closed-form expression for every point.

III. Background

In this section, we provide some definitions and recapture some basics.

A. Notation

We denote by $\mathbb{R}$ the set of real numbers and by $\emptyset$ the empty set. For two real numbers $a, b \in \mathbb{R}$, with $a \leq b$, we denote by $[a,b]$ and (a,b) the closed and open set between a and b, respectively. For a vector $x \in \mathbb{R}^n$ we denote by x^T its transpose, and we refer to its elements with $x_i \in \mathbb{R}$. We denote the identity matrix by $I \in \mathbb{R}^{n\times n}$ and the unit vectors by $e^{(i)}$, i.e., $e^{(i)}$ is the $i-$th column of I. The vector containing all ones is denoted by $\mathbf{1}$, and the vector containing only zeros is referred to as $\mathbf{0}$. The boundary of a set $\mathcal{S} \subset \mathbb{R}^n$ is denoted by $\mathcal{B}(\mathcal{S})$, and we write $\mathcal{I}(\mathcal{S})$ for the interior of $\mathcal{S}$, such that $\mathcal{S} = \mathcal{B}(\mathcal{S}) \cup \mathcal{I}(\mathcal{S})$.

B. Convexity

Let us recall some basics about convex combinations and convex set [16]. A vector $x \in \mathbb{R}^n$ is a convex combination of p given vectors $\hat{x}^{(1)},\dots,\hat{x}^{(p)} \in \mathbb{R}^n$ if x can be written as

$$x = \sum_{i=1}^{p} \lambda_i(x)\hat{x}^{(i)}, \quad \lambda_i(x) \geq 0, \quad \sum_{i=1}^{p} \lambda_i(x) = 1.$$

A set $\mathcal{S} \subset \mathbb{R}^n$ is convex if any convex combination of two points $x^{(1)}, x^{(2)} \in \mathcal{S}$ lies inside the set, i.e., $\forall \lambda \in [0,1]$: $\lambda x^{(1)} + (1-\lambda)x^{(2)} \in \mathcal{S}$. If $\mathcal{S}$ is convex and compact[1] then we call a point $\hat{x}^{(0)} \in \mathcal{S}$ an extreme point of $\mathcal{S}$ if it cannot be represented by a convex combination of any two points $\hat{x}^{(1)}, \hat{x}^{(2)} \in \mathcal{S}, \hat{x}^{(1)} \neq \hat{x}^{(0)}, \hat{x}^{(2)} \neq \hat{x}^{(0)}$. For any set $\mathcal{S} \subset \mathbb{R}^n$, its convex hull $\mathbf{conv}(\mathcal{S})$ is the set of all convex combinations of points in $\mathcal{S}$, i.e.

$$\mathbf{conv}(\mathcal{S}) = \left\{ \sum_{i=1}^{p} \lambda_i \hat{x}^{(i)} | \hat{x}^{(i)} \in \mathcal{S}, \lambda_i \geq 0, \sum_{i=1}^{p} \lambda_i = 1 \right\}.$$

C. Special Forms of Convex Sets

A set is called a parallelotope if it can be written as

$$\mathcal{P} = \{x \in \mathbb{R}^n | x = c_{\mathcal{P}} + G_{\mathcal{P}}\alpha, \alpha_i \in [-1,1]\},$$

with $\alpha \in \mathbb{R}^n$. Therein $c_{\mathcal{P}} \in \mathbb{R}^n$ defines the center point of the parallelotope, and the matrix $G_{\mathcal{P}} \in \mathbb{R}^{n\times n}$ has full rank and contains the n generators as its columns.

[1]A set is compact if it is closed and bounded.

The convex hull of p extreme points $\hat{x}^{(1)}, \ldots, \hat{x}^{(p)}$ is called a polytope P, i.e., it can be written as

$$P = \mathbf{conv}(\{\hat{x}^{(1)}, \ldots, \hat{x}^{(p)}\}).$$

We call a polytope $P \subset \mathbb{R}^n$ with $n+1$ extreme points a simplex, if n of those extreme points are linearly independent.

IV. Closed-Form Expression of Convex Combinations

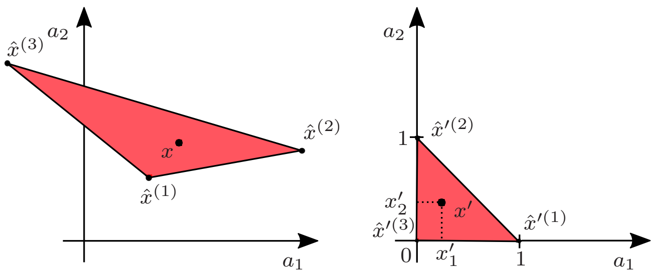
As discussed in Sec. II, when using convex combinations for control or other online computation tasks, we have to find the parameters $\lambda_i(x)$ which are used to express a point x in a polytope P as a convex combination of the p extreme points $\hat{x}^{(i)}$ of P in real-time. If we do not want to solve the convex decomposition problem (1) online in every time step, we need closed-form expressions of the convex combinations. In this section we provide such closed-form expressions. We do this first for simplices and parallelotopes, for which we can provide analytical closed-form expressions. After this we present non-analytical closed-form expressions for arbitrary polytopes.

A. Simplices

Simplices are polytopes with the least number of extreme-points, which still span a full-dimensional subset in $\mathbb{R}^n$. The convex combination of the extreme points for a given point inside a simplex is unique. We show that any simplex can be transformed into the unit simplex, i.e., the simplex with one extreme point equal to the origin and the others equal to the unit vectors $e^{(i)}$. This is achieved using an affine transformation, i.e., a linear transformation together with a translation. Both operations do not change any convexity properties [16]. We denote the affine transformation of a point y by y', i.e., $y' = G^{-1}\left(y - \hat{x}^{(n+1)}\right)$. The new coordinates y' are known as barycentric coordinates [17]. By subtracting $\hat{x}^{(n+1)}$, we shift the simplex such that the transformed extreme point $\hat{x}'^{(n+1)}$ lies in the origin. By multiplying $\left(y - \hat{x}^{(n+1)}\right)$ with G^{-1}, we transform the other extreme points to the unit vectors, i.e., $\hat{x}'^{(i)} = e^{(i)}$ for $i \in \{1, \ldots, n\}$. Note that since $\hat{x}^{(i)}$ are all extreme points, the vectors $\hat{x}^{(i)} - \hat{x}^{(n+1)}, i \in \{1, \ldots, n\}$, are all linearly independent and therefore G^{-1} exists. For this special simplex, we show how a unique, analytical closed-form expression can be obtained. The advantage of this result is that the convex combination obtained in this way can be used for the original, non-unit simplex. This is stated in the following theorem and is illustrated in Fig. 1:

Theorem 1 *We consider an arbitrary simplex $\mathcal{S} \subset \mathbb{R}^n$ described by its $n+1$ extreme points $\hat{x}^{(1)}, \ldots, \hat{x}^{(n+1)}$. Let us define the matrix*

$$G = \left[\hat{x}^{(1)} - \hat{x}^{(n+1)}, \ldots, \hat{x}^{(n)} - \hat{x}^{(n+1)}\right].$$



(a) Original simplex (b) Simplex after affine transformation

Fig. 1. Any simplex (a) can be transformed to the unit simplex (b), for which the closed-form decomposition problem can easily be solved.

Given a point $x \in \mathcal{S}$, this point can be expressed as a convex combination of the extreme points as

$$x = \sum_{i=1}^{n+1} \lambda_i(x)\hat{x}^{(i)}, \tag{2}$$

where the parameters $\lambda_i(x)$ are given by the following closed-form expression:

$$\lambda_i(x) = \left(G^{-1}\left(x - \hat{x}^{(n+1)}\right)\right)_i \quad \text{for } i \in \{1, \ldots, n\} \tag{3}$$

$$\lambda_{n+1}(x) = 1 - \sum_{i=1}^{n} \lambda_i(x). \tag{4}$$

Proof: We have to show that the two statements of the theorem hold: (2) is a convex combination and (2) with $\lambda_i(x)$ resulting from (3)-(4) actually describes the point x.

Since the transformed extreme points $\hat{x}'^{(i)}$ are the unit vectors $e^{(i)}$, any transformed x', with $x \in \mathcal{S}$, can be uniquely written as

$$x' = \sum_{i=1}^{n} x'_i e^{(i)} = \sum_{i=1}^{n} x'_i \hat{x}'^{(i)},$$

see also Fig. 1. Adding the last transformed extreme point $\hat{x}'^{(n+1)} = \mathbf{0}$ does not change the result, since it is the origin. From the definition of the simplex as the convex hull of the extreme points, it follows that for any $x \in \mathcal{S}$, $\sum_{i=1}^{n} \lambda_i(x') = \sum_{i=1}^{n} x'_i \leq 1$, because $\lambda_i(x') = x'_i, i \in \{1, \ldots, n\}$ and x' is inside the unit simplex. Since $\sum_{i=1}^{n} \lambda_i(x') \leq 1$ is not necessarily a convex combination, as the sum might be smaller than one, we define $\lambda_{n+1}(x') = 1 - \sum_{i=1}^{n} \lambda_i(x')$. We use this weight to add $\hat{x}'^{(n+1)} = \mathbf{0}$, which does not change the result of the original decomposition, i.e.,

$$\begin{aligned} x' &= \sum_{i=1}^{n+1} \lambda_i(x')\hat{x}'^{(i)} = \sum_{i=1}^{n} \lambda_i(x')\hat{x}'^{(i)} + \lambda_{n+1}(x')\mathbf{0} \\ &= \sum_{i=1}^{n} \lambda_i(x')\hat{x}'^{(i)}. \end{aligned}$$

However, this definition of $\lambda_{n+1}(x')$ ensures that $\sum_{i=1}^{n+1} \lambda_i(x') = 1$.

Because $x \in \mathcal{S}$ and $\mathcal{S}$ is transformed to the unit simplex, all points are mapped to a nonnegative set. Therefore it holds that $\lambda_i(x') = x'_i \geq 0, i \in \{1, \ldots, n\}$. From $\sum_{i=1}^{n} \lambda_i(x') =$

$\sum_{i=1}^{n} x'_i \leq 1$ it follows that $\lambda_{n+1}(x') = 1 - \sum_{i=1}^{n} \lambda_i(x') \geq 0$. Therefore, the $\lambda_i(x')$ are a convex combination and define any point $x' \in \mathcal{S}'$ in terms of the transformed extreme points $\hat{x}'^{(i)}$.

In the last step, we show that we can use the parameters of the transformed states $\lambda_i(x')$ for the original state x as well, by transforming the point x' back into the original coordinates. Since $x' = G^{-1}\left(x - \hat{x}^{(n+1)}\right)$ by definition, the following holds:

$$x = Gx' + \hat{x}^{(n+1)} = G\left(\sum_{i=1}^{n+1} \lambda_i(x')\hat{x}'^{(i)}\right) + \underbrace{\hat{x}^{(n+1)}}_{=\sum_{i=1}^{n+1} \lambda_i(x')\hat{x}^{(n+1)}}$$

$$= \sum_{i=1}^{n+1} \lambda_i(x')\left(G\hat{x}'^{(i)} + \hat{x}^{(n+1)}\right) = \sum_{i=1}^{n+1} \lambda_i(x')\hat{x}^{(i)}, \tag{5}$$

which is a direct result from the fact that the convexity properties are not changed by affine transformations. Therefore, x can be written as a convex combination of the extreme points of $\mathcal{S}$, and it has the same parameters $\lambda_i(x')$ as the convex combination of x'. ■

B. Parallelotopes

In the previous subsection, our solution involved transforming simplices to unit simplices, for which the analytical closed-form expression is easy to obtain. In this section, we present a similar approach for parallelotopes. We first transform a general parallelotope to the unit hypercube $[0,1] \times [0,1] \times \dots [0,1]$ and then present a way to obtain a closed-form expression for this special class of parallelotopes. The advantage is that in the case of boxes whose edges are parallel to the axes, we can consider each dimension independently. Although there are infinitely many different combinations for choosing $\lambda_i(x)$, except for points on the boundary, we reduce the degrees of freedom such that we obtain a unique solution. We do this by defining for any point $x \in \mathcal{P}$

$$x' = \frac{1}{2}G_{\mathcal{P}}^{-1}\left(x - c_{\mathcal{P}}\right) + \frac{1}{2}\mathbf{1}, \tag{6}$$

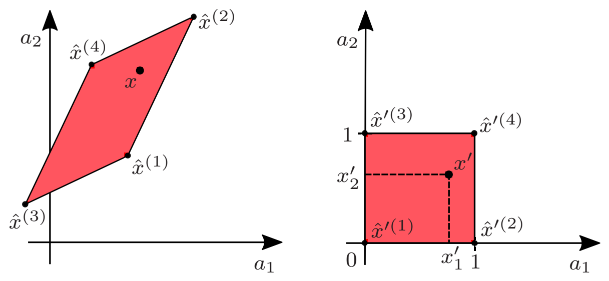
where $c_{\mathcal{P}}$ is the center point and $G_{\mathcal{P}}$ is the generator matrix of $\mathcal{P}$. This affine transformation maps the parallelotope to

$$\begin{aligned}\mathcal{P}' &= \frac{1}{2}G_{\mathcal{P}}^{-1}\left(\mathcal{P} - c_{\mathcal{P}}\right) + \frac{1}{2}\mathbf{1} \\ &= \{x' \in \mathbb{R}^n | x' = \frac{1}{2}G_{\mathcal{P}}^{-1}\left(c_{\mathcal{P}} - c_{\mathcal{P}}\right) + \frac{1}{2}G_{\mathcal{P}}^{-1}G_{\mathcal{P}}\alpha + \frac{1}{2}\mathbf{1}, \\ &\qquad \alpha_i \in [-1,1]\} \\ &= \{x' \in \mathbb{R}^n | x' = \frac{1}{2}\mathbf{1} + \frac{1}{2}I\alpha, \alpha_i \in [-1,1]\},\end{aligned}$$

which is the unit hypercube. Note that $G_{\mathcal{P}}^{-1}$ always exists since $G_{\mathcal{P}}$ has full rank. The transformation for an example in $\mathbb{R}^2$ is visualized in Fig. 2.

Theorem 2 *We consider an arbitrary parallelotope $\mathcal{P} \subset \mathbb{R}^n$ given by*

$$\mathcal{P} = \{x \in \mathbb{R}^n | x = c_{\mathcal{P}} + G_{\mathcal{P}}\alpha(x), \alpha_i(x) \in [-1,1]\},$$



(a) Original parallelotope

(b) Parallelotope after affine transformation

Fig. 2. Any parallelotope (a) can be transformed to the unit hypercube (b), for which the closed-form decomposition problem can easily be solved.

with 2^n extreme points $\hat{x}^{(1)}, \dots, \hat{x}^{(2^n)}$. Given a point $x \in \mathcal{P}$, this point can be expressed as a convex combination of the extreme points as

$$x = \sum_{i=1}^{2^n} \lambda_i(x)\hat{x}^{(i)}, \tag{7}$$

where the parameters $\lambda_i(x)$ are given by the following closed-form expression

$$\lambda_i(x) = \prod_{j=1}^{n} \mu_{i,j},$$

where

$$\mu_{i,j} = \begin{cases} x'_j & \text{if } \alpha_j(\hat{x}^{(i)}) = 1 \\ 1 - x'_j & \text{if } \alpha_j(\hat{x}^{(i)}) = -1. \end{cases} \tag{8}$$

Thereby, x' is the transformed point of x under the affine transformation (6).

Note that for a point $\hat{x}^{(i)} \in \mathcal{P}$ to be an extreme point, the entries in the corresponding parameter vector $\alpha(\hat{x}^{(i)})$ must all be ± 1; therefore one of the cases in (8) is always satisfied.

Proof: We first prove by induction that (7) is a convex combination. We then show that this convex combination (7) actually results in the point x.

Since $\mathcal{P}'$ is the unit hypercube, it follows that for any $x' \in \mathcal{P}'$, $x'_i \in [0,1], \forall i$. Because $\lambda_i(x)$ is a product of the entries x'_j or $(1 - x'_j)$, i.e., a product of n numbers between 0 and 1, its value must be between 0 and 1, too, which proves the first of the two statements.

Let us now show the second part by induction over the number of states n. We state that

$$\sum_{i=1}^{2^n} \lambda_i = \sum_{i=1}^{2^n} \prod_{j=1}^{n} \mu_{i,j} = 1. \tag{9}$$

Before we begin, note that it follows from the fact that the $\alpha(\hat{x}^{(i)})$ consist of all 2^n possible combinations of ± 1 entries, i.e., $\alpha(\hat{x}^{(1)}) = \begin{bmatrix}-1 & -1 & \dots & -1\end{bmatrix}^T, \alpha(\hat{x}^{(2)}) = \begin{bmatrix}1 & -1 & \dots & -1\end{bmatrix}^T, \dots, \alpha(\hat{x}^{(2^n)}) = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^T$, that we can find for any given $\lambda_i(x)$ and an arbitrary $l \in \{1, \dots, n\}$ another $\lambda_k(x)$ for which $\mu_{i,j} = \mu_{k,j}, \forall j \neq l$ and $\mu_{i,l} = 1 - \mu_{k,l}$. For an easier notation, we order the extreme points such that $\mu_{1,n} = \mu_{2,n} = \cdots = \mu_{2^{n-1},n}$ and $\mu_{2^{n-1}+1,n} = \mu_{2^{n-1}+2,n} = \cdots = \mu_{2^n,n} = 1 - \mu_{1,n}$.

Let us now begin by showing that (9) holds for $n = 1$:

$$\sum_{i=1}^{2^1} \prod_{j=1}^{1} \mu_{i,j} = \mu_{1,1} + \mu_{2,1} = \mu_{1,1} + (1 - \mu_{1,1}) = 1,$$

which follows from (8) and from the fact that we have only two extreme points in the one-dimensional case.

The induction step is done by showing that we can reduce (9) from $n = N+1$ to $n = N$, for any $N \geq 1$:

$$\sum_{i=1}^{2^{N+1}} \prod_{j=1}^{N+1} \mu_{i,j} = \sum_{i=1}^{2^{N+1}} \mu_{i,N+1} \prod_{j=1}^{N} \mu_{i,j} \tag{10}$$

$$= \mu_{1,N+1} \sum_{i=1}^{2^N} \prod_{j=1}^{N} \mu_{i,j} + \underbrace{\mu_{2^N+1,N+1}}_{=(1-\mu_{1,N+1})} \sum_{i=1}^{2^N} \prod_{j=1}^{N} \mu_{i,j} \tag{11}$$

$$= \sum_{i=1}^{2^N} \prod_{j=1}^{N} \mu_{i,j}. \tag{12}$$

Since we can order the $\lambda_i(x)$ such that the same is possible for the $\mu_{i,j}$ for $n = N$, we can reduce this sum down to $n = 1$, for which we know that $\sum_{i=1}^{2^1} \prod_{j=1}^{1} \mu_{i,j} = 1$. This shows that $\sum_{i=1}^{n} \lambda_i(x) = 1$.

The only remaining part to show is that (7)-(8) actually describes x correctly. We can do so by using the previous results from the induction proof. First note that because of the transformation (6), the extreme points of $\mathcal{P}'$ are the corner points of the unit hypercube, i.e., $\hat{x}'^{(1)} = \begin{bmatrix}0 & 0 & \dots & 0\end{bmatrix}^T, \hat{x}'^{(2)} = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^T, \dots, \hat{x}'^{(2^n)} = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^T$, and that $\hat{x}_j'^{(i)} = 1$ if and only if $\alpha_j(\hat{x}^{(i)}) = 1, \forall i,j$, see also Fig. 2. From this structure it follows that $\lambda_i(x)$ contains the factor x_j' if and only if $\hat{x}_j'^{(i)} = 1$. Let us now consider the $k-$th entry of x' if computed by (7)-(8). For an easier notation and without loss of generality, we assume that we ordered the extreme points such that $\hat{x}_k'^{(1)} = \cdots = \hat{x}_k'^{(2^{n-1})} = 1$ and $\hat{x}_k'^{(2^{n-1}+1)} = \cdots = \hat{x}_k'^{(2^n)} = 0$. Then the following holds:

$$\sum_{i=1}^{2^n} \lambda_i(x) \hat{x}_k'^{(i)} = \sum_{i=1}^{2^n} \prod_{j=1}^{n} \mu_{i,j} \hat{x}_k'^{(i)} \tag{13}$$

$$= \sum_{i=1}^{2^n} \mu_{i,k} \prod_{j=1}^{k-1} \mu_{i,j} \prod_{j=k+1}^{n} \mu_{i,j} \hat{x}_k'^{(i)} \tag{14}$$

$$= x_k' \underbrace{\sum_{i=1}^{2^{n-1}} \prod_{j=1}^{k-1} \mu_{i,j} \prod_{j=k+1}^{n} \mu_{i,j}}_{=1 \text{ (see explanation below)}} 1 \tag{15}$$

$$+ (1 - x_k') \sum_{i=2^{n-1}+1}^{2^n} \prod_{j=1}^{k-1} \mu_{i,j} \prod_{j=k+1}^{n} \mu_{i,j} 0 \tag{16}$$

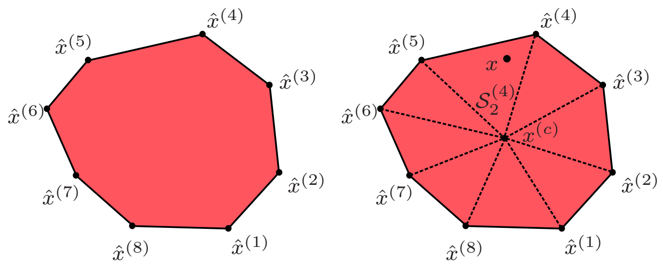
$$= x_k' 1 = x_k', \tag{17}$$

where the sum over the remaining $n-1$ factors in (15) is equal to one because of the same arguments as in the induction proof (10)-(12). Since this holds for every $k \in \{1, \dots, n\}$, this convex combination actually results in x'.

The affine transformation (6) does not change any convexity properties [16]. Therefore, analogous to (5), the state x can be represented by the convex combination of the extreme points $\hat{x}^{(i)}$ by using the parameters for the transformed state $\lambda_i(x')$, which concludes the proof. ■

For the special case of axis-aligned boxes, the presented approach can be simplified to the closed-form expression which was stated without a proof in [1] and [8].

Note that while we describe the approaches for full-dimensional simplices and parallelotopes only, they can be adapted for lower-dimensional simplices and parallelotopes. One simply has to project the points on a lower-dimensional subspace in which they become full-dimensional simplices or parallelotopes. Then our presented techniques can be applied again.



(a) Original polytope (b) Point inside Polytope

Fig. 3. (a) Given is a general polytope $P \subset \mathbb{R}^2$ defined as the convex hull of its $p = 8$ extreme points $\hat{x}^{(i)}$. (b) The boundary $\mathcal{B}(P)$ is divided into $q = 8$ simplices $\mathcal{S}_1^{(i)}$ of dimension $n-1$ (here: line segments). Together with a center point $x^{(c)}$, the boundary simplices define q simplices $\mathcal{S}_2^{(i)}$ of dimension n (here: triangles). Any point x inside the polytope can be expressed by the closed-form convex combination of the simplex it is contained.

C. General Polytopes

In the previous two subsections, we discussed how to find analytical closed-form expressions for simplices and parallelotopes. In the case of simplices, the resulting convex combination is unique. This is not necessarily the case with parallelotopes and general polytopes. For parallelotopes, we presented an approach which always results in the same convex combination. This is possible since we are able to uniquely transform simplices and parallelotopes to axis-aligned simplices and axis-aligned boxes. Since the corresponding mapping matrices are square, they are invertible. For other shapes like general polytopes, it is not possible to find such a map.

While it is not possible to provide an analytical closed-form expression for general polytopes, we are nevertheless always able to obtain a closed-form expression which can be solved without using any iterations. Therefore, this approach can be used online in real-time without the disadvantages of using linear programming approaches.

The basic idea for the general polytope is the fact that any polytope with finitely many extreme points can be expressed by the union of finitely many non-overlapping simplices

[18]. For each of these simplices an analytical closed-form expression can be found. This process is summarized in Algorithm 1 and visualized in Fig. 3.

Algorithm 1 Convex Decomposition Algorithm for General Polytopes

```
1:  Given: Polytope $P$ with $p$ extreme points $\hat{x}^{(1)}, \dots, \hat{x}^{(p)}$    ▷ offline
2:  Connect each extreme point with its neighboring extreme points along the edges of the polytope $\rightarrow$ results in $q$ simplices $\mathcal{S}_{n-1}^{(1)}, \dots, \mathcal{S}_{n-1}^{(q)}$ in $\mathbb{R}^{n-1}$
3:  Define $x^{(c)} \leftarrow \sum_{i=1}^{p} \frac{1}{p} \hat{x}^{(i)}$
4:  Connect $x^{(c)}$ with all extreme points $\hat{x}^{(i)} \rightarrow$ results in $q$ simplices $\mathcal{S}_{n}^{(1)}, \dots, \mathcal{S}_{n}^{(q)}$ in $\mathbb{R}^{n}$
5:  for $i = 1, \dots, q$ do
6:      Use Theorem 1 to obtain the closed-form for the convex combination of all points inside simplex $\mathcal{S}_n^{(i)}$ and store them
7:  end for
8:  Given $x \in P$    ▷ online
9:  Find $i$, such that $x \in \mathcal{S}_n^{(i)}$
10: Use the closed-form convex expression of simplex $\mathcal{S}_n^{(i)}$ to describe $x$ as convex combination of its extreme points
```

The boundary of the polytope $P \subset \mathbb{R}^n$, denoted by $\mathcal{B}(P)$, defines an $n-1$ dimensional subspace of $\mathbb{R}^n$. We divide this space into q simplices $\mathcal{S}_{n-1}^{(i)}, i \in \{1, \dots, q\}$, each of them defined by n extreme points of P such that $\bigcup_{i=1}^{q} \mathcal{S}_{n-1}^{(i)} = \mathcal{B}(P)$, and $\forall i,j : \mathcal{S}_{n-1}^{(i)} \cap \mathcal{S}_{n-1}^{(j)} = \emptyset$. We use this division of the boundary of P to divide P itself into q simplices. We use the fact that if we choose any point $\tilde{x} \in \mathcal{I}(P)$ in the interior of P and compute the convex hull of this $\tilde{x}$ and the extreme points of a simplex $\mathcal{S}_{n-1}^{(i)}$ on the boundary of P, we obtain an $n-$dimensional simplex $\mathcal{S}_n^{(i)} \subset P$. By doing this for every simplex on the boundary, we obtain q simplices $\mathcal{S}_n^{(1)}, \dots, \mathcal{S}_n^{(q)}$ such that $\bigcup_{i=1}^{q} \mathcal{S}_n^{(i)} = P$ and $\forall i,j : \mathcal{S}_n^{(i)} \cap \mathcal{S}_n^{(j)} = \emptyset$. Note that this works with any point in the interior of P. However, we choose $\tilde{x} = x^{(c)} = \sum_{i=1}^{p} \frac{1}{p} \hat{x}^{(i)}$ because we then immediately know that $\lambda_i(x^{(c)}) = \frac{1}{p}$ are the corresponding parameters of the convex combination for $x^{(c)}$. Since $x^{(c)}$ is in many cases around the center, the simplices will have similar sizes.

When this approach is used to obtain the convex combination for a given point $x \in P$ online, we check which simplex contains x. We then simply have to use the offline-computed closed-form expression for the convex combinations in this simplex in the same way as in Subsection IV-A. Checking which simplex contains x is a point location problem, for which efficient solutions exist, e.g., [19]. As we show in the next section, it is even feasible to check all simplices.

V. Numerical Results

In this section, we compare the previously presented closed-form approaches to obtain convex combinations with existing approaches which solve (1) using linear programming. All computations are performed using MATLAB on a computer with a 3.1 GHz dual-core i7 processor and without using parallel computing. We performed all pre-computations which do not depend on the actual point inside the polytope offline in advance.

TABLE I
Average run times for simplices and parallelotopes (in ms)

Dimension	Simplices		Parallelotopes	
	Cl.-Form	Lin. Progr.	Cl.-Form	Lin. Progr.
2	0.0052	7.3922	0.0049	8.3247
3	0.0053	7.8886	0.0052	9.7929
5	0.0053	8.4302	0.0071	10.5306
8	0.0055	8.5203	0.0438	16.9912
10	0.0057	8.5928	0.1243	30.3809
12	0.0058	8.6812	0.4810	77.1852
15	0.0058	8.9057	3.5609	666.9693
50	0.0077	11.4315	–	–
100	0.0123	20.5933	–	–
200	0.0254	75.5463	–	–
1,000	0.5471	10,902	–	–

A. Simplices and Parallelotopes

We randomly generate 100 simplices and 100 parallelotopes each in 2, 3, 5, 8, 10, 12, and 15 dimensions. We also generate 100 simplices in 50, 100, 200, and 1,000 dimensions. Then we randomly generate points inside these parallelotopes and solve the convex decomposition problem first with our approach and then by solving a linear program using the `linprog` function with the default settings and the solution space bounded to $[0, 1]$. The results are presented in Table I.

We see that our approach is significantly faster in any of the presented dimensions for both simplices and parallelotopes. For simplices we are faster with factors between 1,400 and up to almost 20,000. The computation times for parallelotopes are between 160 and 1,900 times faster than using linear programming. We can compute the simplices for much higher dimensions than parallelotopes, since the number of extreme points increases linearly with the dimensions for simplices, while the number of extreme points for parallelotopes increases exponentially. Therefore, the computation times almost do not increase for small dimensions for simplices. For high dimensions, the closed-form expressions for parallelotopes become quite large, which takes a significant time to load the functions in Matlab before the online computations can begin.

B. General Polytopes

In this subsection, we compare our closed-form approach to the linear programming approach for general polytopes. During the implementation of our closed-form algorithm, we must decide how to check in which simplex the point is contained. There are many ways which can reduce the amount of simplices checked, for example the techniques in [19], or by computing the distance of the point to the center points of the q simplices $\mathcal{S}_{\mathbb{R}^n}^{(i)}$ and starting the computation with the nearest simplex. We choose the simplest strategy,

TABLE II

AVERAGE RUN TIMES FOR GENERAL POLYTOPES (IN MS)

# Extreme Points	Closed-Form Approach	Linear Programming
Dimension $n = 3$		
20	0.1585	9.5866
50	0.3234	10.1925
100	0.5317	11.9193
150	0.8613	13.2450
Dimension $n = 4$		
20	0.2888	9.9935
50	0.6673	10.1101
100	1.3337	11.2544
150	2.1542	12.7434
Dimension $n = 5$		
20	0.5205	10.1305
50	1.6159	10.2715
100	4.0592	11.2688
150	5.4805	12.0558

which is to check all simplices. Since it is also the worst-case scenario, it provides a lower bound for the performance of our approach. Checking all simplices is feasible, since for each simplex only a few operations have to be performed. We compute polytopes in 3, 4, and 5 dimensions with 20, 50, 100, and 150 vertices. For each of these combinations we randomly compute 10 polytopes, and for each polytope we randomly choose 10 points for which we solve the convex decomposition problem. The results are summarized in Table II. We see that our approach is faster in all of these cases, even though we used the easiest method of checking all simplices. The results can be further improved by either using better heuristics for choosing the order in which the simplices are checked or by using multi-core processors, as these tests can be performed in parallel without overhead since each computation is independent.

Note that we consider only the online computation times, since these are the critical times in control. This is of course only feasible if the polytopes are known in advance. For simplices and parallelotopes, our approach is still significantly faster than the linear programming approach, even if everything has to be computed online (i.e., mainly matrix inverses).

VI. CONCLUSION

Convex combinations are used for many applications in control, such as for robust gain-scheduling control approaches for polytopic LPV systems. In many cases it is necessary to have closed-form expressions of convex combinations, for example for formal verification of safety or for faster online computation times for fast dynamical systems. We provide analytical closed-form expressions for simplices and polytopes and prove their validity. Furthermore, we present a technique for obtaining closed-form expressions for general polytopes. We compare the computation times of our approaches to linear programming algorithms and show that our approaches provide significantly faster computation times, therefore making them suitable for the control of fast dynamical systems.

ACKNOWLEDGMENTS

The author gratefully acknowledges financial support by the European Commission project UnCoVerCPS under grant number 643921 and the German Research Foundation (DFG) under grant number AL 1185/2-1.

REFERENCES

[1] H. Chaofan, Y. Lingyu, W. Zhenchao, S. Bin, and Z. Jing, "Linear parameter-varying attitude controller design for a reusable launch vehicle during reentry," in *IEEE Chinese Guidance, Navigation and Control Conference*, 2014, pp. 2723–2728.

[2] P. Apkarian, P. Gahinet, and G. Becker, "Self-scheduled H-infinity control of linear parameter-varying systems: a design example," *Automatica*, vol. 31, no. 9, pp. 1251–1261, 1995.

[3] P. Apkarian and R. J. Adams, "Advanced gain-scheduling techniques for uncertain systems," *IEEE Transactions on Control Systems Technology*, vol. 6, no. 1, pp. 21–32, 1998.

[4] P. Apkarian, J.-M. Biannic, and P. Gahinet, "Self-scheduled H-infinity control of missile via linear matrix inequalities," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 3, pp. 532–538, 1995.

[5] L. Chen and D. Duan, "Polytopic LPV system based control design for multi-propeller airship," in *Chinese Control Conference*, 2014, pp. 4273–4276.

[6] E. B. Muhando, T. Senjyu, A. Uehara, and T. Funabashi, "Gain-scheduled control for wecs via LMI techniques and parametrically dependent feedback part ii: Controller design and implementation," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 1, pp. 57–65, 2011.

[7] X.-J. Yao, C.-C. Guo, and Y. Li, "LPV H-infinity controller design for variable-pitch variable-speed wind turbine," in *IEEE International Power Electronics and Motion Control Conference*, 2009, pp. 2222–2227.

[8] Dengying and Zhoujie, "LPV H-infinity controller design for a wind power generator," in *IEEE Conference on Robotics, Automation and Mechatronics*, 2008, pp. 873–878.

[9] A. Forrai, T. Ueda, and T. Yumura, "Electromagnetic actuator control: A linear parameter-varying (LPV) approach," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 3, pp. 1430–1441, 2007.

[10] Z. Yu, H. Chen, and P.-y. Woo, "Gain scheduled LPV H-infinity control based on LMI approach for a robotic manipulator," *Journal of Robotic Systems*, vol. 19, no. 12, pp. 585–593, 2002.

[11] H. Koc, D. Knittel, M. De Mathelin, and G. Abba, "Modeling and robust control of winding systems for elastic webs," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 2, pp. 197–208, 2002.

[12] Z. Liu, Q. Zhu, and L. Wang, "Predictive control for multi-joint manipulator with polytopic model," in *IEEE International Conference on Robotics and Biomimetics*, 2009, pp. 2130–2133.

[13] V. F. Montagner, R. C. L. F. Oliveira, V. J. S. Leite, and P. L. D. Peres, "Gain scheduled state feedback control of discrete-time systems with time-varying uncertainties: an LMI approach," in *IEEE Conference on Decision and Control and European Control Conference*, 2005, pp. 4305–4310.

[14] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler, "Recent progress in continuous and hybrid reachability analysis," in *IEEE Conference on Computer Aided Control Systems Design*, 2006, pp. 1582–1587.

[15] M. Althoff and B. H. Krogh, "Reachability analysis of nonlinear differential-algebraic systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 371–383, 2014.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[17] A. F. Möbius, *Der barycentrische Calcul*. Verlag von Johann Ambrosius Barth, 1827.

[18] C. W. Lee, "Subdivisions and triangulations of polytopes," in *Handbook of Discrete and Computational Geometry*, 2nd ed., J. E. Goodman and J. O'Rourke, Eds. Chapman and Hall/CRC, 2004.

[19] P. Tøndel, T. A. Johansen, and A. Bemporad, "Evaluation of piecewise affine control via binary search tree," *Automatica*, vol. 39, no. 5, pp. 945–950, 2003.