Primal-Dual Enumeration for Multiparametric Linear Programming

Colin N. Jones[1] and Jan M. Maciejowski[2]

[1] Automatic Control Laboratory, Swiss Federal Institute of Technology
Physikstrasse 3, CH-8092 Zurich, Switzerland
cjones@ee.ethz.ch
[2] Control Group, Department of Engineering, University of Cambridge
Trumpington St, Cambridge, UK
jmm@eng.cam.ac.uk

Abstract. Optimal control problems for constrained linear systems with a linear cost can be posed as multiparametric linear programs (pLPs) and solved explicitly offline. Several algorithms have recently been proposed in the literature that solve these pLPs in a fairly efficient manner, all of which have as a base operation the computation and removal of redundant constraints. For many problems, it is this redundancy elimination that requires the vast majority of the computation time. This paper introduces a new solution technique for multiparametric linear programs based on the primal–dual paradigm. The proposed approach reposes the problem as the vertex enumeration of a linearly transformed polytope and then simultaneously computes both its vertex and halfspace representations. Exploitation of the halfspace representation allows, for smaller problems, a very significant reduction in the number of redundancy elimination operations required, resulting in many cases in a much faster algorithm.

1 Introduction

It is standard practice to implement a model predictive controller (MPC) by solving an optimisation problem on–line. For example, when the system is linear, the constraints are polyhedral and the cost is linear (e.g. $1-$ or $\infty-$norm), this amounts to computing a single linear program (LP) 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. 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 [1].

The computation of the optimal PWA function, mapping the measured state to the control input, can be posed as the following (multi)parametric linear program (pLP) [1]:

$$\min_u \left\{ c^T u \mid (x,u) \in P \right\}, \tag{1}$$

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

Several methods of computing the solution to pLP (1) can be found in the literature (e.g., [1–3]). All of these approaches enumerate the affine regions of the optimal PWA function one at a time. For each of the affine pieces of the function, the main computational burden is the determination of a minimal description of the polytope in which it is optimal. Computing this minimal representation is a so–called *redundancy elimination* operation, which requires the solution of a number of linear programs equal to the size of the input (the number of inequalities describing the polytope P). In most cases, these redundancy elimination LPs take the vast majority total computation time.

In this paper we present a new method of computing the optimiser of pLP (1) based on a *primal-dual* approach [4]. We first show that parametric linear programming can be posed as a vertex enumeration problem of an affine transform of the dual constraints in (1). The primal-dual algorithm then computes the convex hull of this transformed polytope as it is enumerating the vertices. The availability of these two descriptions of the same polytope then allow a significant reduction in the amount of work that the algorithm is required to do by removing the need to compute a large number of the redundancy elimination LPs.

The remainder of this paper is organised as follows. Section 2 provides required background on parametric linear programming. Section 3 introduces a polytope such that there is a one-to-one mapping from its vertices to the affine pieces of the solution. The primal-dual approach described in [4] is then adapted such that it can be applied to this polytope, and hence to the associated pLP in Section 3.2. Finally, examples and conclusions are given in Sections 4 and 5 respectively.

Notation

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 *polyhedron* is the intersection of a finite number of halfspaces and a *polytope* is a bounded polyhedron. If $P = \{x \mid Ax \leq b\}$ is a polyhedron and $H = \left\{x \,\middle|\, a^T x \leq d\right\}$ is a halfspace such that $P \subseteq H$, then $P \cap \left\{x \,\middle|\, a^T x = d\right\}$ is a *face* of P. One– and zero–dimensional faces are called *edges* and *vertices* respectively. Faces of dimension $\dim(P) - 1$ are called *facets* and $\dim(P) - 2$, *ridges*.

A vector $r \in \mathbb{R}^d$ defines a *ray* as $R = \{r\alpha \mid \alpha > 0\}$. A set C is called a cone if for every $x \in C$ and scalar $\alpha > 0$, 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 $C = \text{cone}(F) \triangleq \{F\alpha \mid \alpha \geq 0\}$. The generator $F_{*,i}$ 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 $A \oplus B \triangleq \{x + y \mid x \in A,\ y \in B\}$.

2 Preliminaries

2.1 Linear Programming

Consider the following linear program:

$$\max_{\lambda} \left\{ c^T \lambda \mid \lambda \in D \right\} \ , \tag{2}$$

where $\lambda \in \mathbb{R}^n$ is the optimiser, $c \in \mathbb{R}^n$ is a vector and the constraint polytope D is defined by the matrix $A \in \mathbb{R}^{m \times n}$ and the vector $b \in \mathbb{R}^m$ as

$$D \triangleq \{\lambda \mid A\lambda = b,\ \lambda \geq 0\} \ . \tag{3}$$

Any set $B \subset \{1, \ldots, n\}$ such that $|B| = m$ and $\operatorname{rank} A_{*,B} = m$ is called a *basis* and we write $N = \{1, \ldots, n\} \setminus B$ for its complement and call λ_B and λ_N the *basic* and *non-basic variables* respectively. Every basis B defines a *basic solution*[1] λ^B to the linear equations in (3), which is given by restricting the non-basic constraints to zero $\lambda^B_B = A^{-1}_{*,B} b,\ \lambda^B_N = 0$. A basis is called *primal feasible* if the resulting solution also satisfies the inequality constraints in (3): $A^{-1}_{*,B} b \geq 0$. Note that all solutions represented by bases occur at a vertex of the constraint polyhedron D.

2.2 Optimality Conditions

Definition 1 (Tangent cone [5]). *Let λ be an element of the polyhedron $D \subseteq \mathbb{R}^n$. A vector $\gamma \in \mathbb{R}^n$ is said to be a* feasible direction *at λ if there exists a strictly positive scalar α for which $\lambda + \alpha\gamma \in D$. The set of all feasible directions at λ is called the* tangent cone *and is written $\mathcal{T}_D(\lambda)$. The* support cone *$\mathcal{S}_D(\lambda)$ is the translation of $\mathcal{T}_D(\lambda)$ by the vector λ, $\mathcal{S}_D(\lambda) \triangleq \mathcal{T}_D(\lambda) \oplus \{\lambda\}$. Note that strictly speaking, the support cone is not a cone as it has a vertex at λ rather than zero.*

Given a basis B, the extreme feasible directions at the solution λ^B are given by increasing each non-basic variable in a feasible (positive) direction:

$$\lambda_B = \lambda^B - A^{-1}_{*,B} A_{*,i} \lambda_i, \quad \lambda_i \geq 0, \quad \lambda_{N \setminus \{i\}} = 0, \quad \forall i \in N \ .$$

The set of all convex combinations of the extreme rays give the tangent cone:

$$\mathcal{T}_D(\lambda^B) = \operatorname{cone}(F) \ , \tag{4}$$

where $F_{B,*} \triangleq -A^{-1}_{*,B} A_{*,N}$, $F_{N,*} \triangleq I$.

[1] Where clear from the context, we will refer to the basic solution as simply the solution.

Theorem 1 (Optimality Condition). *Let λ be an element of the polyhedron D. A necessary and sufficient condition for λ to be a global minimum of the linear program* (2) *is $c^T\gamma \geq 0$ for all feasible directions γ at λ.*

Definition 2. *The* normal cone *to D at λ is the orthogonal complement of the tangent cone:*

$$\mathcal{N}_D(\lambda) \triangleq \left\{ v \;\middle|\; v^T\gamma \leq 0,\ \forall \gamma \in \mathcal{T}_D(\lambda) \right\} \ .$$

From the above definition and (4), the normal cone to the basic feasible solution λ^B is:

$$\mathcal{N}_D(\lambda^B) = \left\{ v \;\middle|\; F^T v \leq 0 \right\} \ , \tag{5}$$

where F is as defined in (4). A direct result of Theorem 1 and (5) is that the basic solution λ^B, and hence the basis B, is optimal if and only if[2]

$$-c \in \mathcal{N}_D(\lambda^B) \ . \tag{6}$$

2.3 Parametric Linear Programming

The problem we will consider in this paper is the following *parametric linear program*:

$$\max_{\lambda} \left\{ x^T E\lambda \mid \lambda \in D \right\} \ , \qquad\qquad x \in \mathcal{X} \tag{7}$$

where $x \in \mathcal{X}$ is the parameter, $\mathcal{X} \subseteq \mathbb{R}^d$ and $E^T \in \mathbb{R}^{d\times n}$ is a matrix of rank d, $d < n$. It is assumed throughout this paper that the set of feasible parameters $\mathcal{X}$ is full–dimensional, which is common to most pLP algorithms [1–3]. This assumption can be easily guaranteed through a pre-processing operation [1]. The standard assumption is also made that pLP (7) has an optimal bounded solution for every $x \in \mathcal{X}$.[3]

Definition 3 (Critical Region). *If B is a basis of pLP* (7)*, then the* critical region *$\mathscr{R}_B$ is defined as the set of all parameters $x_\circ \in \mathcal{X}$ such that B is optimal for $x = x_\circ$.*

From (6) and (7) that the critical region $\mathscr{R}_B$ is the polyhedral set $\mathscr{R}_B = \left\{ x \;\middle|\; F^T E^T x \geq 0 \right\} \cap \mathcal{X}$.

Our goal is to enumerate all full–dimensional critical regions. In [6] it was shown that by *lexicographically* perturbing the problem (7), the following properties hold:

1. Every full–dimensional critical region is uniquely defined by a single basis
2. The interiors of the full–dimensional critical regions do not overlap

[2] The vector $-F^T c$ is often referred to as the reduced cost $\bar{c}$ and condition (6) then becomes $\bar{c} \geq 0$.

[3] Note that this also implies that the dual solution is feasible and bounded.

3. The union of all full–dimensional critical regions is the set of feasible parameters $\mathcal{X}$.

In the remainder of this paper we will assume that the problem has been lexicographically perturbed and will therefore not discuss possible degeneracy of the solution.[4]

Remark 1. The standard parametric linear program resulting from model predictive control problems has a cost of the form $(x^T E + c)\lambda$ [1], which differs from that used here by the constant c. The procedure developed in this paper can be applied to such problems with no added complexity by first *homogenizing* the cost as detailed in [6].

3 pLP as 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 constraint polytope D.

Theorem 2. *If B is a feasible basis of pLP (7) and λ^B is the basic solution, then B defines a full–dimensional critical region if and only if $E\lambda^B$ is a vertex of the polytope $ED \triangleq \{E\lambda \mid A\lambda = b,\ \lambda \geq 0\}$.*

Proof. pLP (7) can be re-written as $\max_z \left\{x^T z \mid z \in ED\right\}$ through the change of variable $z \triangleq E\lambda$. It follows from (6) and Definition 3 that x is in the critical region $\mathscr{R}_B$ if and only if $-x$ is in the normal cone $\mathcal{N}_{ED}(E\lambda^B)$. Finally, the normal cone of a point $E\lambda^B$ in a polytope ED is full–dimensional if and only if $E\lambda^B$ is a vertex of ED [7, Sec. 3.2].

Two vertices of a polytope are called *neighbours*, or *adjacent*, if they are contained in the same edge, or one–dimensional face of the polytope. The proposed algorithm begins at a single vertex of ED and then recursively computes neighbours until all vertices have been found. In the following section we see that the adjacent vertices of a vertex $v \in ED$ are given by the intersection of ED and the extreme rays of the support cone $\mathcal{S}_{ED}(v)$. The proposed method is outlined as Algorithm 1 below.

Remark 2. Note that the extreme rays of the tangent cone (and hence the support cone) are given directly by the normals of the irredundant inequalities describing the normal cone. The negative normal cone of ED at a vertex $E\lambda^B$ is exactly the critical region of the basis B, $\mathscr{R}_B = -\mathcal{N}_{ED}(E\lambda^B)$ and therefore determining the extreme rays of the support cone is an operation that is entirely equivalent to the redundancy elimination operations that are done to compute the facets of the critical regions in other methods, e.g. [1–3].

[4] Note that our definition of a critical region differs from that generally found in the literature. However, under the assumption that the problem is non–degenerate, or equivalently, lexicographically perturbed, the two definitions are equivalent.

Algorithm 1 below is similar to others presented in the literature [1–3], although the formulation is in the dual. The main contribution of this paper is in Step 4, where one must determine the extreme rays of the support cone of ED at the point $E\lambda^B$. In current algorithms, this requires a redundancy elimination operation in order to determine which rays are extreme. For problems that are of interest to control and are yet small enough to be computed, this redundancy elimination requires the majority of the computation time, as is illustrated in Section 4. Section 3.2 presents a new primal–dual approach that can significantly reduce the computation time for these smaller problems.

Remark 3. Algorithm 1 can be seen as a form of *gift-wrapping* algorithm in a polar dual context (also called *pivoting algorithms*) in which the vertices are not known *apriori* but are provided by an oracle.

Algorithm 1 Parametric Linear Programming

Require: Basis B_0 of pLP (7) such that $\dim \mathscr{R}_B = d$
Ensure: All bases B such that $\dim \mathscr{R}_B = d$
1: $\mathcal{L}_{unexplored} \longleftarrow \{B\}\,, \quad \mathcal{L}_{discovered} \longleftarrow \{B\}$
2: **while** $\mathcal{L}_{unexplored}$ is not empty **do**
3: Select any basis B from $\mathcal{L}_{unexplored}$
4: **for all** extreme rays r of $E\mathcal{S}_D(\lambda^B)$ **do** Section 3.2
5: $B' \longleftarrow$ `neighbour`(r, B) Section 3.1
6: $\mathcal{L}_{unexplored} \longleftarrow \mathcal{L}_{unexplored} \cup (\{B'\} \setminus \mathcal{L}_{discovered})$
7: $\mathcal{L}_{discovered} \longleftarrow \mathcal{L}_{discovered} \cup \{B'\}$
8: **end for**
9: **end while**
10: Return list $\mathcal{L}_{discovered}$

3.1 Neighbour Function

This section outlines the function `neighbour`$(\cdot,\cdot)$, which is used in Step 5 of Algorithm 1. The edges of a polytope P that intersect at a vertex $v \in P$ are given by the intersection of P with the extreme rays of the support cone at v [5]. The following lemma describes the tangent cone of a basic solution of ED, where we recall that the support cone is equal to the tangent cone shifted by v.

Lemma 1. *If λ is an element in the polytope D, then $\mathcal{T}_{ED}(E\lambda) = E\mathcal{T}_D(\lambda)$.*

Proof. From Definition 1, $\gamma \in \mathbb{R}^d$ is in $E\mathcal{T}_D(\lambda)$ if and only if there exists a scalar $\alpha > 0$ and a vector g such that

$$\gamma = Eg, \quad \lambda + \alpha g \in D \ . \tag{8}$$

By assumption, E is rank d and therefore such a g always exists for each $\gamma \in E\mathcal{T}_D(\lambda)$. Under the mapping E, (8) becomes $E\lambda + \alpha\gamma \in ED$, which is true if and only if $\gamma \in \mathcal{T}_{ED}(E\lambda)$.

Lemma 1 and (4) give a description of the tangent cone to a vertex $E\lambda^B$ of ED defined by the basis B:

$$\mathcal{T}_{ED}(E\lambda^B) = \text{cone}(EF) \ , \tag{9}$$

where $F_{B,*} \triangleq -A_{*,B}^{-1}A_{*,N}$, $F_{N,*} \triangleq I$. Note however, that not every column $EF_{*,i}$ defines an extreme ray of $\mathcal{T}_{ED}(E\lambda^B)$, as some of them may well be redundant. Determining if a ray is redundant or not requires the majority of effort during the computation of a pLP and is the main topic of this paper. A new approach to determining redundancy will be introduced in Section 3.2.

We can now define the function $B' = \texttt{neighbour}(r, B)$, where $r = \left\{E(F_{*,i}\alpha + \lambda^B) \mid \alpha \geq 0\right\}$ is an extreme ray of the support cone $\mathcal{S}_{ED}(E\lambda^B)$. The neighbour function returns the basis B' such that $E\lambda^{B'}$ is the vertex of $r \cap ED$, which is different from $E\lambda^B$. Note that there are exactly two vertices on each edge.

The following new theorem provides an efficient method of computing the basis that represents the adjacent vertex given an irredundant ray of the tangent/support cone.

Theorem 3. *If B is a feasible basis of D, $E\lambda^B$ is a vertex of ED and $r = \left\{E(F_{*,i}\alpha + \lambda^B) \mid \alpha \geq 0\right\}$ is an extreme ray of the support cone $\mathcal{S}_{ED}(E\lambda^B)$, then the adjacent vertex of $E\lambda^B$ in the direction r is the optimal basis of the LP:*

$$\max_{\lambda} \left\{(EF_{*,i})^T E\lambda \mid \lambda \in D,\ \lambda_j = 0,\ \forall j \notin \mathcal{Q}\right\} \ , \tag{10}$$

where $\mathcal{Q} \triangleq \{j \mid \exists \rho \geq 0,\ EF_{*,i} = \rho EF_{*,j}\}$ and $F_{B,*} \triangleq -A_{*,B}^{-1}A_{*,N}$, $F_{N,*} \triangleq I$.

Proof. The adjacent vertex is reached by moving along the edge $r \cap ED$ away from $E\lambda^B$. Every point $\lambda \in D$ can be written as $\lambda = \lambda^B + F\gamma$, for some $\gamma \geq 0$, where F is as defined in the statement of the theorem, because every tangent cone is a superset of the polytope [7]. Consider the column $F_{*,j}$ and the resulting ray $\lambda = \lambda^B + F_{*,j}\gamma_j$, $\gamma_j \geq 0$. Clearly, $E\lambda \in r$ if and only if there exists a $\rho \geq 0$ such that $EF_{*,j} = \rho EF_{*,i}$. Therefore, the face P of D such that $EP = r \cap ED$ is given by $P = \{\lambda \mid \lambda_i = 0,\ \forall i \notin \mathcal{Q}\} \cap D$.

The LP given in the statement of the theorem then maximises in the direction of the ray r, while restricting λ to be in the face P.

Remark 4. Note that if the set $\mathcal{Q}$ in Theorem 3 contains only one element more than the basis B, then LP (10) will compute the adjacent basis in a single simplex pivot. This is a significant improvement over current methods [1–3], which always require the calculation of an LP of dimension equal to that of D.

3.2 Primal-Dual Enumeration

The standard method for redundancy elimination, or determining which rays are extreme in Step 3 of Algorithm 1 requires a single linear program of dimension d

per ray [8]. Testing if the i^{th} ray of the support cone $\mathcal{S}_{ED}(E\lambda^B)$ for some vertex $E\lambda^B$ of ED is redundant can be done using the following linear program [8]:

$$\begin{aligned} J(i) = \text{minimise} \quad & (-EF_{*,i})^T x \\ \text{subject to} \quad & \left(EF_{*,\{1:i-1,i+1,n\}}\right)^T x \geq 0 \\ & (EF_{*,i})^T x \geq -1 \end{aligned} \tag{11}$$

where the ray r^i is extreme if $J(i) < 0$. Current pLP methods reported in the literature [1–3] require the solution of LP (11) for each column of F at every vertex, resulting in the computation of a very large number of linear programs. This paper seeks to reduce this requirement through a heuristic based on [4], which can significantly reduce the work required to compute the extreme rays.

The approach presented in this section is called 'primal-dual' because both the primal (vertex) and dual (halfspace) representations of ED are computed. At the q^{th} step of the algorithm, q vertices of ED will have been found. At this point, the algorithm has a list of these q vertices $\left\{v^1, \ldots, v^q\right\}$; it also stores a halfspace representation of their convex hull $\mathcal{H}^q \triangleq \left\{z \,|\, G^q z \leq g^q\right\} = \text{conv}\left\{v^1, \ldots, v^q\right\}$. When a new vertex v^{q+1} is found, the existing description of the convex hull is first extended to include it: a new matrix G^{q+1} and vector g^{q+1} are computed such that $\mathcal{H}^{q+1} = \left\{z \,\middle|\, G^{q+1} z \leq g^{q+1}\right\} = \mathcal{H}^q \cup \left\{v^{q+1}\right\}$.

We are now able to use this information to improve the efficiency of redundancy elimination. Given a basis B, we begin by writing down the known description $\mathcal{S}_{ED}(E\lambda^B) = \left\{E(\lambda^B + F\gamma) \,|\, \gamma \geq 0\right\}$ of the support cone at the vertex $E\lambda^B$ from (9). The goal is now to test each ray $r^i \triangleq \left\{E\left(\lambda^B + F_{*,i}\gamma_i\right) \,|\, \gamma_i \geq 0\right\}$ to determine if it is an extreme ray of $\mathcal{S}_{ED}(E\lambda^B)$.

We note that the polytope $\mathcal{H}^{q+1}$ is an inner approximation of the set ED. It follows that if the ray r^i intersects the interior of $\mathcal{H}^{q+1}$, then it also intersects the interior of ED and is therefore not an extreme ray of $\mathcal{S}_{ED}(ED)$. This notion is formalised in the following theorem.

Theorem 4. *Let $\mathcal{H}^q = \text{conv}\left\{v^1, \ldots, v^q\right\} = \left\{z \,|\, G^q z \leq g^q\right\}$, where v^i are q vertices of ED and $\dim(\mathcal{H}^q) = \dim(ED)$. If $E\lambda^B$ is a vertex of ED and of $\mathcal{H}^q$, then $r^i \triangleq \left\{E\left(\lambda^B + F_{*,i}\gamma_i\right) \,|\, \gamma_i \geq 0\right\}$ is a redundant ray of the support cone $\mathcal{S}_{ED}(E\lambda^B)$ if for each j such that $G_{j,*}v = g_j$ the condition $G_{j,*}EF_{*,i} < 0$ holds, where F is defined as in (4).*

Proof. The test is simply to check if a point on the ray r^i is internal to $\mathcal{H}^q$ for a strictly positive γ_i:

$$\begin{aligned} GE\left(\lambda^B + F_{*,i}\gamma_i\right) &\leq g \\ GEF_{*,i}\gamma_i &\leq g - GE\lambda^B \quad . \end{aligned} \tag{12}$$

Recall that $E\lambda^B$ is a vertex of $\mathcal{H}^q$ and therefore $g - GE\lambda^B \geq 0$. For those constraints that are strictly greater than zero, (12) will clearly be satisfied for some $\gamma_i > 0$ and therefore we have only to test those constraints that are equal to zero. Clearly, there exists a strictly positive α such that (12) is satisfied if and only if $G_{j,*}EF_{*,i} < 0$ for all j such that $G_{j,*}E\lambda^B = g_j$.

Theorem 4 can now be used during Step 3 of Algorithm 1 to determine which, if any, of the rays of $\mathcal{S}_{ED}(E\lambda^B)$ are redundant. As this test can only prove redundancy and not irredundancy, if the conditions of the theorem are not met, then the linear program (11) must still be solved.

Convex Hull Computation The use of Theorem 4 requires the computation of the convex hull $\mathcal{H}^q$ of the first q discovered vertices $\left\{v^1, \ldots, v^q\right\}$ of ED. While any convex hull algorithm could be used, ideally the algorithm should be incremental, or able to add one vertex at a time, and have the ability to quickly identify which inequalities are active at the most recently added vertex.

An incremental approach takes as input a full–dimensional polytope $\mathcal{H}^{q-1} = \left\{z \;\middle|\; G^{q-1}z \leq g^{q-1}\right\}$ and computes the convex hull $\mathcal{H}^q = \mathcal{H}^{q-1} \cup \{v^q\}$ for a point v^q. The facet $\mathcal{H}^{q-1} \cap \left\{z \;\middle|\; G^{q-1}_{i,*}z = g^{q-1}_i\right\}$ is called visible from v^q if its supporting hyperplane separates $\mathcal{H}^{q-1}$ and v^q (i.e. $G^{q-1}_{i,*}v^q > g^{q-1}_i$), otherwise the facet is obscured. The set of facets of $\mathcal{H}^q$ then consists of the obscured facets of $\mathcal{H}^{q-1}$ as well as a new set of facets to replace the visible ones, which include the point v^q. Updating $\mathcal{H}^{q-1}$ to $\mathcal{H}^q$ therefore consists of two subproblems: finding all visible facets of $\mathcal{H}^{q-1}$ and computing new facets to replace them. The inequalities that must be tested in Theorem 4 is then exactly the set of new facets that are computed to replace the visible ones and are therefore computed as a side effect of the convex hull algorithm.

There are two approaches available for determining the set of visible facets that can be applied in the context of this paper, where the list of points v^i for i larger than q is not available while computing $\mathcal{H}^q$. The first is to simply check each facet of $\mathcal{H}^{q-1}$ to determine if v^q is on its positive (obscured) or negative (visible) side. This is the approach used in Kallay's beneath–beyond [9] and Motzkin's double description [10] methods and requires time linear in the number of facets of $\mathcal{H}^{q-1}$. An improvement on this is to store a so–called facet graph, whose nodes are facets and arcs connect facets if they share a common ridge. The set of visible facets then forms a subgraph of the facet graph and can be efficiently enumerated in time proportional to the number of visible facets [11].

Once the visible facets are computed and removed, the supporting hyperplanes of the new facets containing the point v^q can be efficiently computed by noting that they must contain the point v^q as well as the ridges formed by the intersection of the removed facets and the obscured facets.

The reader is referred to [12] for a more complete handling of incremental convex hull algorithms.

Complexity Current methods of computing pLPs are in a sense output sensitive, in that they require a fixed number of redundancy elimination LPs (11) to be computed per critical region of the solution (one for each column of the matrix F). The approach introduced in this paper aims to reduce the number of redundancy elimination LPs through the use of Theorem 4. However, since Theorem 4 is a sufficient condition for redundancy and not a necessary one, no

guarantee can be made that any of the rays will satisfy the conditions of the theorem, and as a result it may be the case that LP (11) must still be computed for each ray, which would therefore result in no improvement over current methods in the worst–case.

The additional cost of using the primal-dual test is the calculation and storage of the halfspace description of ED. While this convex hull can be computed efficiently in an incremental fashion as outlined above, the relationship between the number of vertices in ED and the number of inequalities can be exponential. As a result the applicability of this algorithm is limited to those polyhedra ED that can be described with both relatively few inequalities and few vertices. In Section 4 it will be seen that there are problems that are of a size and structure that are interesting in a control context and which satisfy these requirements.

4 Examples

The primary motivation for pLPs in control is the calculation of so–called closed–form or explicit Model Predictive Control (MPC) laws. In standard MPC an optimisation problem, which is a function of the current state, is solved at each sampling instant, whereas in closed-form MPC the problem is posed in multi-parametric form, with the state as a parameter, and solved offline.

The goal is to regulate the linear time invariant (LTI) system $x^+ = Ax + Bu$ to the origin, where $x \in \mathbb{R}^n$ is the state, x^+ is the successor state and $u \in \mathbb{R}^m$ is the input. A standard Model Predictive Controller (MPC) can be written as the solution to the following optimisation problem, in which the optimiser u_1 is the input that is applied to the system, given the measured state x:

$$\begin{array}{rll} J(x) = \underset{u_1,\ldots,u_{N-1},x_1,\ldots,x_N}{\text{minimise}} & \sum_{i=1}^{N-1} \|Ru_i\|_p + \sum_{i=1}^{N-1} \|Qx_i\|_p + \|Q_F x_N\|_p & \\ \text{subject to} & x_0 = x & \\ & x_{i+1} = Ax_i + Bu_i, & i = 0,\ldots,N-1 \\ & x_{i+1} \in \mathcal{X}, \quad u_i \in \mathcal{U} & i = 0,\ldots,N-1 \end{array} \tag{13}$$

where x_i and u_i are future predicted states and inputs respectively, which are constrained to be in the polytopes $\mathcal{X}$ and $\mathcal{U}$. If the norm p is taken to be either the $1-$ or $\infty-$norm, then a linear program results, which is the case of interest in this paper. Conversion of this problem to the form of pLP (1) is discussed in [1] and requires the introduction of several slack variables.

4.1 Closed-Form MPC for a 4D System

Consider the problem (13) with the following randomly generated system, which is given as an example in the MPT toolbox [13]:

$$x^+ = \begin{bmatrix} 0.7 & -0.1 & 0.0 & 0.0 \\ 0.2 & -0.5 & 0.1 & 0.0 \\ 0.0 & 0.1 & 0.1 & 0.0 \\ 0.5 & 0.0 & 0.5 & 0.5 \end{bmatrix} x + \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 1.0 \\ 0.1 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} u$$

with a prediction horizon $N = 5$ and the constraints $\|u_i\|_\infty \leq 1$, $\|x_i\|_\infty \leq 5$ on the input and state respectively. The cost is the minimisation of the $\infty-$norm of the states and inputs at each point in time and the matrices Q, Q_F and R are taken as the identity.

When the problem is written in the form of pLP (2), (3) the matrix A is in $\mathbb{R}^{20\times 120}$ and E is in $\mathbb{R}^{4\times 120}$ and the solution contains $12,128$ critical regions. While this problem may seem fairly small, there are many interesting and useful control problems of this size and there are only a very small number of applications reported in the literature in which the parameter size is larger.

The proposed approach was compared against the two main methods for computing pLPs reported in the literature. The method used in the Multiparametric Toolbox (MPT [13]) is based on a similar exploration strategy as the proposed method and solves an LP of the form (11) for every possible redundant ray. The computation of adjacent critical regions is done using a linear program of dimension equal to that of D, and is therefore less efficient than that given in Theorem 3. The second method is that implemented in the Hybrid Toolbox [14] and is based on an entirely different exploration strategy. The reader is referred to [1] for details of this method.

From Table 1 one can see that the primal–dual algorithm offers a significant reduction in the number of pivots required to compute the solution. The computation of the convex hull for this example required 71.1 seconds to compute using qhull [15]. To give an idea of speed, a 3GHz Pentium IV machine using the Stanford Systems Optimization Laboratory (SOL) toolbox [16] can execute the required pivots for the primal–dual approach in 36.6 seconds, which when added to the time to compute the convex hull totals 107.7 seconds compared to a total of 280.1 seconds for the MPT [13] approach.

Table 1. Comparison of pLP Methods for Example 4.1

Method	**Simplex Pivots**	
	$\mathbb{R}^4$	$\mathbb{R}^{20}$
Primal–Dual	$761,487$	$76,488$
MPT [13]	$6,409,503$	$670,940$
Hybrid Toolbox [14]	$> 2GB$ RAM	

5 Conclusions

This paper has introduced a new method of enumerating the solution to a parametric linear program based on a primal–dual paradigm. It was shown that the proposed algorithm can significantly reduce the number of linear programs that need to be solved in order to determine irredundant descriptions of the critical regions. The code used in the paper is available as part of the Multiparametric Toolbox [13].

Acknowledgments

The authors would like to thank Eric Kerrigan and Komei Fukuda for their valuable discussions while preparing this paper.

References

1. Borrelli, F., Bemporad, A., Morari, M.: A Geometric Algorithm for Multi-Parametric Linear Programming. Journal of Optimization Theory and Applications **118**(3) (2003) 515–540
2. Tøndel, P., Johansen, T., Bemporad, A.: An algorithm for multi-parametric quadratic programming and explicit MPC solutions. Automatica (2003) 489–497
3. Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.: The explicit linear quadratic regulator for constrained systems. Automatica **38**(1) (2002) 3–20
4. Bremner, D., Fukuda, K., Marzetta, A.: Primal-dual methods for vertex and facet enumeration. Discrete and Computational Geometry **20** (1998) 333–357
5. Bertsekas, D., Tsitsiklis, J.: Introduction to Linear Optimization. Athena Scientific (1997)
6. Jones, C.: Polyhedral Tools for Control. PhD thesis, University of Cambridge (2005)
7. Ziegler, G.: Lectures on Polytopes. Springer-Verlag, New York (1995)
8. Fukuda, K.: Frequently asked questions in polyhedral computation. `http://www.ifor.math.ethz.ch/fukuda/polyfaq/polyfaq.html` (2000)
9. Preparatat, F., Shamos, M.: Computational Geometry: An Introduction. Springer-Verlag, New York (1985)
10. Motzkin, T., Raiffa, H., Thompson, G., Thrall, R.: The double description method. In Kuhn, H., Tucker, A., eds.: Contributions to the Theory of Games II. Volume 8 of Ann. of Math. Stud. Princeton University Press (1953) 51–73
11. Seidel, R.: A convex hull algorithm optimal for point sets in even dimension. Master's thesis, Dept. of Computer Science, University of British Columbia, Vancouver, Canada (1981)
12. Goodman, J.E., O'Rourke, J., eds.: Handbook of Discrete and Computational Geometry. CRC Press, New York (1997)
13. Kvasnica, M., Grieder, P., Baotić, M.: Multi-Parametric Toolbox (MPT) (2004) `http://control.ee.ethz.ch/~mpt/`.
14. Bemporad, A.: Hybrid toolbox (2005) Version 1.0.10, `http://www.dii.unisi.it/hybrid/toolbox/`.
15. Barber, C., Dobkin, D., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**(4) (1996) 469–483
16. Murray, W., Saunders, M.: Systems Optimization Laboratory (SOL) (2006) http://www.sbsi-sol-optimize.com.