European Journal of Control (2007)13:152–170
© 2007 EUCA
DOI:10.3166/EJC.13.152–170

European Journal of Control

Multiparametric Linear Programming with Applications to Control

C.N. Jones*, M. Barić and M. Morari

Automatic Control laboratory, ETH Zurich, Physikstrasse 3, Zurich, Switzerland

Parametric programming has received a lot of attention in the control literature in the past few years because model predictive controllers (MPC) can be posed in a parametric framework and hence pre-solved offline, resulting in a significant decrease in on-line computation effort. In this paper we survey recent work on parametric linear programming (pLP) from the point of view of the control engineer. We identify three types of algorithms, two arising from standard convex hull paradigms and one from a geometric intuition, and classify all currently proposed methods under these headings. Through this classification, we identify a third standard convex hull approach that offers significant potential for approximation of pLPs for the purpose of control. We present the resulting algorithm, based on the beneath/beyond paradigm, that computes low-complexity approximate controllers that guarantee stability and feasibility.

Keywords: Parametric linear programming; approximation; explicit model predictive control

1. Introduction

It is standard practice to implement a model predictive controller (MPC) by solving an optimization problem on-line. For example, when the system is linear, the constraints are polyhedral and the cost is linear or quadratic, this amounts to computing a single linear or quadratic program (LP/QP) at each sampling instant. In recent years, it has become well-known that for this class of systems the optimal input is a piecewise affine function (PWA) defined over a polyhedral partition of the feasible states [10,34,59]. By pre-computing this PWA function off-line, the on-line calculation of the control input then becomes one of evaluating the PWA function at the current measured state, which allows for significant improvements in sampling speed.

In this paper, we restrict our attention to the case when the cost is linear, 1- or ∞-norm. The computation of the optimal PWA function, mapping the measured state to the control input, can then be posed as the following (multi) parametric linear program (pLP) with parameters entering in the right-hand side (RHS) of the constraints:

$$\min_{y}\{b^T y \mid (\theta, y) \in \mathcal{P}\}, \tag{1}$$

where $\theta \in \mathbb{R}^d$ is the parameter, or state, $y \in \mathbb{R}^m$ is the optimizer, or control input and slack variables and $\mathcal{P}$ is a polyhedron, which incorporates the system constraints and is assumed bounded.

Both the properties and the computation of multiparametric linear programs (pLP) has been discussed in the literature for many years. Early work on parametric linear programming dates back to W. Orchard-Hays in his Master's thesis of 1952, published in [50] and Saaty and Gass [54]. Further studies have been ongoing ever since, primarily focusing on one-dimensional parametric problems and sensitivity analysis. The interested reader is referred to [24] for a survey of early work. There has, however, been a significant resurgence of interest on multi-parametric linear programming in the past few years in the control community due to the aforementioned link with model predictive control.

Correspondence to: C.N. Jones, E-mail: cjones@ee.ethz.ch

*Received 20 December 2006; Accepted 20 January 2007.
Recommended by S.G. Tzafestas and P.J. Antsaklis*

In this paper we will survey current methods of solving the pLP (1) from the point of view of control theory. We will first demonstrate that all parametric linear programs of the form (1) can be posed as vertex or facet enumeration problems and then conduct the survey by classifying the methods in the literature into one of the three main paradigms for vertex/facet enumeration known in computational geometry and a fourth paradigm arising from geometrical intuition. It will be seen that current methods fall into three of these four classes of algorithms.

We will then propose a new method based on the fourth, as yet unstudied paradigm, and demonstrate various beneficial properties of this approach to control. Specifically, we will see that methods based on the so-called beneath/beyond method can be used to generate approximate, yet stabilizing, and invariance-inducing control laws. Furthermore, tuning knobs are available that allow the designer to trade-off between complexity, region of attraction and performance of the approximate control law.

The remainder of the paper is organized as follows. Sections 2 and 3 provide the basic background and problem setup for parametric linear programming. Section 4 demonstrates that all pLP of the form (1) can be posed as vertex or facet enumeration problems. Section 5 presents a survey of current methods in terms of the established approaches to vertex/facet enumeration and their relative complexity is analyzed in Section 6. Finally, a numerical examples demonstrating effectiveness of the proposed beneath/beyond scheme is given in Section 7.

2. Notation and Background

If $A \in \mathbb{R}^{m\times n}$ and $I \subseteq \{1, \ldots, n\}$, then $A_{*,I} \in \mathbb{R}^{m\times|I|}$ is the matrix formed by the columns of A indexed by I. If $c \in \mathbb{R}^n$ is a vector then c_I is the vector formed by the elements of c in I. If $R \subseteq \{1, \ldots, m\}$ then we will use the notation $A_{R,*} \in \mathbb{R}^{|R|\times n}$ to denote the matrix formed by the rows of A indexed by R.

A set S is called *affine* if $(1-\lambda)x + \lambda y \in S$ for all $x,\ y \in S$ and $\lambda \in \mathbb{R}$ and *convex* if λ is restricted to lie between zero and one. The *affine (convex) hull* of a set S is the intersection of all affine (convex) sets containing S, denoted aff S (conv S). If S is a finite point set, then aff $S = \{\sum s_i\lambda_i | s_i \in S, \sum \lambda_i = 1\}$ and conv $S = \{\sum s_i\lambda_i | s_i \in S, \sum\lambda_i = 1, \lambda_i \geq 0\}$. The *dimension* of a set is the dimension of the subspace parallel to its affine hull.

A *polyhedron* is the intersection of a finite number of halfspaces and a *polytope* is a bounded polyhedron. If $P = \{x | Ax \leq b\}$ is a polyhedron and $H = \{x | a^T x \leq d\}$ is a halfspace such that $P \subseteq H$, then $P \cap \{x | a^T x = d\}$ is a *face* of P. One- and zero-dimensional faces are called *edges* and *vertices* respectively. If P is of dimension d, then (d-1)- and (d-2)-dimensional faces are called *facets* and *ridges* respectively. The inequality $A_{i,*}x \leq b_i$ is called *redundant* if $P = \{x | A_{\{1,\ldots,n\}\setminus\{i\},*}x \leq b_{\{1,\ldots,n\}\setminus\{i\}}\}$ and *irredundant* otherwise.

A set C is called a cone if for every $x \in C$ and scalar $\alpha \geq 0$, we have $\alpha x \in C$. The columns of a matrix $F \in \mathbb{R}^{m\times n}$ are called the *generators* of the cone $C = cone\ F := \{F\alpha \,|\, \alpha \geq 0\}$, if F is a single column, then cone F is called a *ray*. The generator $A_{*,i}$ is called *redundant* if $F_{*,i} \in \text{cone}(F_{*,\{1,\ldots,n\}\setminus\{i\}})$ and *irredundant*, or *extreme* otherwise.

The *Minkowski sum* of two sets, denoted $A \oplus B$ is defined as $A \oplus B := \{x + y \,| x \in A,\ y \in B.\}$. Every polyhedron P can be written as the Minkowski sum of a convex hull of a finite point set V and the *conic hull* of a finite number of rays R, $P = \text{conv}\ V \oplus \text{cone}R$.

A hyperplane $h := \{x | a^T x = c\}$ is said to *separate* two sets S_1 and S_2 if $S_1 \subseteq \{x | a^T x \leq c\}$ and $S_2 \subset \{x | a^T x \geq c\}$.

3. Preliminaries

3.1. Optimal Control

The recent interest in parametric programming in the control community has arisen from the ability to pose certain optimal control problems as parametric problems and thereby pre-compute the optimal control law offline. In this paper, we are specifically interested in the following standard semi-infinite horizon optimal control problem:

$$J^*(x) = \min_{\{u_0,\ldots,u_{N-1}\}} V_N(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i) \qquad (2)$$

$$\begin{aligned} \text{s.t. } & x_{i+1} = Ax_i + Bu_i, \quad \forall i = 0,\ldots,N-1 \\ & (x_i, u_i) \in \mathcal{X} \times \mathcal{U}, \quad \forall i = 0,\ldots,N-1 \\ & x_N \in \mathcal{X}_F, \\ & x_0 = x \end{aligned}$$

where $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{X}_F$ are polytopic constraints on the states and inputs and the stage cost l is defined as $l(x_i, u_i) := \|Qx_i\|_p + \|Ru_i\|_p$. Under the standard assumptions that $\mathcal{X}_\mathcal{F} \subseteq \mathcal{X}$ is an invariant set, V_N is a Lyapunov function and that the decay rate of V_N is greater than the stage cost within the set $\mathcal{X}_F$ then the problem (2) generates a stabilizing control law when applied in a receding horizon fashion [44]. If the norm is taken to be the one or infinity norm, then we get a pLP of the form:

$$u^*(x) = \arg\min_u \{g^T u | Gu \leq Fx + w\}, \tag{3}$$

where u is a vector containing the sequence of inputs $u_0, \ldots, u_{N-1}$ and appropriate slack variables. See [14] for details on the conversion for the form (2) to (3).[1]

'Solving' the pLP (2) means to compute the optimal control input $u^\star(x)$ for every feasible value of the state.

3.2. Parametric Linear Programming

In this paper we consider the following primal-dual pair of parametric linear programs:

$$J^*(\theta) = \min_y \{b^T y | (\theta, y) \in \mathcal{P}\}, \tag{4a}$$

$$= \max_\lambda \{\theta^T E\lambda | \lambda \in \mathcal{D}\} \tag{4b}$$

The polyhedra $\mathcal{P}$ and $\mathcal{D}$ are defined as

$$\mathcal{P} := \{(\theta, y) \in \mathbb{R}^d \times \mathbb{R}^m | A^T y \leq E^T \theta\}, \tag{5a}$$

$$\mathcal{D} := \{\lambda \in \mathbb{R}^n | A\lambda = b, \ \lambda \geq 0\}, \tag{5b}$$

where $A \in \mathbb{R}^{m\times n}$ and $E \in \mathbb{R}^{d\times n}$ are full-rank matrices and $b \in \mathbb{R}^m$ is a vector. Throughout the remainder of the paper, we will refer to (4a) as the primal and (4b) as the dual. The polytopes $\mathcal{P}$ and $\mathcal{D}$ will be called the primal and dual constraints respectively.

Remark 3.1. Note that the pLP (4) differs from the control pLP (3) by the lack of the offset term w. This discrepancy will be removed in Section 3.4, where pLP (3) will be homogenized into the appropriate form.

It is assumed that the set of feasible parameters $\pi_\theta \mathcal{P} := \{\theta | \exists y, (\theta, y) \in \mathcal{P}\}$, given by the projection of $\mathcal{P}$ onto the parameter space, is full-dimensional, or equivalently, has a non-empty interior. This is a common assumption and can be easily guaranteed through a pre-processing operation [14]. The standard assumption is also made that pLP (4) has an optimal bounded solution for every $\theta \in \pi_\theta \mathcal{P}$[2].

Any set $B \subset \{1, \ldots, n\}$ such that $|B| = m$ and rank $A_{*,B} = m$ is called a *basis* and we write $N = \{1, \ldots, n\} \backslash B$ for its complement. We partition the constraints in (5a) and the variables in (5b) by the sets B and N and refer to the partitions as the basic and non-basic constraints and variables respectively.

Every basis B defines a basic primal and a basic dual solution by requiring the basic and non-basic constraints to be active for the primal and dual respectively

$$y = \beta^T E^T_{*,B} \theta, \tag{6a}$$

$$\lambda_B = \beta b, \quad \lambda_N = 0, \tag{6b}$$

where we will use the standard notation $\beta := A^{-1}_{*,B}$ throughout the paper for simplicity. A basis is called primal or dual feasible if the resulting solution also satisfies the inequality constraints in $\mathcal{P}$ or $\mathcal{D}$ respectively.

3.3. Optimality Conditions

Definition 3.2 (Tangent and normal cones [11]). Let z be an element of the polyhedron $\mathcal{Z} \subseteq \mathbb{R}^n$. A vector $\gamma \in \mathbb{R}^n$ is said to be a feasible direction at z if there exists a strictly positive scalar α for which $z + \alpha\gamma \in \mathcal{Z}$. The set of all feasible directions at z is called the tangent cone and is written $\mathcal{T}_z(z)$. The normal cone to $\mathcal{Z}$ at z is the orthogonal complement of the tangent cone:

$$\mathcal{N}_z(z) := \{v | \ v^T \gamma \leq 0, \quad \forall \gamma \in \mathcal{T}_z(z)\}$$

Theorem 3.3 (Optimality Condition (e.g. [47]). If z^* is an element of the polyhedron $\mathcal{Z}$, then z^* is a global minimizer of the linear program $\min_z\{c^T z | z \in \mathcal{Z}\}$ if and only if $-c \in \mathcal{N}_z(z^*)$.

Given a basis B and the resulting feasible basic primal solution y, the tangent cone to the primal constraints is given by relaxing the basic constraints from equality to inequality

$$\begin{aligned} \mathcal{T}_\mathcal{P}(y) &= \{\rho | A^T_{*,B} \rho \leq 0\}, \\ \mathcal{N}_\mathcal{P}(y) &= \{v | \beta v \geq 0\}. \end{aligned} \tag{7}$$

Similarly, the feasible directions from a feasible basic dual solution λ that is defined by a basis B are given by increasing each non-basic variable in a positive, and hence feasible, direction:

$$\begin{aligned} \mathcal{T}_\mathcal{D}(\lambda) &= \{\gamma | \gamma_B = -\beta A_{*,N} \gamma_N, \quad \gamma_N \geq 0\}, \\ \mathcal{N}_\mathcal{D}(\lambda) &= \{v | A^T_{*,N} \beta^T v_B \geq v_N\}. \end{aligned} \tag{8}$$

Note that neither the primal nor the dual normal cones are a function of the parameter θ, but only of the basis B.

[1] Note that we here assume that constraints that appear only on the parameter, or state, θ are incorporated into the polyhedron $\mathcal{P}$ rather than given separately.

[2] Note that this also implies that the dual solution is feasible and bounded for all $\theta \in \pi_\theta \mathcal{P}$.

From the general optimality condition given in Theorem 3.3 and Equations 7 and 8, we have the following standard result, which is that a basis defines an optimal solution if and only if the primal and dual basic solutions are both feasible.

Corollary 3.4. [46] If B is a basis of LP (4), y and λ are the primal and dual basic solutions respectively, then y and λ are optimizers of LP (4a) and LP (4b) if and only if both y and λ are feasible.

Definition 3.5 (Critical Region [25]). If B is a basis of pLP (4), then the critical region $\mathcal{R}_B$ is the set of all parameters $\hat{\theta}$ such that the basic solutions (6) defined by B are optimal for $\theta = \hat{\theta}$.

Corollary 3.4 and Eqs 7 and 8 show that for a given basis B, the critical region $\mathcal{R}_B$ is the polyhedral set

$$\begin{aligned}\mathcal{R}_B &= \{\theta | -b \in \mathcal{N}_{\mathcal{P}}(y), \quad E\theta \in \mathcal{N}_{\mathcal{D}}(\lambda)\} \\ &= \{\theta | \beta b \geq 0, \quad \bar{E}\theta \geq 0\},\end{aligned}$$

where y and λ are the basic solutions for the basis B and we use the standard notation $\bar{E}$ for the *reduced cost*, which is defined as $\bar{E} := E^T_{*,N} - A^T_{*,N}\beta^T E^T_{*,B}$.

Remark 3.6. Note that we have taken the original definition of critical region here [25], rather than that used in much of the recent literature, where a critical region is defined as the set of all parameters for which the constraints active at the optimizer do not change. In the case of non-degeneracy, these two definitions are equivalent and since, as discussed in Section 3.5, we will use lexicographic perturbation to simulate non-degeneracy we can consider these definitions equivalent in this paper.

The following results follow immediately from the definition of critical region.

Lemma 3.7. [5]

1. For each θ in the set of feasible parameters $\pi_\theta\mathcal{P}$ there exists a basis B such that $\theta \in \mathcal{R}_B$
2. Let $\mathbb{B}$ be the set of all bases for which $\dim \mathcal{R}_B = d$, then $\pi_\theta\mathcal{P} = \cup_{\mathbb{B}}\mathcal{R}_B$

The goal of 'solving' a parametric linear program can now be re-stated as finding a set of bases $\{B_0, \ldots, B_N\}$ such that the corresponding critical regions $\{\mathcal{R}_{B_0}, \ldots, \mathcal{R}_{B_N}\}$ are full-dimensional and cover the set of feasible parameters $\pi_\theta\mathcal{P}$, i.e. $\pi_\theta\mathcal{P} = \bigcup_{i=0}^N \mathcal{R}_{B_i}$.

Remark 3.8. In Section 3.5 we demonstrate that for non-degenerate pLPs, or those that have been lexicographically perturbed that the interiors of the full-dimensional critical regions do not overlap.

Remark 3.9. Since we are interested in solving the parametric linear program (4) in order to compute a control law, we are not required to find the bases that define lower-dimensional critical regions. This is because, as will be seen in Section 3.5, it is possible to ensure that the primal optimizer, or control law, is everywhere continuous and as a result the control action defined in the lower-dimensional regions is equal to that given on the boundary of the full-dimensional ones.

3.4. Homogenization

The standard parametric linear program resulting from model predictive control problems takes the form [5]:

$$u^*(x) = \arg\min_u \{g^T u | Gu \leq Fx + w\}, \tag{9}$$

where x is the measured state vector, and the decision variable u is a sequence of inputs that will be applied to the system and some slack variables. The goal is to compute the optimal input $u^*(\cdot)$ as a function of the state x. The set of feasible states $\{x | \exists u, Gu \leq Fx + w\}$ is assumed to be a polytope and therefore bounded, and the pLP (9) is assumed to have a bounded optimizer for every feasible value of the parameter.

In this section we will demonstrate how to transform pLP (9) into an equivalent pLP of the form used in this paper (4), which differs by the affine offset w. The following theorem demonstrates that any problem of the form (9) can be *homogenized*, or *lifted*, into an equivalent problem of the form (4) by adding an extra positive parameter t [64,§1.5].

Theorem 3.10. [35] B is an optimal basis of pLP (9) for the parameter x if and only if it is an optimal basis of

$$\hat{J}^*(\hat{x}, t) = \min_{\hat{u}} \{g^T \hat{u} | G\hat{u} \leq F\hat{x} + wt\} \tag{10}$$

for the parameter $(\hat{x}, t) = (xt, t)$, for any strictly positive t.

Furthermore, the optimal cost and optimizer of pLP (10) and of pLP (9) are related by

$$\hat{J}^*(tx, t) = tJ^*(x), \quad \hat{u}^*(tx, t) = tu^*(x) \tag{11}$$

for all positive t.

Remark 3.11. Note that in order to satisfy assumptions made in Section 3.2, we must here assume that $[E\ c]$ is full rank.

The process of lifting the constraints in (10) does not effect the combinatorial structure of the polyhedron [64, §1.5]. Facets that were adjacent are still adjacent, and adjacent vertices become adjacent rays. There are two main benefits of homogenizing the pLP. First, as will be seen in Section 4, a pLP of the form (4) can be viewed as a vertex or a facet enumeration problem and hence the well-developed tools for these problems can be brought to bear. Second, if the constraints are polyhedral and unbounded, then it is possible that the *boundary complex*, which is the set of faces of the constraints, is disconnected and this can cause difficulties in many algorithms that operate by traversing this complex. The lifting removes this problem as it converts all polyhedra into cones that have connected boundary complexes.

3.5. Degeneracy

A basis B is called *primal-degenerate* if there are more constraints active at the basic primal solution than are in the basis and *dual-degenerate* if some of the basic dual-variables are zero. The basis is non-degenerate if it is neither primal nor dual degenerate. If a basis is primal-degenerate, then there exists more than one feasible basis that gives the same basic primal solution. Dual-degeneracy means that there is more than one optimal primal solution, or equivalently that the dual problem is primal degenerate.

Primal degeneracy is not in general a problem for control, as the primal optimizer, which defines the control input, is still uniquely defined. However, the non-unique representation can cause difficulty with some algorithms since it is possible for different bases that give the same control law to generate overlapping critical regions.

From a control point of view the non-uniqueness of the primal optimizer is a problem as this optimizer defines the control input that will be applied. If the pLP is dual-degenerate and therefore multi-valued for a particular parameter, or state, then a choice must be made as to which of this set of optimizers will be applied to the system. If a systematic procedure is not in place to make this selection, it is possible to generate a discontinuous control input, which can cause chattering and instability.

There have been three proposals made in the control literature to handle degeneracy. In [60], all dual-degenerate regions of the feasible parameter set are identified and then a strictly convex multiparametric quadratic program (mpQP) is solved over the set of optimal solutions in each such region. It is proven in [60] that the resulting polyhedral subdivision is non-overlapping and unique, and that the primal optimizer is both unique and continuous. This approach has been shown to perform well in practice and is implemented in the Multiparametric Toolbox (MPT) [41]. A *parametrized vertex* approach is proposed in [49], that can generate a continuous selection from dual-degenerate problems. The approach allows for a lot of flexibility, although the requirement to compute the parametrized vertices can be prohibitive.

The third method [38] is based on *lexicographic perturbation*, which is a well-established [18] method of dealing with degeneracy in the simplex algorithm. In this approach, the right hand side and the cost of the constraints are symbolically perturbed in such a way as to guarantee that there is a single unique optimal basis for each value of the parameter. The primary benefit is that it 'simulates simplicity' [20] of the optimization problem, ensuring that the perturbed problem is everywhere non-degenerate. This is a common technique used in the geometry literature since it allows the algorithm designer to ignore the difficulties and special cases arising from degeneracy and hence makes for simpler algorithms both conceptually and in implementation. The main result of lexicographic perturbation is the following theorem.

Theorem 3.12. [22,38] There exists a positive number $\gamma > 0$, such that whenever $0 < \epsilon_0 < \delta_0 < \gamma$, the following perturbed problem is both primal and dual non-degenerate for every value of the parameter θ

$$\min_y \{(b+\delta)^T y \mid A^T y \le E^T\theta + \epsilon\} \tag{12}$$

where $\epsilon^T := [\epsilon_0\, \epsilon_0^2 \ldots \epsilon_0^m]$ and $\delta^T := [\delta_0 \;\; \delta_0^2 \ldots \delta_0^m]$. Furthermore, each critical region is uniquely defined by a single basis, the relative interiors of critical regions do not overlap and the union of all full-dimensional critical regions is the set of feasible parameters.

The mechanics of solving a lexicographically perturbed problem can be found in [38] and require a very small amount of additional computation over the non-perturbed case.

Remark 3.13. Note that ϵ and δ are symbolic perturbations. Only their effect on the problem is considered and they are never assigned real values. As a result, there are no numerical concerns resulting from taking γ to be too small or too large, since it is at no point assigned any value.

For the simplicity of the exposition we will assume the use of *lexicographic perturbation* in the remainder of this paper, since it allows the reader to ignore the effects of degeneracy and so to focus on the main purpose of this paper: solution methods for parametric linear programming.

4. Parametric Linear Programming as Convex Hull Calculation

In this paper we survey parametric linear programming methods by categorizing them into three main approaches for computing convex hulls. Therefore, we must first interpret the computation of the bases that define full-dimensional critical regions as a convex hull calculation problem.

4.1. Dual Solution: Vertex Enumeration

The following theorem demonstrates that the goal of enumerating all bases that define full-dimensional critical regions can be re-posed as a vertex enumeration problem of an affine transform of the dual constraint polytope $\mathcal{D}$, given in (5b).

Theorem 4.1. [38] If B is a non-degenerate feasible basis of pLP (4) and λ is the resulting basic solution, then B defines a full-dimensional critical region if and only if $E\lambda$ is a vertex of the polytope $E\mathcal{D} := \{E\lambda | \lambda \in \mathcal{D}\}$.

Remark 4.2. Note that critical regions are defined as all parameters for which a given basis is optimal. It follows, therefore, that the negative normal cone of $E\mathcal{D}$ at a vertex $E\lambda$ is exactly the critical region of the basis B, $\mathcal{R}_B = -\mathcal{N}_{ED}(E\lambda)$.

4.2. Primal Solution: Facet Enumeration

In this section we consider the primal problem pLP (4a) and view the solution as an enumeration of the facets of the *epigraph*, which is defined as

$$\text{epi}\, J^* := \{(\theta, J) \in \mathbb{R}^d \times \mathbb{R} \mid \theta \in \pi_\theta \mathcal{P},\ J^*(\theta) \le J\}, \tag{13}$$

where $J^*: \mathbb{R}^d \to \mathbb{R}$ is the optimal cost of pLP (4), and $\mathcal{P}$ is the primal constraint polyhedron (5a). From the definition of pLP (4a) and (13) it can be seen that the epigraph is given by the convex polyhedron

$$\begin{aligned} \text{epi}\ J^* = \{(\theta, J) \in \mathbb{R}^d \times \mathbb{R} \mid \exists y \in \mathbb{R}^m, \\ J \ge b^T y, \quad (\theta, y) \in \mathcal{P}\} \end{aligned} \tag{14}$$

Theorem 4.3. [5,14] If pLP (4) is non-degenerate, F is a facet of the epigraph epi J^* then $\pi_\theta F$ is either a full-dimensional critical region of pLP (4) or aff $\pi_\theta F$ is the affine hull of a facet of the set of feasible parameters $\pi_\theta \mathcal{P}$.

From Theorem 4.3 we can see that if the pLP is non-degenerate, then there is a one-to-one mapping from facets of epi J^* to critical regions and boundary facets of the feasible set. Therefore, our goal of computing all critical regions can be re-stated as enumerating all the bases that define facets of epi J^*.

The epigraph is an unbounded cone if the lifting introduced in Section 3.4 has been used. While it is entirely possible with modification to apply all methods discussed in this paper to unbounded cones, it simplifies the exposition and the implementation if we first bound them. To do this, we borrow a standard technique from projective geometry, known as projective transformations or 'lift and cut'.

If we begin with the control problem pLP (3) then we first homogenize, or lift the epigraph to the form (13) as detailed in Section 3.4. We then intersect, or 'cut' it with a hyperplane that intersects each ray of the homogenization cone of epi J^*[3] in order to generate a polytope with the same combinatorial structure [64, §2.6]. The process of homogenization of the cost J^* is depicted in Fig. 1.

Throughout the remainder of the paper, we shall refer to the resulting bounded polytope as epi $\bar{J}^*$. Since the combinatorial structure has not changed, the goal of enumerating all bases of plp (4) that define full-dimensional critical regions can now be re-stated as the enumeration of all the facets of the polytope epi $\bar{J}^*$.

5. Enumeration

In the previous section we saw that the solution to the pLP (4) can either be seen as facet enumeration of the epigraph in the primal (4a), or as vertex enumeration of the polytope $E\mathcal{D}$ in the dual (4b). In both cases, the problem of parametric linear programming has been reduced to the most fundamental operation in computational geometry: the *convex hull problem*.

Given some implicit definition of a polytope P, the generic convex hull problem is the computation of an explicit *description* of P. There are several descriptions of a polytope P that are of use depending on the context. As seen in the previous section, the two that are of interest in this paper are the vertex and the facet descriptions. We also consider here a third, called the double description, which includes both the set of vertices and the set of facets of P and well as the incidence relation between the vertices and the facets.

Standard implicit definitions of polytopes would include the convex hull of a finite set of points or the

[3] The normal of such a hyperplane is any vector in the strict interior of the homogenization cone of epi J^*

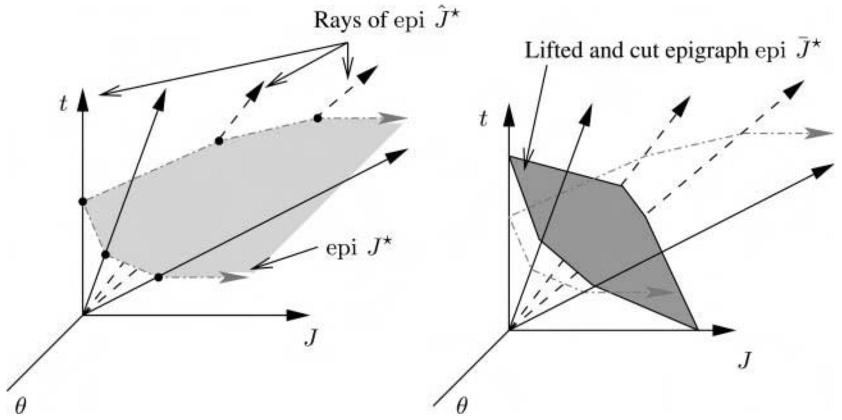

Rays of epi $\hat{J}^\star$
Lifted and cut epigraph epi $\bar{J}^\star$
t
epi $J^\star$
J
θ
t
J
θ

Fig. 1. Lift and Cut of the epigraph. (a) Homogenization of the epigraph of pLP (3). (b) Cutting of the lifted epigraph.

intersection of a finite set of halfspaces. Less common definitions would include, for example, the Minkowski sum or the extended convex hull of a set of polytopes, or the removal of redundant constraints or non-extreme points in a point set. As seen in the previous section, the definitions that are of interest in this paper are the affine transform, or projection, of the constraints of the dual pLP (4b) $\mathcal{ED}$, or the epigraph epi $\bar{J}^*$ in the primal.

The two problems of vertex and facet enumeration are equivalent through *polar duality* (see, for example [64,§2.3] and so we will focus on either vertex or facet enumeration in the following sections in order to simplify the exposition.

In this section we will survey the three established classes of convex hull computation in the computational geometry literature and we will see that most recent pLP developments in the control literature can be placed in either the first or the third class. We will then introduce a new approach, based on the second class, called the incremental methods, and show that this approach has many favorable characteristics from a control point of view. Finally, we will then survey recent new methods proposed in the control literature based on a geometric intuition.

For much of this section, we will rely heavily on the careful survey of convex hull methods given in the text [27].

5.1. Graph Traversal

The class of methods that fall under the heading 'graph traversal' can be equivalently applied to the problem of computing the vertex description, or the facet description, but not both simultaneously. We consider here some generic polytope P, which is taken to be either the implicitly defined polytope $\mathcal{ED}$, or the epigraph of the cost function epi $\bar{J}^*$.

Definition 5.1. Let P be a polytope. The graph, denoted $G(P)$, is defined over the skeleton, or vertices and edges, of the polytope P such that its nodes are the vertices of P, with two vertices joined by an arc if they form the endpoints of an edge of P. The dual graph $G^\Delta(P)$ is the graph of the polar dual of the polytope P.

Remark 5.2. The dual graph $G^\Delta(P)$ can also be interpreted as a graph whose nodes are the facets of P, with two nodes of $G^\Delta(P)$ joined by an arc if the two facets contain a common ridge. For this reason the dual graph is also often called the facet graph of P.

For simplicity we discuss here the graph of P and the problem of vertex enumeration. Facet enumeration is entirely equivalent, but is applied to the dual graph. The small differences for the specific case of parametric programming will be pointed out in the following.

The idea behind graph-based approaches is to traverse $G(P)$ in some organized fashion. The algorithm is initialized by finding a single vertex v of P and determining the set of edges Σ that contain v. At each step the algorithm takes a vertex v of the polytope and an edge e that contains v, and it finds the second vertex v' contained in e and the set of edges Σ' that contain v'. The algorithm continues until there are no edges that have not been explored, at which point all vertices of the polytope have been enumerated.

Remark 5.3. In the case of vertex enumeration, this class of algorithms are often called 'pivoting algorithms' due to the similarity with the simplex algorithm, which moves from vertex to vertex along edges. The polar dual case, in which facets are enumerated, is usually referred to as 'gift-wrapping' algorithms as the algorithm moves from one facet to an adjacent one by 'wrapping' the supporting hyperplane of the facet around the ridge that they both contain.

The following theorem proves the correctness of the algorithm, or that every vertex/facet of the polytope will be discovered by following the above procedure.

Theorem 5.4 (Balinski [2]). The graph of a polytope is connected.

There are three details that need to be determined in order to apply the algorithm:

1. How to find the edges of P that contain a given vertex?
2. How to determine the adjacent vertex given an edge and a vertex?
3. How to maintain the set of unexplored edges?

There have been various proposals for tackling these three issues for parametric programming, beginning from the early work [26] and moving to several variations proposed since then [3,29,38]. In the following section, we introduce a generic graph traversal approach, which is essentially equivalent to [26] when the problem is non-degenerate. [38] extends this to the degenerate case by lexicographically perturbing the LP. In [3,29] an approach is proposed that solves the second question above by relying on the geometry of the critical regions, rather than operating directly on the graph $G(\mathcal{D})$.

A related method was reported in [55] in which it was proposed to enumerate the entire graph $G(\mathcal{D})$ and then to test each basis to determine if it defines a vertex of $E\mathcal{D}$, which requires significantly more effort that computing directly the sub-graph $G(E\mathcal{D})$ as outlined in this section. A similar method was proposed in [57,59] for mpQPs where all combinations of active constraints are considered. [48] improves this by enumerating only the d-dimensional faces of $\mathcal{P}$ by using *parametrized vertices* [43]. All methods that work by enumerating the faces of $\mathcal{D}$ rather than the polytope $E\mathcal{D}$ are very unlikely to be efficient, as there are generally exponentially more faces in $\mathcal{D}$ than in $E\mathcal{D}$, since it is normally of much higher dimension.

Remark 5.5. Virtually all of the computational geometry literature that relates to this class of algorithms deals with the specific case of computing a facet description of the convex hull of a finite set of points, or equivalently computing the vertex description of the intersection of a finite number of halfspaces. Most of the improvements possible over the basic algorithm require that one of the descriptions of the polytope is already known. In the case of parametric linear programming, the polytope whose description is sought is only described implicitly as the linear transform, or projection, of a polytope and thus most of these improvements cannot be applied.

Graph Traversal for Parametric Linear Programming

We discuss here how to compute the vertices of the polytope $E\mathcal{D}$ for simplicity, although every operation can be equally well applied to facet enumeration of epi $\bar{J}^*$.

How to find the edges of $E\mathcal{D}$ that contain a given vertex? We begin with the first of the three points above and describe the set of edges for a given vertex of $E\mathcal{D}$. The following standard lemma gives the relationship between the tangent cone at a vertex and the edges containing the vertex.

Theorem 5.6 (See, for example [46]). If $\mathcal{Z}$ is a polytope and v is a vertex of $\mathcal{Z}$, then the edges of $\mathcal{Z}$ containing v are

$$(\text{extr } \mathcal{T}_{\text{Z}}(\text{v}) \oplus \{\text{v}\}) \cap \mathcal{Z}, \tag{15}$$

where extr $\mathcal{T}_{\mathcal{Z}}(v)$ is the set of extreme rays of the tangent cone of $\mathcal{Z}$ at v.

From Theorem 5.6 we can see that the edges of $E\mathcal{D}$ that contain a given vertex $E\lambda$, where λ is in $\mathcal{D}$, are precisely given by the extreme rays of the tangent cone $\mathcal{T}_{E\mathcal{D}}(E\lambda)$. As E is linear, we have that $E\mathcal{T}_{\mathcal{D}}(\lambda) = \mathcal{T}_{E\mathcal{D}}(E\lambda)$ [38], which allows for a direct representation of the edges containing a particular vertex of $E\mathcal{D}$, as shown in the following theorem.

Corollary 5.7. [38] If B is a basis of pLP (4) and λ is the basic dual solution such that $E\lambda$ is a vertex of $E\mathcal{D}$, then the edges of $E\mathcal{D}$ that contain $E\lambda$ are

$$(\text{extr cone}\bar{E} \oplus \{E\lambda\}) \cap E\mathcal{D}, \tag{16}$$

where $\bar{E} := E_{*,N} - E_{*,B}\beta A_{*,N}$ is the reduced cost.

A direct implementation of Corollary 5.7 provides the list of edges that contain a given vertex $E\lambda$. The computation of the extreme rays of a pointed cone that is represented as the positive combination of a finite set of rays requires one linear program to be computed per column of $\bar{E}$ and is a standard operation referred to as *extreme point computation* or *redundancy elimination.*

Testing if $\bar{E}_{*,i}$ is a generator of an extreme ray of cone $\bar{E}$ can be done using the following linear program (see, for example [21]):

$$\begin{aligned} \rho(i) := \text{minimise} & -\bar{E}_{*,i}^T\theta \\ \text{subject to } & \bar{E}_{*,\{1,\dots,n\}\setminus\{i\}}^T\theta \geq 0 \\ & \bar{E}_{*,i}^T\theta \geq -1 \end{aligned} \tag{17}$$

where the ray cone $\bar{E}_{*,i}$ is extreme if $\rho(i) < 0$.

The approach presented here for computing the edges containing a given vertex is used in all of the currently proposed methods that use a graph traversal approach [3,26,38], including the many proposals for parametric quadratic programs (pQPs) [3,10,29,58, 62,63] and parametric linear complementarity problems (pLCPs) [36]. Most of these approaches tackle the problem in the framework of the primal pLP (4a) rather than the dual presented here, in which case the redundancy elimination is done for the critical region, which is precisely the polar dual of cone $\bar{E}$. As a result, the extreme rays of cone $\bar{E}$ are precisely the irredundant facets of the critical region and the process of determining them is entirely equivalent to the redundancy elimination operation utilized in most of the literature on pLPs.

Remark 5.8. There have been several heuristic proposals for reducing the computational load of this redundancy elimination [17,28,51,61]. These methods can offer a significant benefit in some cases and on certain problems, but in the general case it is still required to solve (17) for every column of $\bar{E}$ for every vertex of $E\mathcal{D}$.

How to determine the adjacent vertex given an edge and a vertex? We are now ready to tackle the second implementation question given above. The following theorem demonstrates that this adjacent vertex can be computed using a single linear program in the case of parametric linear programming.

Theorem 5.9. [38] If B is a feasible basis of pLP (4) and λ is the resulting basic dual solution, $E\lambda$ is a vertex of $E\mathcal{D}$ and $r = \{\bar{E}_{*,i}\alpha | \alpha \geq 0\}$ is an extreme ray of $\bar{E}$, then the adjacent vertex of $E\lambda$ contained in the edge $e = (r \oplus \{E\lambda\}) \cap E\mathcal{D}$ is $E\lambda'$, where λ' is the optimizer of the LP

$$\max_{\lambda}\{\bar{E}^T_{*,i}E\lambda | \lambda \in \mathcal{D}, \quad \lambda_j = 0, \quad \forall j \notin S\}, \tag{18}$$

where $S := \{j | \exists \rho \geq 0, \quad \bar{E}_{*,i} = \rho \bar{E}_{*,j}\}$ and $\bar{E} := E_{*,N} - E_{*,B}\beta A_{*,N}$ is the reduced cost.

The LP given in the statement of the theorem maximizes in the direction of the ray r, while restricting $E\lambda$ to be on the edge of interest.

Two proposals have been made in the literature to handle this second question and LP (18) can be seen as an extension of these ideas. In [63] the authors propose, for multiparametric quadratic programs, to test all combinations of possible active constraints in order to determine which defines the adjacent basis. While this is in general combinatorial, it can be applied to mpLPs and be efficient in the case where the set S in (18) contains almost all of the constraints, which is the case when the edge is non-degenerate. In the non-degenerate case, LP (18) reduces to a one-dimensional LP and hence the same test as proposed in [63], whereas in the degenerate case, LP (18) gives an efficient method to find the adjacent vertex rather than the combinatorial test proposed in [63].

The second approach [3,29] proposes to solve the LP

$$\max_{\lambda}\{(\theta_\circ + \gamma \bar{E}_{*,i})^T E\lambda | \lambda \in \mathcal{D}\}, \tag{19}$$

for some $\theta_\circ$ on the i^{th} facet of the critical region $\mathcal{R}_\mathcal{B}$ and some small, positive γ. For sufficiently small γ, the parameter $\theta = \theta_\circ + \gamma \bar{E}_{*,i}$ will be in the adjacent critical region and therefore will find the adjacent vertex λ'. There are two difficulties with this proposal. First, one must compute a point $\theta_\circ$, which requires an LP and second, it is not clear how to choose γ sufficiently small in order to ensure that $\theta_\circ + \gamma \bar{E}_{*,i}$ will be in the adjacent critical region without causing numerical problems in LP (19).

How to maintain the set of unexplored edges? The third question has not been studied extensively in the control literature as it does not generally cause a significant impact on the performance for small problems and hence only three proposals have been made.

The first is to store a description of each critical region $\mathcal{R}$ as it is discovered [3,29]. As each new region is found, a point is computed in its interior and then this point is tested against all known regions for containment. This test is clearly linear in the number of critical regions, and since it is executed for each region discovered, results in a quadratic time algorithm that requires linear space. The second proposal is to store the discovered bases as sets of integers and then to test each new basis against this list [25,38]. Using an AVL tree (see, for example [52]), this test can be computed in logarithmic time, but is still executed a linear number of times and requires linear space. The final proposal [37] is to store the set of unexplored edges directly, rather than the explored vertices, by recording the set S from (18), which can be shown to be unique. Again using an AVL tree, such a test can be implemented in logarithmic time and linear space in the number of edges, although it is possible for the number of edges to greatly outnumber the number of vertices. The benefit of this method is that LP (18) does not need to be solved in order to test if a given edge has already been explored.

A method was introduced in [39] that applies the *reverse search* [1] paradigm to parametric linear programming. This approach defines an ordering on the edges of $E\mathcal{D}$ based on the simplex algorithm, which induces a tree structure over the vertices of the graph $G(E\mathcal{D})$. All of the elements of this tree can then be enumerated in linear time with no requirement to maintain the set of unexplored edges or vertices, resulting in a constant space complexity. A second benefit arising from the tree structure is that the algorithm can be perfectly parallelized, or distributed across multiple machines without any requirement for communication between them. These benefits are gained at the cost of an increased overhead when computing the edges that contain a given vertex, which generally makes this method well-suited to larger problems only.

5.2. Beneath/Beyond: An Incremental Approach

In this section we will introduce a new method for parametric linear programming based on the second common approach for convex hull calculation in the computational geometry literature: the beneath/beyond method.

The earliest convex hull approach was developed by Fourier in 1824 and has been rediscovered and improved many times since and given various names including the beneath/beyond [4,30,40] and the double description [23,45] methods. For many problems it, or one of its derivatives, is still the best approach.

We will consider in this section the facet enumeration problem of the primal pLP (4a) because as we will see below, this can generate a stabilizing approximation of the problem. The name generally given to such an incremental method that solves the facet enumeration problem is beneath/beyond, which is what we shall call the following proposed approach.

Given an ordered set of points $V = (p_1, \ldots, p_N)$ the incremental methods successively compute a description of $P_i := \operatorname{conv} V_i$ from the description of conv V_{i-1}, where $V_i = (p_1, \ldots, p_i)$ is the first i points.

A point p is said to be *beyond* a facet F if the supporting hyperplane aff F separates P and p, and *beneath* otherwise. We can now introduce the following slightly simplified version of Günbaum's classic beneath/beyond theorem [31,Th. 5.2.1].

Theorem 5.10 (Simplified Beneath/Beyond [4]). If P is a polytope in $\mathbb{R}^n$ and p is a point in $\mathbb{R}^n \backslash P$, then F is a facet of $\operatorname{conv}(P \cup \{p\})$ if and only if

1. F is a facet of P and p is beneath F or
2. F is not a facet of P and its vertices are p and the vertices of a ridge $R = F_1 \cap F_2$ of P where p is beneath F_1 and beyond F_2.

The beneath/beyond method is essentially a direct implementation of Theorem 5.10. At each iteration of the algorithm, we are given a point p_i as well as a double description of $P_{i-1} = \operatorname{conv} V_{i-1}$ and the goal is to compute a double description and facet graph of the polytope $P_i = (\operatorname{conv} P_{i-1} \cup \{p_i\})$. From Theorem 5.10, it is clear that P_i consists exactly of the 'old' facets that p_i is beneath plus a set of 'new' facets of the form conv $(R \cup \{p_i\})$, where R is a 'horizon ridge' of P_{i-1}, which means that R is contained in one facet beneath p_i and one beyond.

Two questions must be answered in order to implement this algorithm:

1. How to find the beyond facets and the horizon ridges?
2. How to compute the new facets?

In the case of parametric linear programming, there is one additional question that must be answered:

3. How to compute the next point to add p_i?

The simple process of the beneath/beyond algorithm is depicted in Fig. 2.

Raković [53] proposed an iterative idea for computation of the projection of a polytope. At each stage of the algorithm, an LP is solved for each facet of P_{i-1} that returns a new set of vertices of the projection. The convex hull is then computed anew for the entire set of points discovered so far. The method proposed in this section can also be applied to the computation of projections, and would improve on the method of [53] by not discarding the information used to compute the convex hull of P_{i-1} at each iteration.

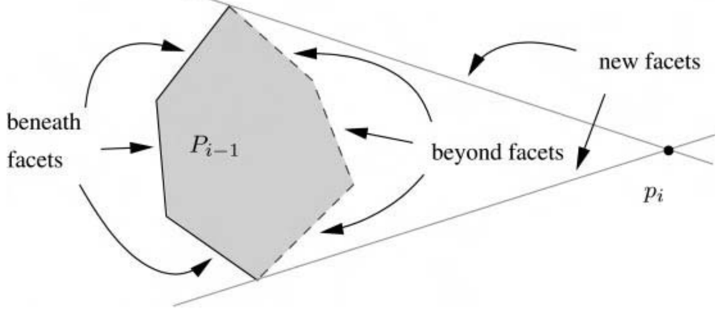


Fig. 2. Update process of the beneath/beyond algorithm.

5.2.1. Beneath/Beyond for Parametric Linear Programming

We follow the development given in [27, §22.3.1] in order to answer these questions while extending it to handle the more general case of parametric linear programming. In the following we consider the problem of facet enumeration of a set of points $V = \{p_1, \ldots, p_N\} = \text{extr epi}\, \bar{J}^*$.

How to find the beyond facets and the horizon ridges? The most obvious approach for classifying each facet F of P_{i-1} as beneath or beyond is to simply multiply p_i by the hyperplane aff F and consider the resulting sign. This is the approach taken in Kallay's beneath/beyond method [40] and in the double-description method of Motzkin et al. [45]. However, if we assume that along with the inequalities describing the facets of the polytope P_i, we also store the facet graph[4] $G^{\Delta}(P_{i-1})$, then a significant improvement can be made.

The set of facets that p_i is beyond, forms a connected subgraph of $G^{\Delta}(P_{i-1})$ [27, §22.3.1], and therefore can be easily enumerated using a standard graph search method, such as depth-first. Assuming that initially a facet that the point p_i is beyond is given, the beneath/beyond test procedure starts from its neighbors and propagates further through the graph of facets until all facets that the point p_i is beyond are identified. It follows that the only facets that must be tested to determine whether p_i is beneath or beyond are exactly those that it is beyond. The boundary arcs of this sub-graph then form the set of horizon ridges, since they join the beyond facets to the beneath.

Remark 5.11. A set of points in d-dimensions is called geometrically degenerate, or not in general position, if there is a facet of their convex hull that contains more than $d+1$ points. Such a situation manifests itself in the beneath/beyond algorithm when a new point is neither beneath, nor beyond a given facet, but is actually contained in its supporting hyperplane.

In the following section, the beneath/beyond algorithm will be used as a method to compute an approximate control law. For this purpose, it is required that each facet of P_i is a simplex, or contains exactly $d+1$ vertices. This requirement of general position can be simulated by returning 'beneath' whenever a point is in fact contained in the supporting hyperplane of a facet.

How to compute the new facets? Given a point p_i and two adjacent facets F_1 and F_2 of P_{i-1}, one beyond and one beneath respectively, the goal is to compute a single hyperplane that contains both $R := F_1 \cap F_2$ and the point p_i. Let the facets be defined by their supporting hyperplanes as aff $F_1 = \{x \mid a_1^T x = b_1\}$ and aff $F_2 = \{x \mid a_2^T x = b_2\}$. The hyperplane that contains the ridge R and the point p_i is simply the positive combination of the supporting hyperplanes of F_1 and F_2 that satisfies the equation $(\nu_1 a_1 + \nu_2 a_2)^T p_i = \nu_1 b_1 + \nu_2 b_2$ for some $\nu_1, \nu_2 \geq 0$. Solving for ν_1, ν_2 gives an equation for the new facet, which is unique up to positive scaling

$$\begin{aligned} \text{aff } F_{\text{new}} = \{x | ((a_1^T p_i) a_2^T - (a_2^T p_i) a_1^T) x \\ = ((a_1^T p_i) b_2 - (a_2^T p_i) b_1)\}. \end{aligned} \tag{20}$$

If we compute the facet graph of P_i, which as pointed out above, can speed the computation for determining which facets are above and which beyond, then the new facets must be incorporated into the graph $G^{\Delta}(P_{i-1})$ in order to form $G^{\Delta}(P_i)$. Clearly, we must add the new facets as new nodes in the facet graph, and remove the beyond facets from it. Arcs are then added between each new facet and the beneath facet that was used to form it. Finally, arcs must be added between the new facets that contain a common ridge. These arcs can be easily determined by looking at the vertices of the new facets: two of them share a ridge and are therefore adjacent if and only if they have at least dim P vertices in common. Each boundary arc can be added in fixed time, and determining the set of arcs between the new facets can be done in time proportional to the number of newly added facets.

How to compute the next point to add? It is here that the beneath/beyond method applied to parametric linear programming diverges from the standard approach used for convex hull computation. In the case considered here, the algorithm does not have the list of vertices *a priori*, but rather they must be computed during the execution of the algorithm.

Theorem 5.12. Let V be a subset of vertices of the polytope P. If $h := \{x | a^T x = b\}$ is the affine hull of a facet of conv V, where a is an outward facing normal, and x^* is the optimizer of the LP

$$x^* := \arg\max_x \{a^T x | x \in P\} \tag{21}$$

then h is a supporting hyperplane of a face of P if and only if $x^* \in \text{conv} V$.

Proof. Recall that a hyperplane h defines a face of P if and only if $P \subset \{x \mid a^T x \leq b\}$, which is true if and only if $x^* \in \text{conv } V$. □

Remark 5.13. Note that Theorem (5.12) can be as readily applied to an implicitly defined polytope such as the cut epigraph defined in Section 4.2.

[4] Recall that the facet graph is the graph of the polar dual, or equivalently a graph whose nodes are the facets of P_{i-1} and whose arcs connect two nodes if the facets share a common ridge.

A direct implementation of Theorem 5.12 gives a simple procedure for finding the next vertex p_i to add in order to increment from P_{i-1} to P_i. It also provides a termination condition, as stated in the following corollary.

Corollary 5.14. Let V be a set of vertices of the polytope P. If the optimizer x^* of LP (21) satisfies condition of the Theorem 5.12 each facet of conv V, then conv $V = P$.

If the goal is to compute the polytope exactly, then the order in which Theorem 5.12 is applied to the facets is not important, although this ordering can significantly impact the complexity of the increments P^i. However, as one will see in Section 5.2.2, if the goal is to compute an approximate control law then the ordering can make a significant impact.

5.2.2. Approximation of pLPs for Control

In this section we will go into some detail on how the beneath/beyond method can be used to develop a stabilizing, approximate and invariance-inducing control law with properties that are useful from a control point of view. We will now assume that the above method is not used for computation of a generic implicitly defined polytope, but rather is put to use specifically computing the polytope epi $\bar{J}^*$ and the resulting control law, which is derived from the optimal bases at the vertices, as detailed below.

As outlined in Section 4.2, there is a mapping from vertices of epi $\bar{J}^*$ to the vertices and rays of the epigraph of pLP (3), where we assume that the pLP results from a control problem and therefore was homogenized as shown in Section 3.4.

In the following, we assume that the vertices of P_i were de-homogenized in order to get the states, optimal costs and inputs used in this section. This simple procedure is also covered in Section 3.4 and consists of simply dividing through by the homogenization variable. Note also that the *combinatorial structure* of the epigraph is maintained through the homogenization process and therefore the vertices that define a facet of P_i are precisely those that define a facet of the de-homogenized polytope [64, §2.6].

Approximation. At each iteration of the algorithm, the polytopes P_i form inner approximations of the polytope epi $\bar{J}^*$ and therefore an upper approximation of epi J^*. We now define an approximate control law from the de-homogenized vertices of this approximation P_i.

Definition 5.15 (Approximate simplical control law). Let $Q = \{(x_1, J_1), \dots, (x_n, J_n)\} \subset \mathbb{R}^{d+1}$ be a set of vertices of the epigraph of pLP (9) and u_i be the optimizer for vertex (x_i, J_i). If $F = \{(x_{F_1}, J_{F_1}), \dots, (x_{F_{d+1}}, J_{F_{d+1}})\} \subset Q$ are the vertices contained in a facet of the lower convex hull of Q, then the approximate critical region defined by F is $\tilde{\mathcal{R}}_F := \operatorname{conv} \pi_x Q$ and the control law is given as

$$\tilde{u}(x) := U_F X_F^{-1} \begin{pmatrix} x \\ 1 \end{pmatrix}, \quad \forall x \in \tilde{R}_F, \tag{22}$$

where

$$U_F := \left[u_{F_1} \dots u_{F_{d+1}} \right]$$

$$X_F := \begin{bmatrix} x_{F_1} & \cdots & x_{F_{d+1}} \\ 1 & \cdots & 1 \end{bmatrix}$$

Furthermore, we define the approximate value function as $\tilde{J}(x) := b^T \tilde{u}(x)$.

Definition 5.15 simply defines the approximate control law as the interpolation of the optimal control action given at the vertices of each approximate critical region. Note that we have assumed that all facets of the epigraph are simplical. If the beneath/beyond procedure discussed above is followed, then this assumption will be met, which also ensures that the inverse in (22) exists.

The following theorem shows that from any such inner approximation we can immediately derive a suboptimal, feasible and continuous controller.

Theorem 5.16. If $\tilde{u}(x)$ is the approximate control law of Definition 5.15, then it is

- feasible: $(x, \tilde{u}(x)) \in \mathcal{P}, \quad \forall x \in \cup \tilde{\mathcal{R}}_F$
- defined over non-overlapping critical regions: $\operatorname{int} \tilde{\mathcal{R}}_F \cap \operatorname{int} \tilde{\mathcal{R}}_{F'} = \emptyset, \quad \forall F \neq F'$
- defined over a convex set: $\cup \tilde{\mathcal{R}}_F$ is convex

Proof. Follows directly from Definition 5.15 and convexity of the primal constraints.

One of the most important properties of the proposed approximation scheme is that in each iteration of the beneath/beyond algorithm one new vertex of the epigraph is discovered and therefore the optimal solution is found in a number of iterations exactly equal to the number of vertices and rays in the epigraph epi J^*.

Theorem 5.17. If there are N vertices and rays in the epigraph of pLP (4), then $\tilde{u}(x) = u^*(x)$ after exactly N iterations of the beneath/beyond algorithm.

Stability and Invariance. We first give the standard condition under which an approximate control law is stabilizing and then discuss the invariant set that the approximate control law induces. The following theorem gives a condition for the approximate control law to be stabilizing.

Theorem 5.18. [56] Let J^*: $\mathbb{R}^d \to \mathbb{R}$ be the value function of the optimal control problem (2) and a Lyapunov function. Approximate value function $\tilde{J}$ is a Lyapunov function if for all $x \in \cup\tilde{\mathcal{R}}$ $\tilde{J}(x) - J^*(x) < l(x, \tilde{u}(x))$, where $l : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ is the stage cost (2).

The condition of Theorem 5.18 is insufficient to ensure the stabilizing character of the approximate control law $\tilde{u}(x)$, unless feasibility of the close loop system is guaranteed for all time. The approximate control law defined by the beneath/beyond procedure is not defined over the entire feasible set $\pi_\theta\mathcal{P}$ unless all of the vertices on the boundary of the feasible region have been added. As a result, the approximate control law cannot guarantee that the system remains feasible for all time and all $\theta \in \pi_\theta\mathcal{P}$, as it is not necessarily a control law that induces *invariance* over its entire domain. However, if we assume that the beneath/beyond algorithm has been run for enough iterations to satisfy the conditions of Theorem 5.18, then we have that $\tilde{J}$ is a Lyapunov function and since level sets of Lyapunov functions are invariant [13], this leads to the following corollary.

Corollary 5.19. If Q is a set of vertices of the epigraph of pLP (2) that satisfy the conditions of Theorem 5.18 and $J^{\min} := \min\{J_i | \exists x_i,\ x_i, J_i) \in \operatorname{extr} \pi_x Q\}$, then the set

$$\tilde{I} = \{x \in \operatorname{conv}\pi_x Q \mid \tilde{J}(x) \le J^{\min}\} \tag{23}$$

is invariant under the control law $\tilde{u}(x)$.

Proof. Level sets of Lyapunov functions are invariant [13]. The condition in the corollary gives the largest level set of the Lyapunov function $\tilde{J}$ by considering the vertices at the extreme of the approximate epigraph.

Remark 5.20. Corollary 5.19 provides the description of a set in which the approximate control law is invariant. Such a set would be of use for analysis, although for implementation it is not necessary to explicitly compute this set.

Remark 5.21. The extreme vertices in Corollary 5.19 can be immediately identified from the approximate polytope P_i as follows. Recall from Section 4.2 that the homogenization and cut procedure introduces a finite vertex v^∞ in the cut polytope that corresponds to infinity in the homogenized one. Since the combinatorial structure is maintained during the lift and cut procedure, the vertices of P_i that are adjacent to v^∞ correspond to the vertical rays of P_i that define the boundary of the domain of the approximate control law.

Which point to choose next? We now re-visit this implementation question, with the specific aim of approximation of a control law. The goal is to determine an approximate control law defined over a small number of regions that is both stabilizing and invariant in a sufficiently large region of the state space. We propose to maintain a list of potential vertices that could be added at each iteration of the beneath/beyond algorithm, and then at each iteration to choose either a vertex corresponding to a vertex of the epi $J^*(x)$ or a vertex corresponding to a ray of epi $J^*(x)$. In the former case an increase of the feasible set is achieved, while in the latter one the upper bound of the error between the approximate and the optimal value function is reduced, moving the approximation error towards the condition of Theorem 5.18. A specific policy can be implemented, giving a preference to the best increase in the size of the invariant set or the largest improvement in the approximation error, where ‘best’ is a tuning knob defined by the user. Note that the Hausdorff distance between the exact and the approximate epigraph of the value function reduces in each iteration of the beneath/beyond algorithm. However, the maximum of the value function approximation error may increase in some iterations as new parts of the feasible set are added to the approximate solution, what is demonstrated on an example in section 7.

The list of vertices S_i that we consider are those given by Theorem 5.12 when it is applied to *all* facets of P_{i-1} which are not facets of P. We then order the vertices S_i based on a function of the amount that adding a particular vertex would decrease the error between $\tilde{J}$ and J^*, and hence move closer to stability and optimality, and the amount that adding the vertex would increase the value of $J^{\min}$ in Corollary 5.19.

Remark 5.22. Note that in each iteration we have only to solve LP (21) for each *new* facet discovered when updating from P_{i-1} to P_i, since it has already been solved for all other facets of P_i. Furthermore, by checking if the optimizer is already in the list V^i of already computed vertices before solving LP (21) for a new facet we avoid duplicating any work.

5.3. Primal/Dual

The primal/dual method discussed here was first developed in [16] for the purpose of vertex enumeration of polytopes and can be seen as a combination of the graph traversal and incremental approaches. In this section, we discuss its extension to the solution of parametric linear programs, which was presented in [35].

Primal/dual enumeration is so named since it simultaneously enumerates both the primal (vertex) and dual (halfspace) descriptions. In other words it solves the double-description problem. As in the beneath/beyond method, at the i^{th} step of the algorithm, a double description is stored of the polytope $P_i = \text{conv}V_i$, where V is the set of vertices of $\mathcal{ED}$. Using the procedures of Section 5.1.1, a vertex $v \notin V_i$ can be found that is adjacent to one of the vertices in the set V_i. The set V_i is then updated to $V_{i+1} = V_i \cup \{v\}$ and the double description is also updated using the same approach as Section 5.2.

Recall from Theorem 5.7 that the adjacent vertices of a given vertex v are found by computing the extreme rays of the tangent cone at v. For parametric linear programming, determining these extreme rays requires a redundancy elimination operation (17), which can often take more than 98% of the computation time for graph traversal approaches [37]. In the primal-dual approach, the halfspace description is used to provide a sufficient condition for redundancy of a ray of the tangent cone, which can be easily tested.

We note that the polytope P_{i+1} is an inner approximation of the polytope $\mathcal{ED}$. It follows that if a ray r of the tangent cone $\mathcal{T}_{\mathcal{ED}}(v)$ for some vertex v of $\mathcal{ED}$ intersects the interior of P_{i+1}, then it also intersects the interior of $\mathcal{ED}$ and is therefore not an extreme ray. This notion is formalized in the following theorem.

Theorem 5.23. [35] Let $S \subseteq \text{extr}P$ be a set of vertices of the polytope P such that dim conv $S = \dim P$ and $H := \{x | Gx \leq g\} = \text{conv } S$. If $v \in S$ is a vertex of P, then a ray $r = \{\gamma\alpha | \alpha \geq 0\}$ is a redundant ray of the tangent cone $\mathcal{T}_P(v)$ if $G_{i,*}\gamma \leq 0$ for all i such that $G_{i,*}v = g_i$.

Theorem 5.23 can now be used to test the possibly redundant rays of $\mathcal{T}_{\mathcal{ED}}(E\lambda) = \text{conv}\bar{E}$. Since this test can only prove redundancy and not irredundancy, if the conditions of the theorem are not met, then the linear program (17) must still be solved. It will be seen in Section 7 that there are problems of interest to control for which this redundancy test offers a significant improvement over standard graph traversal approaches, although for larger problems the complexity of computing the halfspace description can significantly outweigh the benefit gained.

5.4. Geometric

All of the methods that have been considered so far have focused on computing a sub-graph of $G(\mathcal{D})$ which has appropriate properties. Proposals have appeared in the control literature in the last few years that operate directly on the critical regions of the solution rather than the graph of the constraints. This section will outline these methods.

Facet Flipping [14]. The basic operation of this algorithm takes some randomly selected value of the parameter θ^r in a set R, and computes the constraints that are active at the optimal value, which can be discovered by solving the pLP (4) for $\theta = \theta^r$. The critical region $\mathcal{R}_\nabla$ containing θ^r can be written down from these active constraints, as given by Definition 3.5. The algorithm then computes the *complement* of $\mathcal{R}_r$ as $R \backslash \mathcal{R}_r$, which is represented as the union of a finite number of polytopic regions since the compliment is non-convex. This operation is then simply repeated recursively in each of these regions of the complement until the entire feasible set is covered.

Hierarchical Approximation [6,8,9,19]. This approach provides a general approximation method for parametric convex optimization problems. In the first phase of the algorithm, a set of feasible parameters is computed and their convex hull is tessellated.[5] An approximate cost and optimizer is defined by interpolating across the simplices of the tessellation as in Section 5.2. At each iteration of the algorithm a simplex S is chosen and the point $x^* \in S$ is determined such that the error between the optimal cost and the approximate cost is maximized. The simplex S is then further sub-divided into simplices by *pushing* the point x^* into the tessellation, which means to add a new simplex $F \cup \{x^*\}$ for each facet F of S. This simple procedure is then repeated until the maximum error is below some desired threshold.

Orthogonal Box Decomposition [32,33]. This approach is similar to the hierarchical approximation method discussed above in that it sub-divides the feasible set of parameters into a hierarchy of fixed shapes. In this case, the shapes are axis-aligned hypercubes and at each iteration of the algorithm a hypercube is selected and the optimal control law is computed at each of the 2^d vertices and the center. Each quadrant of the hypercube is then subdivided if an error bound is not satisfied by the resulting approximate cost. The main strength of this approach is the structure of the resulting controller, since such an axis-aligned box decomposition can be searched extremely rapidly during online controller computation.

[5] A tessellation is a set of non-overlapping simplices that cover the convex hull. One commonly used algorithm that generates tessellations is the Delaunay triangulation (see, for example [27, Chapter 23]).

6. Complexity Comparison

6.1. Graph Traversal [3,26,29,38]

One of the primary objectives in computational geometry is the development of algorithms that are *output sensitive*, which means that the complexity is a function of only the size of the input and the size of the output. If we define the size of the output as the number of critical regions, then methods based on a graph traversal paradigm are the only known approaches that are able to achieve output sensitivity. In Section 5.1.1 it was seen that exactly one operation, consisting of solving LP (18) once and LP (17) n times, is done per critical region, or equivalently for each vertex of the dual $\mathcal{ED}$ or facet of the primal epi $\bar{J}^*$.

In big-$\mathcal{O}$ terms, the graph traversal approaches are distinctly superior to all other proposed methods for one reason, which is that they compute only one representation of the solution. Since only one representation is required, any method that computes a second is doing work that is not directly useful. Furthermore, the worst case complexity of the second representation is exponentially worse than the first. However, the best-case situation is that the second representation is exponentially simpler than the first and therefore it is impossible to say in general *a priori* if anything is gained or lost through a double-description approach and hence any such statement will be problem dependent.

The graph traversal approach is inherently exact, in the sense that a partial solution is not meaningful. Stopping after having only traversed part of the graph does not give a solution with useful properties. There have been some recent attempts, however, to apply this approach to approximation of control laws by modification of the problem statement. In [7] the authors relax the optimality criteria for a strictly convex parametric quadratic program and then solve the relaxed problem exactly using a graph traversal paradigm, which may generate a reduced complexity controller. Due to the reliance on an exact solution of the relaxed problem, such an approach requires that the allowed approximation error be specified *a priori*. Once this is done the complexity of the approximate controller is fixed to some unknown value. Furthermore, the resulting approximate control law is, in general, discontinuous, what can cause undesired chattering behaviour of the controller.

6.2. Incremental

There are two weaknesses to an incremental approach to computation. First, as stated in the previous section, the double-description that is required for an incremental algorithm can be exponentially more complex than the single description that is desired. However, this exponential relationship is heavily dependent on the structure and the dimension of the problem. The majority of practical control cases that are reported in the literature are of fairly small dimension and so it is not currently clear how much of a limit this requirement is for the class of problems of interest to control. Second, it was shown in [15] that given a set of vertices V and a subset $V^s \subseteq V$, it is possible for the number of facets in the convex hull convV^s to exponentially outnumber those in conv V. As a result, no incremental algorithm can ever be output sensitive.

Two benefits arise from using an incremental approach to pLP computation. The first is that no redundancy elimination, which is the linear program shown in (17), is required. In all other approaches, this redundancy elimination operation must be executed at least once for each critical region and requires the solution of a number of low-dimensional linear programs equal to the number of constraints in $\mathcal{P}$. Current efficient algorithms that follow a graph-traversal paradigm spend over 98% of their execution time solving these redundancy elimination LPs, which is a major gain for the incremental methods, as is shown for some examples in Section 7 below.

The second strength is the ability for approximation. The algorithm can be stopped at any iteration for which the requirements of the stability Theorem 5.18 are satisfied, and a stabilizing, feasible and invariant approximate controller will result. This allows for a direct trade-off between the complexity of the controller, the size of the region of attraction and the performance of the system, which is a very useful set of tuning knobs.

6.3. Primal/Dual [35]

The primal/dual approach gains some of the strengths of the incremental method, in that it has a reduced requirement for redundancy elimination, but it also gains the main weakness, which is that it must compute a double-description. A second mark against the primal/dual paradigm is that while it is an incremental algorithm, the increments do not make for useful approximations, since it still enumerates the critical regions in a graph traversal fashion. It follows that while the primal/dual approach can be of benefit in very small problems that happen to have a structure that offers a significant reduction in the redundancy elimination operation, it is generally the case that a method based on graph traversal or the incremental paradigms would be superior.

6.4. Geometric

The geometric methods hold one commonalty in that their complexity is neither a function of the size of the input, nor the size of the output, nor the size of the double-description. Both methods artificially subdivide the optimal critical regions, resulting in smaller regions whose number is difficult to approximate or bound.

Facet Flipping [14]. For each sub-division of a critical region the facet-flipping approach of [14] does the same number of computations in terms of redundancy elimination (17) and determining the active constraints in the critical region (18) as the graph traversal methods, however, extra work is required in the computation of a set-difference. The main difficulty with the facet-flipping methods is that the computation of the set difference can subdivide critical regions, causing them to be discovered multiple times. As a result, these methods must take more computation than the graph traversal approaches, since they do at least the same work and likely much more.

Hierarchical Approximation [6,8,9,19]. The method introduced in [6] provides an approximate solution by computing the optimal solution at a set of feasible parameters and interpolating between them. In this it is similar to, and pre-dates that based on the beneath/beyond approach introduced in Section 5.2. There are, however, two significant differences. First, the domain of the approximate control law must be chosen beforehand and cannot grow during the execution of the algorithm. Second, the initial tessellation imposes a structure on the solution that is not related to the critical regions and as a result, the points that are added at each iteration of the algorithm are not necessarily vertices of the epigraph. It follows that in order to compute the exact solution, the algorithm solves a number of linear programs that is larger than the number of vertices in the epigraph. In comparison, the beneath/beyond approach of Section 5.2 solves exactly one LP per vertex and therefore provides both a simpler and a more efficiently computed solution. However, if the goal is to compute an approximate solution then it is difficult to determine which approach will have a lower complexity for a given error bound.

Orthogonal Decomposition [32,33]. The main strength of the orthogonal decomposition is the structure of the resulting control law, which is defined over a hierarchy of hypercubes. This axis-aligned structure allows for an extremely rapid online computation time. However, the offline computation requires the solution to 2^d LPs for each hypercube for a problem with state dimension d. Clearly, this restricts the method to very low-dimensional problem, although it may well be effective for problems of 2 or 3 dimensions that require very fast online computation and have a configuration that makes them well-suited to this type of decomposition.

7. Computational Example

In this section we will illustrate the behavior of the approximate computational scheme based on the beneath/beyond approach introduced in Section 5.2.

The system we consider is a four dimensional discrete-time open-loop stable LTI system, given as an example in the Multi-parametric Toolbox (MPT) [42]:

$$x^+ = \begin{bmatrix} 0.7 & -0.1 & 0 & 0 \\ 0.2 & -0.5 & 0.1 & 0 \\ 0 & 0.1 & 0.1 & 0 \\ 0.5 & 0 & 0.5 & 0.5 \end{bmatrix} x + \begin{bmatrix} 0 & 0.1 \\ 0.1 & 1.0 \\ 0.1 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} u. \quad (24)$$

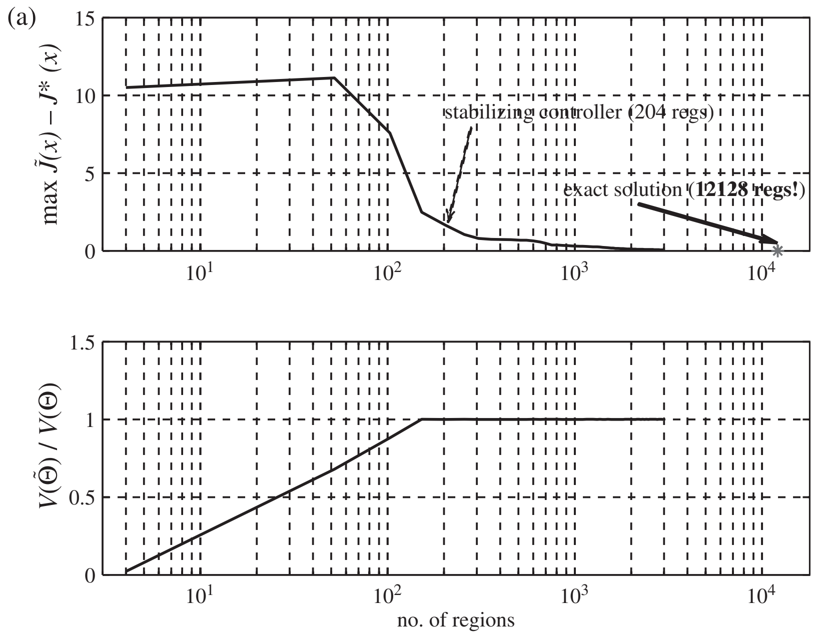
States and control inputs are constrained: $\|x\|_\infty \leq 5$ and $\|u\|_\infty \leq 5$ and the goal is to compute an approximate solution to the optimal control problem (2) to a certain accuracy such that the stability condition given by Theorem 5.18 is satisfied. The norm used in the cost is ∞-norm and the terminal set $\mathcal{X}_\mathcal{F}$ is chosen as a contractive polyhedral invariant set containing the origin, with the corresponding set-induced Lyapunov function defining the terminal cost $V_N(x_N)$ [12].

The system is a good candidate example for showing how complex an exact explicit solution may become in some cases. The exact solution to the problem with a prediction horizon of $N = 5$ comprises 12,128 critical regions [37].

All of the results presented here were obtained using the Multiparametric Toolbox [41]. Approximate controllers have been computed by specifying the desired number of controller regions starting from 4 up to 3,000. The change of the approximation error with respect to the number of regions is shown on Figure 3a.

One can see that the error decreases quite rapidly in the initial phase of the algorithm. Intervals on the plot where the approximation error increases with the increasing number of regions indicate the expansion of the set of feasible states.

Figure 3b shows the time trajectory of the closed loop system for the approximate stabilizing controller consisting of 357 controller regions and the trajectory obtained by the optimal controller (12,128 regions).



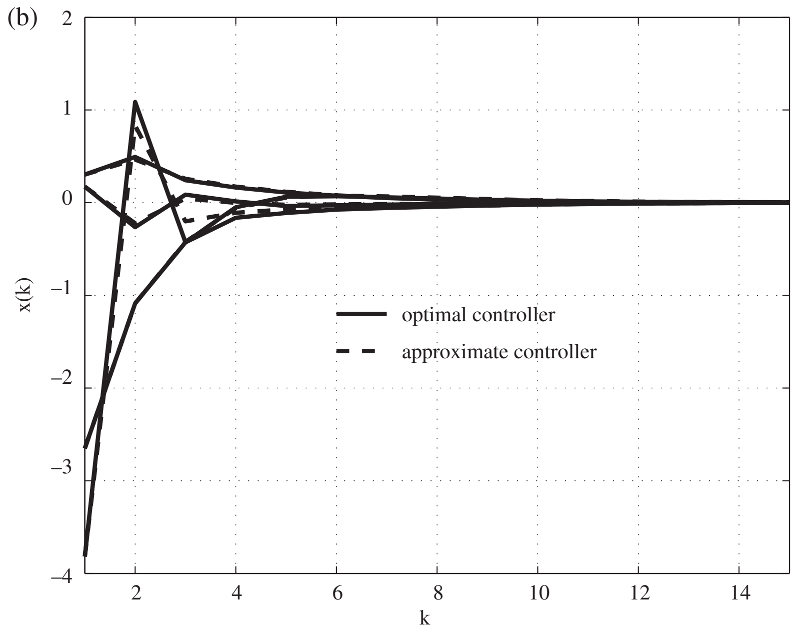


Fig. 3. Beneath/Beyond approximation of the explicit controller for 4 dimensional system. (a) Upper figure shows value function approximation error vs the number of regions in an approximate (beneath/beyond) controller. Lower figure shows ratio between the volumes of an approximate and exact feasible set. (b) Time trajectory of the approximate controller (357 regions) vs response of the exact optimal controller (12,128 regions!) for a random initial condition. The approximate controller fully covers feasible set of the optimal controller.

8. Conclusions

In this paper we have presented a survey of recent developments in parametric linear programming. Current methods have been classified in three categories, including two that are well established in the computational geometry community, with the third arising from recent work using geometric intuition. A new method has been presented that is based on the third standard approach for convex hull computation, the beneath/beyond method. It has been demonstrated that such an approach can be used to efficiently compute approximate and stabilizing control laws with a simple structure.

References

1. Avis D, Fukuda K. Reverse search for enumeration. Discrete Applied Math 1996; 65: 21–46
2. Balinski ML. On the graph structure of convex polyhedra in *n*-space. Pacific J Math 1961; 95(11): 431–434
3. Baotić M. An efficient algorithm for multi-parametric quadratic programming. Tech. rep., ETH Zürich, Institut für Automatik, Physikstrasse 3, CH-8092, Switzerland, 2002
4. Barber CB, Dobkin DP, Huhdanpaa H. The quickhull algorithm for convex hulls. ACM Trans Math Softw 1996; 22(4): 469–483
5. Bemporad A, Borrelli F, Morari M. Model predictive control based on linear programming – the explicit solution. IEEE Trans Autom Control 2002; 47(12): 1974–1985
6. Bemporad A, Filippi C. Approximate convex multiparametric programming. In: Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, Hawaii, December 2003, pp. 3185–3190
7. Bemporad A, Filippi C. Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming. J Optim Theory Appl 2003; 117(1): 9–38
8. Bemporad A, Filippi C. Robust explicit MPC based on approximate multi-parametric convex programming. In Decision and Control, 2004. CDC. 43rd IEEE Conference on (2004), vol.3, pp. 2491–2496
9. Bemporad A, Filippi C. An algorithm for approximate multiparametric convex programming. Comput Optim Appl 2006; V35(1): 87–108
10. Bemporad A, Morari M, Dua V, Pistikopoulos EN. The explicit linear quadratic regulator for constrained systems. Automatica 2002; 38(1): 3–20
11. Bertsekas D, Tsitsiklis JN. Introduction to Linear Optimization. Athena Scientific, 1997
12. Blanchini F. Nonquadratic Lyapunov functions for robust control. Automatica 1995; 31(3): 451–461
13. Blanchini F. Set invariance in control – a survey. Automatica 1999; 35(11): 1747–1768
14. Borrelli F, Bemporad A, Morari M. A geometric algorithm for multi-parametric linear programming. J Optim Theory Appl 2003; 118(3): 515–540
15. Bremner D. Incremental convex hull algorithms are not output sensitive. Discrete Comput Geometry 1999; 21(1): 57–68
16. Bremner D, Fukuda K, Marzetta A. Primal-dual methods for vertex and facet enumeration. Discrete Comput Geometry 1998; 20: 333–357
17. Clarkson K. More output-sensitive geometric algorithms. In: 35th Annual IEEE Symposium on the Foundations of Computer Science (1994), pp. 695–702
18. Dantzig GB, Orden A, Wolfe P. The generalized simplex method for minimizing a linear form under linear inequality restraints. Pacific J Math 1995; 5: 183–195
19. De la Pena D, Bemporad A, Filippi C. Robust explicit MPC based on approximate multiparametric convex programming. Automatic Control, IEEE Trans 2006; 51(8): 1399–1403
20. Edelsbrunner H, Mücke EP. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. ACM Trans Graph 1990; 9(1): 66–104
21. Fukuda K. Frequently asked questions in polyhedral computation. http://www.ifor.math.ethz.ch/fukuda/polyfaq/polyfaq.html, October 2000
22. Fukuda K, Lüthi HJ, Namkiki M. The existence of a short sequence of admissible pivots to an optimal basis in LP and LCP. Int Trans Oper Res 1997; 4(4): 273–384
23. Fukuda K, Prodon A. Double description method revisited. In: Combinatorics and Computer Science Deza M, Euler R, Manoussakis I (Eds.), vol. 1120 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996, pp. 91–111
24. Gal TA. 'histogramme' of parametric programming. J Opl Res Soc 1980; 31: 449–451
25. Gal T. Postoptimal Analyses, Parametric Programming and Related Topics, Second ed. Walter de Gruyter, 1995
26. Gal T, Nedoma J. Multiparametric linear programming. Manage Sci 1972: 18(7): 406–422
27. Goodman JE, O'Rourke J. (Eds.) Handbook of Discrete and Computational Geometry, second ed. CRC Press, New York, 1997
28. Grieder P. Efficient Computation of Feedback Controllers for Constrainted Systems. PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, 2004
29. Grieder P, Borrelli F, Torrisi F, Morari M. Computation of the constrained infinite time linear quadratic regulator. Automatica 2004; 40: 701–708
30. Grünbaum B. Measures of symmetry for convex sets. In: Proceedings of the Seventh Symposium in Pure Mathematics of the American Mathematical Society, Symposium on Convexity (1961), pp. 233–270
31. Grünbaum B. Convex Polytopes, second ed. Springer-Verlag, 2000
32. Johansen TA. Approximate explicit receding horizon control of constrained nonlinear systems. Automatica 2004; 40(2): 293–300
33. Johansen TA, Grancharova A. Approximate explicit constrained linear model predictive control via orthogonal search tree. IEEE Trans. Autom Control 2003; 48: 810–815
34. Johansen TA, Petersen I, Slupphaug O. On explicit suboptimal LQR with state and input constraints. In: Decision and Control, 2000. Proceedings of the 39th IEEE Conference on (2000), vol.1, pp. 662–667
35. Jones C, Maciejowski J. Lecture Notes in Computer Science : Mathematical Software – ICMS 2006. Springer-Verlag, 2006, ch. Primal-Dual Enumeration for Multiparametric Linear Programming, pp. 248–259
36. Jones C, Morari M. Multiparametric Linear Complementarity Problems. In: IEEE Conference on Decision and Control (Dec. 2006)
37. Jones CN. Polyhedral Tools for Control. PhD thesis, University of Cambridge, July 2005
38. Jones CN, Kerrigan E, Maciejowski J. Lexicographic perturbation for multiparametric linear programming with applications to control. Automatica (2006). To appear
39. Jones CN, Maciejowski JM. Reverse search for parametric linear programming. In: 45th Conference on Decision and Control (December 2006)
40. Kallay, M. Convex hull algorithms for higher dimensions. Unpublished Manuscript, 1981
41. Kvasnica M, Grieder P, Baotić M. Multi-Parametric Toolbox (MPT), 2004.
42. Kvasnica M, Grieder P, Baotić M, Morari M. Multi Parametric Toolbox (MPT). In: Hybrid Systems:

Computation and Control (Philadelphia, Pennsylvania, USA, March 2004), vol. 2993 of Lecture Notes in Computer Science, Springer Verlag, pp. 448–462

43. Loechner V, Wilde DK. Parameterized polyhedra and their vertices. Int J Parallel Programming 1997; V25(6): 525–549
44. Mayne D, Rawling J, Rao C, Scokaert P. Constrained model predictive control: Stability and optimality. Automatica 2000; 36(6): 789–814
45. Motzkin TS, Raiffa H, Thompson GL, Thrall RM. The double description method. In: Contributions to the Theory of Games II, Kuhn HW, Tucker AW (Eds.), vol. 8 of Ann Math Stud Princeton University Press, 1953, pp. 51–73
46. Murty KG. Linear Programming, first ed. John Wiley & Sons, Inc., 1983
47. Nocedal J, Wright S. Numerical Optimization. Springer, New York, USA, 1999
48. Olaru S, Dumur D. A parameterized polyhedra approach for explicit constrained predictive control. In: Decision and Control, 2004. CDC. 43rd IEEE Conference on (2004), vol. 2, pp. 1580–1585
49. Olaru S, Dumur D. On the continuity and complexity of control laws based on multiparametric linear programs. In: 45th Conference on Decision and Control (Dec 2006)
50. Orchard-Hays W. Notes on linear programming (part 6): the rand code for the simplex method (sx4). Tech. Rep. 1440, Rand Corporation, 1955
51. Ottmann T, Schuierer S, Soundaralakshmi S. Enumerating extreme points in higher dimensions. In: STACS 95: 12th Annual Symposium on Theoretical Aspects of Computer Science (1995), EW. Mayer and C. Puech, Eds., Lecture Notes in Computer Science 900, Springer-Verlag, pp. 562–570
52. Pfaff B. An Introduction to Binary Search Trees and Balanced Trees. Free Software Foundation, Inc, http://benpfaff.org/, 2002
53. Raković S. Private communication, 2003
54. Saaty T, Gass S. The parametric objective function 1. Oper Res 1954; 2: 316–319
55. Schechter, M. Polyhedral functions and multiparametric linear programming. Journal of Optimization Theory and Applications 1987; 53(2): 269–280
56. Scokaert POM, Mayne DQ, Rawlings JB. Suboptimal model predictive control (feasibility implies stability). Automatic Control, IEEE Trans 1999; 44(3): 648–654
57. Seron M, Doná, JD, Goodwin, G. Global analytical model predictive control with input constraints. In: Proceedings of the 39th IEEE Conference on Decision and Control (Sydney, Australia, 2000)
58. Seron M, Goodwin G, Doná, JD. Characterisation of receding horizon control for constrained linear systems. Asian J Control 2003; 2(5): 271–286
59. Seron MM, Goodwin GC, De Doná, JA. Geometry of model predictive control for constrained linear systems. Tech. Rep. EE0031, The University of Newcastle, Australia, 2000
60. Spjøtvold J, Tøndel P, Johansen TA. A method for obtaining continuous solutions to multiparametric linear programs. In Proceedings of the 16th IFAC World Congress (Prague, 2005)
61. Suard R, Löfberg J, Grieder P, Kvasnica M, Morari M. Efficient computation of controller partitions in multiparametric programming. In: Proceedings of the 43rd IEEE Conference on Decision and Control (Bahamas, 2004)
62. Tondel P, Johansen TA, Bemporad A. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. Automatica 2003; 39(3): 489–497
63. Tøndel P, Johansen, TA, Bemporad A. Further results on multi-parametric quadratic programming. In: 42nd IEEE Conference on Decision and Control (Hawaii, 2003), pp. 3173–3178
64. Ziegler GM. Lectures on Polytopes. Springer-Verlag, New York, 1995