

# Polyhedral Tools for Control

Colin N. Jones  
Pembroke College



Control Group  
Department of Engineering  
University of Cambridge

A thesis submitted for the degree of  
Doctor of Philosophy

4<sup>th</sup> July 2005



*To Karen*



# Abstract

Polyhedral operations play a central role in constrained control. One of the most fundamental operations is that of projection, required both by addition and multiplication. This thesis investigates projection and its relation to multi-parametric linear optimisation for the types of problems that are of particular interest to the control community.

The first part of the thesis introduces an algorithm for the projection of polytopes in halfspace form, called Equality Set Projection (ESP). ESP has the desirable property of output sensitivity for non-degenerate polytopes. That is, a linear number of linear programs are needed per output facet of the projection. It is demonstrated that ESP is particularly well suited to control problems and comparative simulations are given, which greatly favour ESP.

Part two is an investigation into the multi-parametric linear program (mpLP). The mpLP has received a lot of attention in the control literature as certain model predictive control problems can be posed as mpLPs and thereby pre-solved, eliminating the need for online optimisation. The structure of the solution to the mpLP is studied and an approach is presented that eliminates degeneracy. This approach causes the control input to be continuous, preventing chattering, which is a significant problem in control with a linear cost. Four new enumeration methods are presented that have benefits for various control problems and comparative simulations demonstrate that they outperform existing codes.

The third part studies the relationship between projection and multi-parametric linear programs. It is shown that projections can be posed as mpLPs and mpLPs as projections, demonstrating the fundamental nature of both of these problems.

The output of a multi-parametric linear program that has been solved for the MPC control inputs offline is a piecewise linear controller defined over a union of polyhedra. The online work is then to determine which region the current measured state is in and apply the appropriate linear control law. This final part introduces a new method of searching for the appropriate region by posing the problem as a nearest neighbour search. This search can be done in logarithmic time and we demonstrate speed increases from 20Hz to 20kHz for a large example system.

*Proiacio ergo gradum accedo*



# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Jan Maciejowski, for his support and encouragement over the course of this work. Without his guidance this research would not have been possible and without his penchant for finding money none of my teas, parties, BBQs or renovations would have happened.

Dr. Eric (Colin) Kerrigan has been an enormous help throughout this work both with the technical and the ‘big picture’. His initial comment in my first year “come up with a better way to compute projection, they’ll give you a PhD for that” has obviously had a significant impact.

Thank you to Prof. Manfred Morari and to Dr. David Mayne for the chance to visit their groups. Exposure to other students interested in the same problems has broadened my knowledge of control and lead to many insights. In particular I would like to thank Pascal Greider, Sasa Raković, Jorgen Spjøtvold and Dr. Komei Fukuda, whose valuable discussions and insights were much appreciated.

A second thanks to Jan and Eric for organising the MPC group, whose diversity filled out my knowledge of what real people do with control. Thank you to Dr. Danny Ralph and to Dr. Paul Austin for demonstrating that I’m not just playing math, but that real people make real money doing this.

Thanks to Dr. Mihai Huzmezan, whose idea it was that I should come to Cambridge and for rounding up many of his old class mates to help make our ‘careers tea’ a success.

The PhD is not all academics - the control group was valued as much for its social aspects as its knowledge of control (which is quite limited after a few pints anyway). A thank you to all the guys (and girl), especially those of you who really suck at poker. Those new and those who just never seem to leave: Frank, Paul, Oli, Paul, Nik, John, Chris and Dan.

Despite the pain and lack of sleep, the Pembroke boat club has been a fantastic way to not turn into a ball while I wrote this thesis. Thanks to the crew and to Mr. Paul Wilkins who

believed that an old git can pull with the rest of the pups. Pembroke has been a great college over the years and the graduate parlour a great escape full of many unforgettable people.

No work is worthwhile without the comfort of a good family and friends. I am lucky to have both who have supported me during this work. Thank you to my parents and to my sister and her fiancé for their unwavering faith that this work could and would be done. Thanks to Lee and Stacy, and Ian and Mette for getting me out of the country and into a pub whenever it got too much. My new godson, Colin, and his parents Jeremy and Evelyn have given me a glimpse at the future, and it looks good.

The final word is as always for my partner Karen, who has heard more about polytopes than by rights anyone should have to endure. As there are not words enough, I'll just say a heartfelt thank you.

Finally, I would like to acknowledge the Government of Canada, the Prince's Trust, Pembroke College and the Department of Engineering for their financial support during this work.

Colin N. Jones  
Cambridge, 2005



# Declaration

As required by the University Statute, I hereby declare that this dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other university. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specified explicitly in the text.

I also declare that the length of this thesis is less than 65,000 words and that the number of figures is less than 150.

Colin N. Jones  
Pembroke College  
Cambridge  
25 January, 2005



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Outline . . . . .	4
1.2	Statement of Collaboration . . . . .	5
<b>I</b>	<b>Equality Set Projection</b>	<b>7</b>
<b>2</b>	<b>Introduction</b>	<b>9</b>
2.1	Related Work . . . . .	10
2.2	Outline . . . . .	12
2.3	Acknowledgements . . . . .	12
<b>3</b>	<b>Polytopic Projections and Equality Sets</b>	<b>13</b>
3.1	Affine Sets and Polytopes . . . . .	13
3.2	Face Lattice . . . . .	16
3.3	Equality Sets of a Polytope . . . . .	19
3.4	Projection of a Polytope . . . . .	22
<b>4</b>	<b>Equality Set Projection</b>	<b>27</b>
4.1	Algorithm Outline . . . . .	27
4.2	Adjacency Oracle . . . . .	29
4.3	Ridge Oracle . . . . .	39
4.4	Shooting Oracle . . . . .	46
4.5	Complexity Analysis . . . . .	48
<b>5</b>	<b>Extensions and Implementation Details</b>	<b>51</b>
5.1	Degeneracy . . . . .	51

5.2	Calculation of the Affine Hull . . . . .	54
5.3	Projection of non Full-Dimensional Polytopes . . . . .	54
5.4	Projection of Polytopes that do not Contain the Origin . . . . .	56
<b>6</b>	<b>Projection Examples</b>	<b>59</b>
6.1	Random Polytopes . . . . .	59
6.2	Feasibility . . . . .	61
<b>II</b>	<b>Parametric Linear Programming</b>	<b>71</b>
<b>7</b>	<b>Introduction</b>	<b>73</b>
7.1	Related Work . . . . .	74
7.2	Outline . . . . .	77
<b>8</b>	<b>Geometry of the Simplex Method</b>	<b>79</b>
8.1	Representation of Vertices: The Basis . . . . .	80
8.2	Pivoting and Polyhedral Skeletons . . . . .	80
8.3	Optimality Conditions . . . . .	81
8.4	Simplex Algorithm . . . . .	83
8.5	Degeneracy . . . . .	83
8.6	Simplex Algorithm with a Unique Optimal Basis . . . . .	92
8.7	One-Dimensional Parametric Programming . . . . .	95
8.8	Primal-Dual Pairs . . . . .	97
<b>9</b>	<b>Parametric Linear Programming</b>	<b>99</b>
9.1	Structure of the Solution . . . . .	99
9.2	The Optimiser . . . . .	106
9.3	Neighbourhood Problem . . . . .	109
9.4	Enumeration Algorithms . . . . .	113
<b>10</b>	<b>mpLP Examples</b>	<b>133</b>
10.1	Double Integrator . . . . .	135
10.2	Closed-Form MPC for Random 3D System . . . . .	139
10.3	Closed-Form MPC for 4D System . . . . .	143
10.4	Degenerate Closed-Form MPC Example . . . . .	146

## CONTENTS

---

<b>III</b>	<b>Projection and Parametric Programming</b>	<b>149</b>
<b>11</b>	<b>Introduction</b>	<b>151</b>
11.1	Outline . . . . .	151
<b>12</b>	<b>mpLP via Projection</b>	<b>153</b>
12.1	Degeneracy . . . . .	154
<b>13</b>	<b>Projection via mpLP</b>	<b>159</b>
<b>14</b>	<b>Projection and mpLP Examples</b>	<b>165</b>
14.1	mpLP via Projection . . . . .	165
14.2	Projection via mpLP . . . . .	168
<b>IV</b>	<b>Point Location Problem</b>	<b>175</b>
<b>15</b>	<b>Introduction</b>	<b>177</b>
15.1	Outline . . . . .	178
15.2	Acknowledgements . . . . .	179
<b>16</b>	<b>Logarithmic Point Location</b>	<b>181</b>
16.1	Introduction . . . . .	181
16.2	Point Location and Nearest Neighbours . . . . .	182
16.3	Degeneracy . . . . .	185
16.4	Approximate Nearest Neighbour . . . . .	185
<b>17</b>	<b>Point Location Examples</b>	<b>187</b>
17.1	Double Integrator . . . . .	187
17.2	Large Random System . . . . .	188
17.3	Randomly Generated Regions . . . . .	188
<b>V</b>	<b>Conclusions and Future Research</b>	<b>191</b>
<b>18</b>	<b>Conclusions</b>	<b>193</b>
18.1	Main Contributions . . . . .	193
18.2	Future Research . . . . .	195
	<b>Bibliography</b>	<b>197</b>
	<b>Author Index</b>	<b>205</b>

Index

206

# List of Figures

3.1	Face Lattice of a Cube . . . . .	18
3.2	Illustration of the ‘Diamond Property’ . . . . .	18
3.3	Illustration of Equality Sets . . . . .	20
3.4	Equality Set Lattice of a Cube . . . . .	22
4.1	Example Projection of a Cube . . . . .	31
4.2	ESP Procedure: Projection of a Cube . . . . .	32
4.3	Example Projection of a Cube: Search Path in Cube Face Lattice . . . . .	33
4.4	Example of the Wrapping Map $\rho$ . . . . .	34
4.5	Subsets of $E_r$ under the Wrapping Map $\rho$ . . . . .	36
4.6	Linear Optimisation Under the Wrapping Map $\rho$ . . . . .	36
5.1	Degenerate Projection Example: Recursive Method . . . . .	52
5.2	Degenerate Projection Example: Perturbation Method . . . . .	53
5.3	Example Projections that are Good/Bad for both Degeneracy Methods. . . . .	55
6.1	Comparative Simulation Results for Randomly Generated Polytopes . . . . .	62
6.2	Feasible Region of the Double Integrator for $N = 2$ . . . . .	66
6.3	Feasible Region for Example 6.2.2 . . . . .	68
6.4	Feasible Set $\mathbb{X}_F$ of Example 6.2.3 . . . . .	69
8.1	Example of Primal Degeneracy . . . . .	85
8.2	3D Example of Primal Degeneracy . . . . .	85
8.3	Lexicographic Perturbation $P^\epsilon$ . . . . .	88
8.4	Maximum Number of Feasible and Lex-Feasible Representations of a Vertex with Degree of Degeneracy $\sigma$ . . . . .	89

8.5	Lexicographic Perturbation: Effect of Ordering . . . . .	90
9.1	Illustration of the Geometry of the Epigraph and the Solution Complex . . .	104
9.2	Example Epigraph and Solution Complex . . . . .	105
9.3	Example Enumeration using the Basic Method . . . . .	115
9.4	Example Enumeration using the Facet-Based Method. . . . .	118
9.5	Example Enumeration using the Primal-Dual Method. . . . .	124
9.6	Reverse Search Illustration . . . . .	127
9.7	Reverse Search Tree for a Polytope $E^T D$ whose Vertices are on a Sphere . . .	128
9.8	Example Enumeration using the Reverse-Search Method. . . . .	129
10.1	Solution to Example 10.1 . . . . .	137
10.2	Time for a Single LP Pivot as a Function of Size . . . . .	138
10.3	Method Comparison for Example 10.2 . . . . .	142
10.4	Method Comparison for Example 10.3 . . . . .	145
10.5	MPT Solution for Degenerate Example 10.4 . . . . .	146
10.6	RCMSolution for Degenerate Example 10.4 . . . . .	147
10.7	RCMSolution for Degenerate Example using Hybrid Toolbox [Bem03] . . . .	148
10.8	Basic Solution (Alg 9.2) for Degenerate Example 10.4 . . . . .	148
12.1	Example of a Degenerate Multiparametric Linear Program . . . . .	155
14.1	Projection of $Q$ for Example 14.1.1 . . . . .	167
14.2	Solution for Example 14.2.1 . . . . .	170
14.3	Critical Regions found by MPT for Example 14.2.2 . . . . .	172
14.4	Solution Complex for Example 14.2.2 . . . . .	172
14.5	Normal Fan and Dual Polytope $E^T D$ for Example 14.2.2 . . . . .	173
14.6	Solution to Example 14.2.2 . . . . .	173
16.1	Example of a Random Voronoi Diagram . . . . .	183
17.1	Search Tree Construction for Example 17.1 . . . . .	188
17.2	Controller Partition for the Double Integrator . . . . .	189
17.3	Comparison of ANN [AMN <sup>+</sup> 98] (Solid lines) to [BBBM01] (Dashed lines) . .	190



# List of Tables

3.1	Face names of an $n$ -dimensional polytope . . . . .	17
6.1	Comparison of Projection Methods for the Calculation of the Feasible Region for Example 6.2.2 . . . . .	67
10.1	Comparison of mpLP Methods for Example 10.2 . . . . .	141
10.2	Comparison of mpLP Methods for Example 10.2 using Redundancy Elimination Heuristic [Gri04] . . . . .	141
10.3	Comparison of mpLP Methods for Example 10.3 . . . . .	144
10.4	Comparison of mpLP Methods for Example 10.3 using Redundancy Elimination Heuristic [Gri04] . . . . .	144
14.1	Comparison of mpLP Methods for Example 10.2 . . . . .	167
14.2	Comparison of Projection and mpLP Methods for the Calculation of the Feasible Region of Example 14.2.2 . . . . .	171



# List of Algorithms

4.1	Equality Set Projection (ESP) . . . . .	30
4.2	Adjacency oracle (ESP) . . . . .	40
4.3	Ridge oracle (ESP) . . . . .	47
4.4	Shooting Oracle (ESP) . . . . .	49
8.1	Simplex Algorithm . . . . .	84
8.2	Simplex Algorithm with a Unique Optimiser . . . . .	94
8.3	One-Dimensional Parametric Linear Program with a Unique Optimiser . . . . .	97
9.1	Facet Oracle . . . . .	112
9.2	Multiparametric Linear Programming: Basic Enumeration . . . . .	116
9.3	Multiparametric Linear Programming: Facet-Based Enumeration . . . . .	119
9.4	Primal-Dual: Compute Potential Facets . . . . .	122
9.5	Multiparametric Linear Programming: Primal-Dual Enumeration . . . . .	123
9.6	Multiparametric Linear Programming: Reverse Search for Enumeration . . . . .	128
12.1	Multiparametric Linear Programming using a Projection Algorithm . . . . .	157
13.1	Projection using a Multiparametric Linear Programming Algorithm . . . . .	163



# Notation

## Acronyms

LP	Linear program
QP	Quadratic program
mpLP	Multi-parametric linear program
mpQP	Multi-parametric quadratic program
MPC	Model Predictive Control
ESP	Equality Set Projection

## Sets

$\mathbb{R}$	Set of real numbers
$\mathbb{N}$	Integers
$\mathbb{R}^n$	Set of real vectors of length $n$
$\mathbb{R}^{n \times m}$	Set of real matrices of size $n \times m$
$\mathbb{N}_R$	The set of integers from 1 to $R$ , $\mathbb{N}_R \triangleq \{1, \dots, R\}$ , for $R \in \mathbb{N}$

## Algebraic Operators

Let  $A, B \in \mathbb{R}^{m \times n}$  be a matrices and  $E \subseteq \mathbb{N}_m$ ,  $F \subseteq \mathbb{N}_n$  be sets.

$\text{null}(A)$	Nullspace of $A$ , $\text{null}(A) \triangleq \{x \in \mathbb{R}^n \mid Ax = 0\}$
$\text{range}(A)$	Columnspace of $A$ , $\text{range}(A) \triangleq \{y \in \mathbb{R}^m \mid \exists x \in \mathbb{R}^n, y = Ax\}$
$N(A)$	Matrix whose columns form an orthonormal basis for $\text{null}(A)$
$R(A)$	Matrix whose columns form an orthonormal basis $\text{range}(A)$
$A^\dagger$	Moore-Penrose pseudo-inverse
$A \otimes B$	Kronecker product of $A$ and $B$
$A_E$	Matrix whose rows are the rows of $A$ whose indices are in the set $E$
$A_{\setminus E}$	Matrix whose rows are the rows of $A$ whose indices are not in the set $E$
$A_{\star, F}$	Matrix whose columns are the columns of $A$ whose indices are in the set $F$

## Set Operators

Let  $S$  and  $U$  be sets.

$S \setminus U$	Set difference, $S \setminus U = \{x \mid x \in S, x \notin U\}$
$S \subset U$	$S$ is a strict subset of $U$
$ S $	Cardinality of $S$

## Polyhedral Operators

Let  $P = \{z \mid Az \leq b\}$ ,  $A \in \mathbb{R}^{m \times n}$  be a polyhedron and  $E \subseteq \mathbb{N}_m$  be a set.

$\pi_x P$	Projection of $P$ onto the variables $x$ , $\pi_x P \triangleq \{x \mid \exists y, (x, y) \in P\}$
$P_E$	The polyhedron $P_E \triangleq \{z \mid A_E z = b_E\} \cap P$

## Other

Let  $a, b \in \mathbb{R}^n$  be vectors.

$I_m$	The identity matrix of size $m$
$\mathbf{1}$	A vector of all ones
$a \propto b$	True if $a$ is proportional to $b$ , false otherwise
$a \succ 0$	True if lexico positive, if $a \neq 0$ and if the first nonzero component of $a$ is positive
$a \succ b$	True if $a - b \succ 0$ , false otherwise

# Introduction

Polyhedra have been studied for thousands of years, with many Greek mathematicians dedicating their lives to unraveling their mysteries. While the tools used to study polyhedra have changed over the years, they have remained vitally important to many branches of science and engineering.

The literal translation from Greek of the word polyhedron is ‘many faced solid’. While this is a fairly good description, a more useful mathematical definition is that a polyhedron is the solution set for a system of simultaneous linear inequalities. Polyhedra have been used to model everything from physical objects such as chairs or motorcycles to more abstract high-dimensional mathematical concepts such as the financial markets or expert system intelligence.

To give a flavour of the types of objects that can be described using polyhedra and of the operations that need to be done on them, we give the following tiny sampling of uses:

**Force Closure** It is surprisingly difficult to teach a robot to pick up an object. The ideal situation is for the robot to place its fingers at points on the object such that there is no direction that a force can be applied that will knock the object out of its hand. This property is known as *force closure*. If the object has reasonably flat sides and is convex, then all possible interactions between the robot’s fingers and the object can be modelled as a polyhedron. The set of finger locations that provide force closure can then be calculated by projecting this polyhedron onto the dimensions that represent the robot’s fingers. [PSS<sup>+</sup>95]

**Path Planning** Getting from one side of a room to the other without running into anything is another obvious robotics problem. Furniture is reasonably well described as objects with flat faces, as are most robots. Good algorithms exist that allow the computation of a path through the furniture without bumping into any of it. However, they generally

assume that the robot is just a point. This assumption can be made valid by ‘adding’ the shape of the robot to the furniture. This method of adding two polyhedra together is called Minkowski Addition, and can also be solved by computing a projection. [Cam85]

**Theorem Proving** Expert systems represent knowledge as rules of the form “If the price of barley goes down by 4%, then beer will be 3% cheaper”. After collecting many such statements, one may wish to query the system with questions of the form “Under what circumstances should I buy more beer?”. While the answer to this particular question is obvious (every chance you get), expressions of this type can be written as polyhedra and the questions can be answered via a projection operation if we wish to know *all* such circumstances or as a linear program (LP) if we want to know just one. [CLL00]

**Visualisation** 3D computer graphics and visualisation is a rapidly growing field, in large part due to the ability to efficiently store and process polyhedral models. All modern graphics cards represent the world in terms of flat surfaces, using thousands of triangles to form shapes. Before these objects can be shown on the screen, they need to be projected from three dimensions down to the two that can be displayed. [AGG03]

**Financial Constraints** One of the biggest uses of polyhedra in recent years is the representation of financial constraints: “If I expect the cost of crude oil to follow a given pattern in the next year and my production facilities have a given set of properties, how much oil should I buy this month?”. Many financial systems can be well modelled as polyhedral constraints, and optimising over these constraints to choose the best option can often be posed as a linear programming problem. [TU03]

The common thread amongst the above examples is the projection operation. This is one of the most fundamental operations that can be done on polyhedra as many basic functions are equivalent to, or can be reduced to projection. For example, both additions (Minkowski addition) and multiplications (rank deficient linear maps) can be computed using a projection operation [Ker00].

Control of linear dynamic systems under state and input constraints is an open problem that is of primary importance. One of the best and most natural ways to model such constraints is through the use of linear inequalities, or polyhedra. Working on constrained control problems therefore requires operating on polyhedra. The following list provides an idea of the types of control problems involving polyhedral theory that one may want to tackle (examples of these problems can be found in [Ker00]):

**Feasibility** A given state is called feasible if there exists a sequence of control inputs such that both the inputs and the future state of the system stays within given constraints.



---

This region is clearly of importance as it tells us the set of states to which we can choose to drive the system without causing problems with constraints in the future. The determination of all such states that have this property can be posed as a polyhedral projection.

**Invariance** If there is an unknown persistent disturbance acting on a system, then it will be impossible to regulate that system exactly to the origin. A natural question would be “How close we can get to the origin for a given level of disturbance?” If a linear control law is applied, then the smallest set of states around the origin that the system is guaranteed to stay in is called the minimally invariant set and can be computed via recursive projection operations.

**Region of Attraction** The region of attraction of a system is the set of states that can be driven to the origin without violating any constraints. For example, we may wish to know the maximum bank angle that an aircraft can achieve before there is insufficient control authority to get it back flat and level. This too can be computed via a projection.

**Reachability Analysis** For safety critical systems an important question is one of reachability: “Does there exist a disturbance that could push the system into an unsafe state from the current state?” Existence type questions such as this can often be directly translated into projection.

**Closed-form MPC** The best existing approach for controlling systems with input and state constraints is called model predictive control (MPC). Recent developments in the control literature have demonstrated that MPC laws can be pre-solved offline using multi-parametric programming (mpLP) techniques. In this thesis we show a very close link between multi-parametric linear programming and projection; namely that an mpLP algorithm can be used to solve a projection problem and a projection algorithm can be used to solve a multi-parametric linear program.

Again, one can see that the common thread amongst the above problems is the projection operation. While there are several methods available that can be used to compute projections, none of them are well-suited to the structure of the problems that are generally seen in control. In fact, current methods can at best handle small, toy problems and a significant improvement is needed before sizeable systems can be computed. It is for these reasons that this thesis focuses on new approaches to polyhedral projection algorithms and to the related field of linear parametric programming that, while general, are particularly efficient for control problems.

## 1.1 Outline

This thesis is separated into five parts. The purpose and content of each is outlined as follows:

### **Part I: Equality Set Projection (ESP)**

In this part we first give an introduction to polytopes and their related mathematics, which is used throughout the thesis. A new algorithm for the projection of polytopes in halfspace form is presented, dubbed Equality Set Projection (ESP). The correctness and completeness of the algorithm are proven and its complexity is explored. This analysis demonstrates that ESP is the first projection algorithm with the important property of output sensitivity for non-degenerate polytopes. In other words, the complexity of the algorithm is a linear function of the size of the output (number of facets in the projection). Comparative simulations are run on examples that are of interest to control problems, in which ESP is demonstrated to be many orders of magnitude faster than the existing state-of-the-art.

### **Part II: Parametric Linear Programming**

Recent interest in multi-parametric programming in the control community has spawned several algorithms for both linear (mpLP) and quadratic (mpQP) problems. However, the properties of the algorithms in the presence of degeneracy are still poorly understood, which has resulted in algorithms whose correctness cannot be guaranteed or, more importantly for control problems, the continuity of the optimiser. In this part we explore this structure of linear problems and propose a procedure that we prove ensures correctness, completeness and continuity for degenerate problems, without sacrificing optimality. This procedure is then incorporated into four enumeration methods, with varying properties depending on the problem under consideration. Finally, simulation results demonstrate the favourable performance of these algorithms over other available approaches.

### **Part III: Projection and Parametric Programming**

This part explores links between projection and parametric linear programming. We demonstrate that any projection algorithm can compute an mpLP and *vice versa*. This important result joining these two problems opens up new possibilities for algorithms drawing from both areas. Simulations demonstrate that for control problems, ESP and the algorithms presented in Part II are still best for projection and mpLPs respectively.

## 1.2 STATEMENT OF COLLABORATION

---

### Part IV: Point Location Problem

The recent interest in parametric programming in the control community is due to its ability to solve model predictive control (MPC) problems offline. Traditionally, the implementation of an MPC controller requires the solution of an optimisation problem at every sampling instant, which has restricted the application of MPC to slower systems. However, posing the MPC problem as an mpLP/QP breaks the state-space into a union of polyhedral regions, such that in each region the optimal MPC control law is a simple linear controller, which has been pre-computed. The online procedure then becomes one of determining the region that contains the current measured state and then applying the appropriate control law; this is the ‘point location problem’. However, if the number of regions is large, solving the point location problem can be more time consuming than solving the original optimisation problem. In this part we prove that the polyhedral regions for an mpLP are in fact a so-called power diagram, which is a type of Voronoi diagram. Voronoi diagrams are an extremely common construct both in nature and engineering. For example, the territorial patterns of some fish and the determination of the transmitted signal on a radio are, both Voronoi diagrams. We borrow an approach developed for searching large databases, called Approximate Nearest Neighbours (ANN) [AMN<sup>+</sup>98] This allows the search of a power diagram in time logarithmic in the number of regions and linear in the dimension. Through simulation we demonstrate that this enables an increase of several orders of magnitude is possible in sampling rate.

### Part V: Conclusions and Future Research

This final part summarises the key contributions of this thesis and discusses possible directions for future research.

## 1.2 Statement of Collaboration

The following sections are work done in collaboration with other researchers:

### Part I: Equality Set Projection (ESP)

The author would like to thank Pascal Greider for his valuable comments early in this work.

This part is based almost entirely on the following technical report:

C.N. Jones, E.C. Kerrigan and J.M. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Tech-

nical report CUED/F-INFENG/TR.463, Department of Engineering, University of Cambridge, 2004.

### **Part IV: Point Location Problem**

The work presented in this part is a collaborative effort with Pascal Grieder and Sasa Raković. The text is based almost entirely on the following paper:

C.N. Jones, P. Grieder and S.V. Raković. A Logarithmic-Time Solution to the Point Location Problem for Closed-Form Linear MPC. To appear in *Proceedings of the 16<sup>th</sup> IFAC World Congress*, Prague, Czech Republic, 2005.

The examples in this part have been prepared with the MPT toolbox [KGBM04] and Figure 17.3 was calculated using the ANN library [MA98].

## Part I

# Equality Set Projection



## Introduction

The calculation of the orthogonal projection of a polytope is a fundamental operation that arises in many applications. For example, in control theory, projection is required for reachability analysis [Bla99] and in decision theory for the elimination of existential quantifiers [VLS00]. It can be shown that the calculation of affine maps or Minkowski sums of polytopes are both equivalent to orthogonal projection [Ker00], making a projection algorithm a necessary tool for working with polytopes.

It is well known that polytopes can be represented in two forms: as the convex combination of a finite number of vertices and as the intersection of a finite number of halfspaces. For any  $d$ -dimensional polytope, represented as the intersection of  $q$  halfspaces, the number of vertices required to describe the same polytope is  $\mathcal{O}(q^{\lfloor \frac{d}{2} \rfloor})$  in the worst case. Polytopes that exhibit an exponential relationship are common in various fields. For example, a hypercube in  $d$ -dimensions can be described by  $2d$  halfspaces or  $2^d$  vertices.

Vertex representation is appropriate in some applications, while in others halfspace is preferred. In this thesis, we are interested in polytopes that are in halfspace representation and whose projection should also be given in halfspace form, as this is the most common form seen in linear constrained control problems. As we are given a polytope in halfspace form and the vertex representation can be exponentially more complex, we ideally want the complexity of the projection algorithm to depend only on the number of halfspaces, and never to compute the vertices. As the effects of this exponential behaviour do not become significant until the dimension of the problem is large, this algorithm is most suited to the projection of high dimensional polytopes.

In this part we give a new algorithm, dubbed Equality Set Projection (ESP), for the computation of the projection of a bounded polytope described as the intersection of a finite number of halfspaces in arbitrary dimension. We do not make the assumption that the

polytope is in general position or that the description is irredundant, although the description of the projection returned by the algorithm is irredundant. It will be shown that for a polytope of fixed size (dimension and number of halfspaces), ESP computes the projection using a number of linear programs that is linear in the number of halfspaces in the projection.

## 2.1 Related Work

We briefly review the literature on projection methods and examine its relation to our work. Current projection methods that can operate in general dimensions can be grouped into three classes: Fourier elimination, block elimination and vertex based approaches. While each approach can be effective for a particular class of polytopes, there is as yet no algorithm whose complexity is a function of the number of halfspaces required to describe the projection. Here, we give a brief overview of these approaches and their derivatives.

Fourier-Motzkin elimination was originally described by Fourier in 1824 and has been improved many times since. This approach can be thought of as the analogue of Gaussian elimination for linear inequalities. At each iteration of the algorithm the polytope is recursively projected by one dimension until the desired dimension is reached. The primary limitation of Fourier-Motzkin elimination is that it generates many redundant constraints at each iteration. It is not practical to remove the redundancies at each step, although modifications of the algorithm due to Černikov [Č63] in 1963 greatly improved the efficiency by identifying many redundant constraints with very little added work. For some polytopes, algorithms based on Fourier elimination can be efficient and recent work [JMSY93] has improved the average case complexity for very sparse constraints. However, for a polytope described by  $q$  halfspaces that is to be projected down by  $k$  dimensions, the time complexity of Fourier elimination is  $\mathcal{O}(q^{2^k})$ , often making it unusable even for small problems.

In [PSS<sup>+</sup>95], a modification of Fourier's method is proposed in which a set of  $k + 1$  constraints is selected and all  $k$  dimensions of this set that are to be projected are removed using Gaussian elimination. There are, however,  $\binom{q}{k+1}$  sets of constraints that must be considered, making this algorithm only suitable for very specific applications.

The second well known approach is block elimination. In this method a polyhedron is defined called the projection cone [BP83]. The extreme rays of this cone can then be used to find the defining halfspaces of the projection. While there exist efficient methods for computing these extreme rays, such as the double description algorithm [FP96] or the reverse search approach [AF92, AF96], this approach may generate a large, and possibly exponential, number of redundant inequalities that need to be removed.

Recent work by Balas [Bal98] has shown that if certain invertibility conditions are sat-



## 2.1 RELATED WORK

---

ified, then every extreme ray of the projection cone generates an irredundant constraint of the projection. This observation has been extended to polytopes that do not satisfy these conditions by the introduction of a transformation of the cone such that each ray of the projection of the transformed cone corresponds to a constraint of the projection of the polytope. The limitation of this approach is that the calculation of the extreme rays of the projection of the transformed cone may be very difficult, although it offers an important insight into the structure of projection.

The final class of approaches covers a variety of methods that compute vertices of the projection. While these approaches are suitable for a certain class of polytopes, we are interested here in polytopes in which the vertices greatly outnumber the inequalities. As a result, any approach that requires the enumeration of the vertices of either the polytope or its projection can be as much as exponentially slower for this class than an approach that considers only inequalities.

All vertices of the polytope can be computed using a vertex enumeration algorithm, each vertex can then be trivially projected, before a convex hull algorithm is used to calculate the inequality constraints of the projection. Vertex enumeration and convex hulls can be computed using the same algorithms (for example, [BDH96, AF92, AF96, FP96, CL97, Cla, BFM98a]). This approach can be efficient for polytopes with large numbers of redundant inequalities or a small number of vertices. However, as there can be exponentially more vertices than there are inequalities, the applicability is limited to polytopes with a small vertex count.

A contour-tracking approach is proposed in [PSS<sup>+</sup>95] in which the skeleton of the projection is traced. The skeleton is formed from the vertices and the one-dimensional lines that join them. The complexity is a linear function of the number of vertices of the projection and as a result is suitable only when the number of vertices is small when compared to the number of inequalities.

An approach, similar to that developed in this part, was outlined in [AZ96]. The purpose was to compute both the vertex and half-space representation of the projection, although the algorithm can be adapted to compute only the inequalities if desired. Three restrictive assumptions are made, which we here relax. First, in [AZ96] it is assumed that the polytope is in general position; while there exist standard techniques to ensure that this is the case for general polytopes [EM90], no insight is given as to how they may be applied to this problem. Second, it is implicitly assumed in [AZ96] that the description of the polytope is irredundant. While this requirement is not necessarily prohibitive and can be satisfied by running as many linear programs as there are constraints, it will greatly slow the algorithm if the dimension is

large, or if there are many redundant inequalities. Finally, the assumption is made that every face of the polytope that projects to a facet of the projection is of dimension  $d - 1$ . While there is a class of polytopes for which this is the case, it is not true in general and we here introduce two methods to handle this degenerate case.

**Remark 2.1.** *Note that the condition we refer to in this part as “degenerate” is not a function of just the polytope, but rather of the direction that the polytope is being projected. Any polytope could be rotated such that the pre-image of one of the facets of the projection has a dimension larger than  $d - 1$ .*

## 2.2 Outline

The remainder of this part is organised as follows: Chapter 3 provides an introduction to polytopes and their projections and introduces the notion of an equality set. Section 4.1 gives an outline of the ESP algorithm while Sections 4.2 through 4.4 provide the details. The complexity of the algorithm is investigated in Section 4.5 and various extensions are discussed in Chapter 5. Finally, both generic and control-specific numerical simulation comparisons between ESP and existing approaches are reported in Chapter 6.

## 2.3 Acknowledgements

The author would like to thank Pascal Greider for his valuable comments early in this work.

This part is based almost entirely on the following technical report:

C.N. Jones, E.C. Kerrigan and J.M. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical report CUED/F-INFENG/TR.463, Department of Engineering, University of Cambridge, 2004.

## Polytopic Projections and Equality Sets

This section provides an overview of projections of polytopes. The notion of an equality set will be introduced and the properties relevant to projection will be demonstrated.

### 3.1 Affine Sets and Polytopes

Vector spaces and the related notions of subspaces and affine sets are fundamental to any discussion of polytopes and therefore we begin this introduction with a brief review of these concepts.

A *vector space*  $V$  over the reals is a set of vectors  $z \in \mathbb{R}^n$  that contains the origin and is closed under vector addition and scalar multiplication. A *subspace* of a vector space  $V$  is any subset of  $L \subseteq V$  that is itself also a vector space. Recall that  $L$  can be represented in two common forms: as the set of all vectors satisfying a finite set of homogeneous linear equations:

$$L = \{z \in \mathbb{R}^n \mid Az = 0\}, \quad \text{for some } A \in \mathbb{R}^{q \times n} \quad (3.1)$$

or in terms of the span of a finite set of vectors  $v_i \in \mathbb{R}^n$ :

$$L = \text{span} \{v_1, \dots, v_p\} = \left\{ z \in \mathbb{R}^n \mid z = \sum_{i=1}^p \lambda_i v_i, \lambda_i \in \mathbb{R} \right\}. \quad (3.2)$$

If the vectors  $v_i$  are linearly independent, then the set  $\{v_1, \dots, v_p\}$  forms a *basis* for  $L$ .

The *dimension* of a subspace  $L \subseteq \mathbb{R}^n$ , denoted  $\dim L$ , is defined as the smallest number of vectors whose span is  $L$ . Note that if the subspace is defined as in (3.1), then the dimension can be calculated as  $\dim L = n - \text{rank } A$ . If it is represented as in (3.2), then its dimension

### 3. POLYTOPIC PROJECTIONS AND EQUALITY SETS

---

is the maximum number of linearly independent vectors  $v_i$ .

A subset  $M \subseteq \mathbb{R}^n$  is called an *affine set* if  $(1 - \lambda)x + \lambda y \in M$  for every  $x, y \in M$  and  $\lambda \in \mathbb{R}$ . Two affine sets  $M_1$  and  $M_2$  are said to be *parallel* if they can be written as

$$M_1 = M_2 + a, \quad \text{for some } a \in \mathbb{R}^n. \quad (3.3)$$

**Theorem 3.1.** [Roc70, Thm 1.2] *Each non-empty affine set  $M$  is parallel to a unique subspace  $L$ . This  $L$  is given by*

$$L = M - M = \{x - y \mid x, y \in M\}.$$

Theorem 3.1 and (3.3) allow us to write every affine set  $M$  as a *translate* of a unique subspace  $L$

$$M = L + a, \quad \text{for some } a \in \mathbb{R}^n. \quad (3.4)$$

The two representations of subspaces defined above give rise to two representations of affine sets. Equation 3.1 gives the representation in terms of all solutions of a finite set of linear equations:

$$\begin{aligned} M &= L + a, & \text{for some } a \in \mathbb{R}^n \\ &= \{z \in \mathbb{R}^n \mid Az = 0\} + a \\ &= \{z \in \mathbb{R}^n \mid A(z - a) = 0\} \\ &= \{z \in \mathbb{R}^n \mid Az = b\}, & b \triangleq Aa. \end{aligned} \quad (3.5)$$

Affine sets of the form  $\{z \mid a^T z = b\}$ ,  $a \in \mathbb{R}^n$ ,  $b \in \mathbb{R}$ ,  $a \neq 0$  are called *hyperplanes*. Equation 3.5 shows that all affine sets can be written as the intersection of a finite number of hyperplanes.

Given a set  $S \subseteq \mathbb{R}^n$ , the *affine hull* of  $S$ , denoted  $\text{aff } S$ , is the intersection of all affine sets containing  $S$ . Clearly, the affine hull of an affine set is itself. It can be shown that the affine hull of  $S$  consists of all the vectors  $y$  of the form  $y = \sum_{i=1}^p \lambda_i v_i$ , such that  $v_i \in S$  and  $\sum_{i=1}^p \lambda_i = 1$  [Roc70]. Equation 3.2 allows us to write any affine set as the affine hull of a

### 3.1 AFFINE SETS AND POLYTOPES

---

finite number of vectors:

$$\begin{aligned}
 M &= L + a, && \text{for some } a \in \mathbb{R}^n \\
 &= \left\{ z \in \mathbb{R}^n \mid z = \sum_{i=1}^p \lambda_i v_i, \lambda_i \in \mathbb{R} \right\} + a \\
 &= \left\{ z \in \mathbb{R}^n \mid z = a + \sum_{i=1}^p \lambda_i v_i, \lambda_i \in \mathbb{R} \right\} \\
 &= \left\{ z \in \mathbb{R}^n \mid z = \lambda_{p+1} a + \sum_{i=1}^p \lambda_i (v_i + a), \sum_{i=1}^{p+1} \lambda_i = 1, \lambda_i \in \mathbb{R} \right\} \\
 &= \left\{ z \in \mathbb{R}^n \mid z = \sum_{i=1}^{p+1} \lambda_i \mu_i, \sum_{i=1}^{p+1} \lambda_i = 1, \lambda_i \in \mathbb{R} \right\} \\
 &= \text{aff } \{\mu_1, \dots, \mu_{p+1}\},
 \end{aligned}$$

where  $\mu_i \triangleq v_i + a$ ,  $i = 1, \dots, p$ ,  $\mu_{p+1} \triangleq a$ .

If  $z_1, z_2 \in M$ , where  $M = L + a$  is an affine set,  $L$  is a subspace and  $a$  is a vector then  $z_1$  and  $z_2$  are called *affinely independent* if and only if  $z_1 - a$  and  $z_2 - a$  are linearly independent. This allows the notion of dimension to be extended to affine sets as follows: the dimension of an affine set  $M$ , denoted  $\dim M$ , is defined as the dimension of the subspace which is parallel to it,  $\dim M \triangleq \dim L$  if  $M = L + a$ , where  $L$  is a subspace.

A subset  $C \subseteq \mathbb{R}^n$  is *convex* if  $(1 - \lambda)x + \lambda y \in C$  whenever  $x, y \in C$  and  $0 \leq \lambda \leq 1$ . Clearly, all affine sets are convex. The *convex hull* of a set  $S \subseteq \mathbb{R}^n$  is the intersection of all the convex sets containing  $S$  and is denoted  $\text{conv } S$ . The *convex combination* of a set of vectors  $\{v_1, \dots, v_p\}$  is all points  $\sum_{i=1}^p \lambda_i v_i$ ,  $\sum_{i=1}^p \lambda_i = 1$ ,  $\lambda_i \geq 0$ .

**Theorem 3.2.** [Roc70, Thm 2.3] For any  $S \subset \mathbb{R}^n$ ,  $\text{conv } S$  consists of all convex combinations of the elements of  $S$ .

**Corollary 3.3.** [Roc70, Cor. 2.4] The convex hull of a finite subset  $\{v_1, \dots, v_p\}$  of  $\mathbb{R}^n$  consists of all vectors of the form  $\sum_{i=1}^p \lambda_i v_i$ , with  $\lambda_i \geq 0$ ,  $\sum_{i=1}^p \lambda_i = 1$ .

A closed halfspace is a convex set of the form  $\{z \in \mathbb{R}^n \mid a^T z \leq b\}$ ,  $a \in \mathbb{R}^n$ ,  $b \in \mathbb{R}$ . Any set that can be expressed as the intersection of a finite number of closed halfspaces is called a *polyhedral* (convex) set:

$$P \triangleq \{z \in \mathbb{R}^n \mid Az \leq b\}, \quad A \in \mathbb{R}^{q \times n}, b \in \mathbb{R}^q.$$

### 3. POLYTOPIC PROJECTIONS AND EQUALITY SETS

---

The dimension of a polyhedron  $P$  is the dimension of its affine hull:

$$\dim P \triangleq \dim \text{aff } P. \tag{3.6}$$

If a halfspace constraint can be removed from the description of  $P$  without changing the polyhedron, then it is called *redundant*. If the description of  $P$  contains no redundant constraints, then it is called *irredundant*.

Note that while the convex hull of a set of points is always bounded, polyhedra may not be. If a polyhedron is bounded, then it is called a polytope. Unless specified, throughout the remainder of this part we shall assume that we are dealing with bounded polytopes.

The Minkowski-Weyl Theorem (Theorem 3.4 below) is fundamental as it allows a polytope to be expressed in two forms: as the intersection of a finite number of halfspaces or as the convex hull of a finite number of vectors.

**Theorem 3.4.** (*Minkowski-Weyl*) *A subset  $P \subset \mathbb{R}^n$  is the convex hull of a finite set of vectors*

$$P = \text{conv} \{v_1, \dots, v_p\}, \quad \text{where each } v_i \in \mathbb{R}^n,$$

*if and only if it is a bounded intersection of halfspaces*

$$P = \{z \in \mathbb{R}^n \mid Az \leq b\}, \quad \text{for some } A \in \mathbb{R}^{q \times n}, b \in \mathbb{R}^q.$$

Efficient methods exist for converting between one representation and the other, see [Fuk99] for a survey. However, there can be an exponential relationship between the representations: if an  $n$ -dimensional polytope can be described by the intersection of  $q$  halfspaces, it may take up to  $\mathcal{O}(q^{\lfloor \frac{n}{2} \rfloor})$  vertices to formulate the convex hull. Similarly, if a polytope is given as the convex hull of  $m$  vertices, an exponential number of halfspaces may be needed to describe the same object. Many applications produce polytopes that exhibit this exponential relationship, and they are very common in control. The projection method presented here operates only on the polytope in halfspace form without computing the vertex representation and so is particularly suited to polytopes that require a small number of halfspaces to describe them but a very large number of points to formulate as a convex hull.

## 3.2 Face Lattice

In this section we recall a natural decomposition of polytopes: the face lattice.

### 3.2 FACE LATTICE

---

Table 3.1: Face names of an  $n$ -dimensional polytope

Dimension	Name
$(n - 1)$ -face	Facet
$(n - 2)$ -face	Ridge
1-face	Edge
0-face	Vertex

**Definition 3.5.** (*Face*) [Zie95, Def. 2.1]  $F$  is a face of the polytope  $P \subset \mathbb{R}^n$  if there exists a hyperplane  $\{z \in \mathbb{R}^n \mid a^T z = b\}$ , where  $a \in \mathbb{R}^n$ ,  $b \in \mathbb{R}$ , such that

$$F = P \cap \{z \in \mathbb{R}^n \mid a^T z = b\}$$

and  $a^T z \leq b$  for all  $z \in P$ .

Note that  $\emptyset$  and  $P$  are both faces of  $P$  (consider the hyperplanes  $\{z \mid 0^T z = 1\}$  and  $\{z \mid 0^T z = 0\}$ , respectively); all other faces are called *proper* faces.

**Theorem 3.6.** [Zie95, Prop. 2.2] Let  $P \subset \mathbb{R}^n$  be a polytope and let  $F$  be a face of  $P$ :

1. The face  $F$  is a polytope.
2. Every intersection of faces of  $P$  is a face of  $P$ .
3. The faces of  $F$  are exactly the faces of  $P$  that are contained in  $F$ .
4.  $F = P \cap \text{aff } F$ .

Note that as each face of a polytope is itself a polytope, the notion of dimensionality applies to faces. If a face is of dimension  $p$ , then it is called a  $p$ -face. Faces of particular dimensions have explicit names as shown in Table 3.1. Note that for a polytope in  $\mathbb{R}^n$ , the word *face* refers to a face of arbitrary dimension, while *facet* refers specifically to the  $(n - 1)$ -faces.

Theorem 3.6 allows a natural partial ordering over the faces of a polytope based on inclusion. If  $P$  is a polytope, then each face of  $P$  is itself a polytope whose faces are a subset of  $P$ 's faces. This allows us to draw a *face lattice* where each node represents a face of the polytope and two nodes are connected if one of the corresponding faces is a subset of the other. The lattice is organized vertically, such that faces of the same dimension appear at the same level and the dimension increases up the diagram. An example lattice of a cube is shown in Figure 3.1.

A property of the face lattice that is key to the projection method developed here is the so-called *diamond property*:

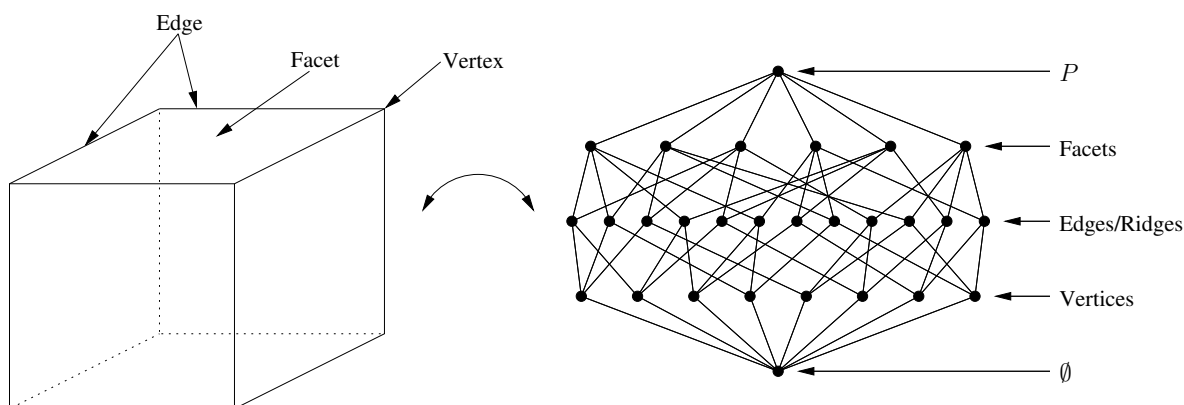


Figure 3.1: Face Lattice of a Cube

**Theorem 3.7.** (*Diamond property*) [Zie95, Thm 2.7(iii)] If  $G$  and  $F$  are faces of a polytope  $P$  and  $G \subset F$  with  $\dim F - \dim G = 2$ , then there are exactly two faces  $H_1, H_2$  with the property  $G \subset H_1 \subset F$  and  $G \subset H_2 \subset F$ .

**Remark 3.8.** The symbol  $\subset$  is used to mean strict subset.  $\subseteq$  will be used when the inclusion is not strict.

Note that, as the name of the proposition implies, the four faces in Theorem 3.7 will ‘look like’ a diamond, as shown in Figure 3.2.

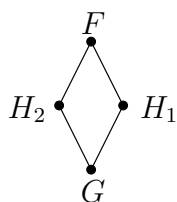


Figure 3.2: Illustration of the ‘Diamond Property’

The immediate implication of Theorem 3.7 is that any two facets ( $(n - 1)$ -faces) of a polytope are either not joined in the face lattice or are connected by the inclusion of exactly one ridge.

Two faces  $F_1$  and  $F_2$  of  $P$  are said to be *adjacent* if the intersection  $F_1 \cap F_2$  is a facet of both.



### 3.3 Equality Sets of a Polytope

We now introduce the notion of the *equality set*, which is the method that will be used in this paper for defining faces.

If  $P \triangleq \{z \in \mathbb{R}^n \mid Az \leq b\}$  is a polytope defined by the intersection of  $q$  halfspaces and  $E \subseteq \{1, \dots, q\}$ , then  $A_E$  is a matrix whose rows are the rows of  $A$  whose indices are in  $E$ . Similarly,  $b_E$  is the vector formed by the rows of  $b$  whose indices are in  $E$ . The relevant rows of  $A_E$  are taken in the same order as those of the matrix  $A$ . If  $E = \{i\}$  is a singleton, then we write  $A_i$  for  $A_{\{i\}}$ . The notation  $P_E$  refers to the set  $P_E \triangleq P \cap \{z \mid A_E z = b_E\}$ .

**Definition 3.9.** (*Equality Set*) Let  $P \triangleq \{z \mid Az \leq b\}$  be a polytope defined by the intersection of  $q$  halfspaces,  $E \subseteq \{1, \dots, q\}$  and

$$G(E) \triangleq \{i \in \{1, \dots, q\} \mid A_i z = b_i, \forall z \in P_E\}.$$

The set  $E$  is an equality set of  $P$  if and only if  $E = G(E)$ .

**Remark 3.10.** Note that this definition of an equality set is similar to the notion of an equality subsystem used in [BO98]. Whereas equality subsystems generally refer to any description of the affine set that describes the affine hull of the face, equality sets describe this affine set specifically in terms of all of the inequalities of  $P$  that are met with equality at all points in the face.

To illustrate the idea of an equality set, consider the pyramid shown in Figure 3.3. Notice that the set  $P_{\{1,2,3\}}$  is the face, or more specifically, the vertex  $v$ , but that  $\{1, 2, 3\}$  is not an equality set, since  $G(\{1, 2, 3\}) = \{1, 2, 3, 4\}$ . There is only one equality set that defines the face  $v$ , and that is  $\{1, 2, 3, 4\}$ .

The goal is to re-write the face lattice in terms of equality sets. First, the relationship between equality sets and faces must be made explicit: there is a one-to-one mapping.

We begin with the following well-known result: equality sets define affine hulls.

**Lemma 3.11.** If  $E$  is an equality set of the polytope  $P \triangleq \{z \in \mathbb{R}^n \mid Az \leq b\}$ , then  $\text{aff } P_E = \{z \in \mathbb{R}^n \mid A_E z = b_E\}$  and  $\dim P_E = n - \text{rank } A_E$ .

*Proof.* The first result follows directly from Definition 3.9. The second result follows from (3.6) and  $\dim P_E = \dim \text{aff } P_E = n - \text{rank } A_E$ .  $\square$

**Lemma 3.12.** (*Uniqueness of Equality Sets*) If  $E$  and  $B$  are equality sets of a polytope  $P \triangleq \{z \in \mathbb{R}^n \mid Az \leq b\}$ , then  $E = B$  if and only if  $P_E = P_B$ .

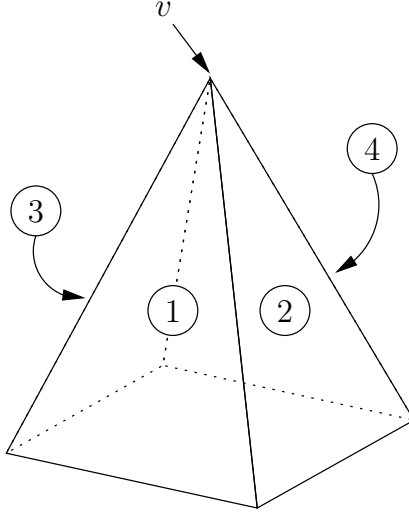


Figure 3.3: Illustration of Equality Sets

*Proof.* Clearly, if  $E = B$ , then  $P_E = P_B$ .

Assume that  $P_E = P_B$ . Since  $E$  and  $B$  are equality sets,  $E = G(E)$ ,  $B = G(B)$  and because  $P_E = P_B$  we have

$$\begin{aligned} E &= G(E) = \{i \mid A_i z = b_i, \forall z \in P_E\} \\ &= \{i \mid A_i z = b_i, \forall z \in P_B\} \\ &= G(B) = B. \end{aligned}$$

□

Theorem 3.13 proves the assertion that there is a one-to-one mapping between equality sets and faces.

**Theorem 3.13.** *If  $E$  is an equality set of the polytope  $P \triangleq \{z \in \mathbb{R}^n \mid Az \leq b\}$ , then  $P_E$  is a face of  $P$ . Furthermore, if  $F$  is a face of  $P$ , then there exists a unique equality set  $E$  such that  $F = P_E$ .*

*Proof.* If  $i \in E$ , then  $A_i z \leq b_i$  is true for all  $z \in P$  and therefore  $P \cap \{z \in \mathbb{R}^n \mid A_i z = b_i\}$  is a face by Definition 3.5. By Proposition 3.6(2), all intersections of faces are faces and thus  $P_E = P \cap_{i \in E} \{z \in \mathbb{R}^n \mid A_i z = b_i\}$  is a face of  $P$ .

Recall from Proposition 3.6(1) that every face  $F$  of  $P$  is a polytope. The affine hull of a polytope can be represented by the intersection of all halfspaces that are satisfied with equality at all points in the polytope. Let the indices of those halfspaces be  $E$ ; clearly  $E = G(E)$  and

### 3.3 EQUALITY SETS OF A POLYTOPE

---

$E$  is an equality set. By Proposition 3.6(4) and using Lemma 3.11,  $F$  can be written as

$$\begin{aligned}
 F &= P \cap \text{aff } F \\
 &= P \cap \text{aff } \{z \mid A_E z = b_E\} \\
 &= P \cap \text{aff } P_E \\
 &= P_E.
 \end{aligned}$$

□

Theorem 3.14 allows the ordering of the face lattice to be written as set inclusion on equality sets.

**Theorem 3.14.** *Let  $E$  and  $B$  be equality sets of  $P$ . The inclusion  $P_E \subset P_B$  holds if and only if  $E \supset B$ .*

*Proof.* Let the polytope be defined by the matrix  $A \in \mathbb{R}^{q \times n}$  and the vector  $b \in \mathbb{R}^q$ :  $P \triangleq \{x \mid Ax \leq b\}$ .

If  $E \supset B$ , we can write  $E = B \cup X$  for some  $X \subseteq \{1, \dots, q\}$  with  $X \cap B = \emptyset$  and therefore

$$\begin{aligned}
 P_E &= P_{B \cup X} \\
 &= P_B \cap P_X \\
 &\subseteq P_B.
 \end{aligned}$$

We now show that there exists an  $i$  in  $X$  that is not in  $B$  and therefore we have strict inclusion. If  $i$  is in  $X$  then it is not in  $B$  because  $X \cap B = \emptyset$  and therefore by the definition of equality set, there exists a  $z \in P_B$  such that  $A_i z \neq b_i$ , which implies that  $z \notin P_X$  and hence we have strict inclusion, i.e.  $P_E \subset P_B$ .

Assume that  $P_E \subset P_B$ . If  $i$  is in the equality set  $B$ , then for all  $z$  in  $P_E$ , we have that  $A_i z = b_i$  because every point in  $P_E$  must satisfy all constraints in  $P_B$ . By definition, an equality set contains all constraints that are satisfied with equality at all points in the polytope and therefore  $i$  must be in the equality set  $E$ . It follows directly that  $E \supset B$ .

□

Theorems 3.13 and 3.14 allow the face lattice to be written in terms of equality sets. The example of the cube lattice is shown again in Figure 3.4, but now in terms of equality sets.

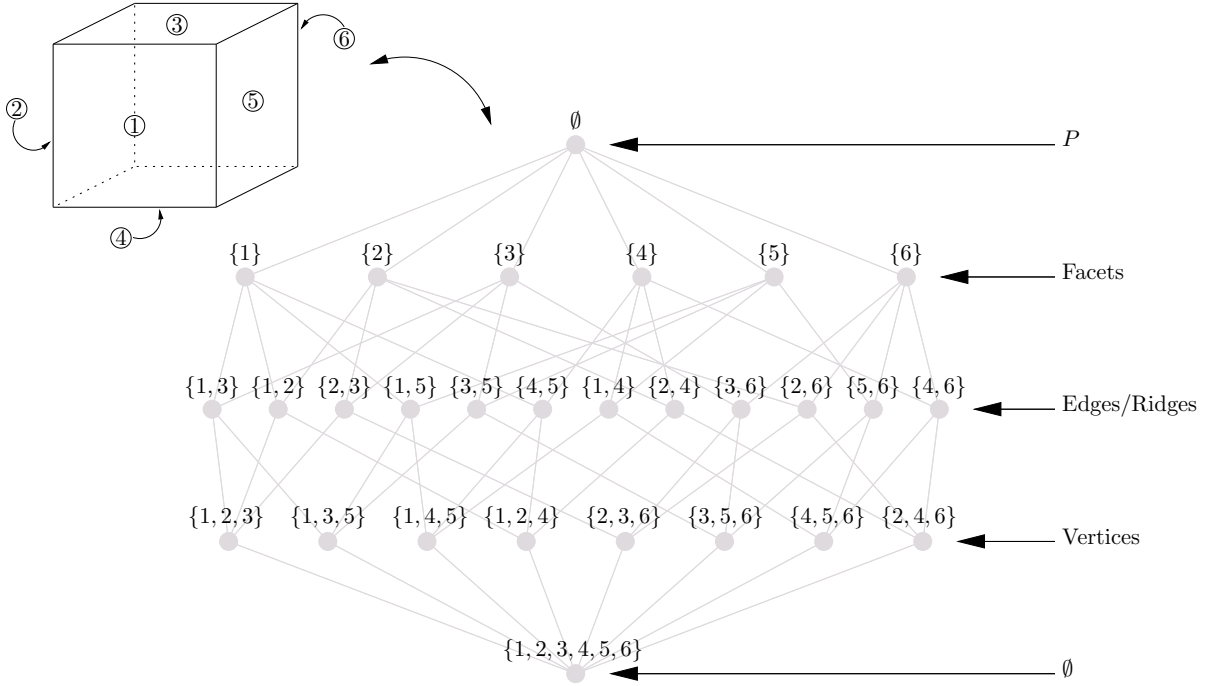


Figure 3.4: Equality Set Lattice of a Cube

### 3.4 Projection of a Polytope

We now turn our attention to the main topic of this part: projection.

**Definition 3.15.** *If  $P \subset \mathbb{R}^d \times \mathbb{R}^k$  is a polytope then the projection of  $P$  onto  $\mathbb{R}^d$  is  $\pi_x P \triangleq \{x \in \mathbb{R}^d \mid \exists y \in \mathbb{R}^k, (x, y) \in P\}$ .*

Note that this definition of projection is also referred to as ‘projection along the coordinate axes’ or ‘orthogonal projection’ in the literature.

The inputs to the algorithm presented in this report are the matrices  $C \in \mathbb{R}^{q \times d}$  and  $D \in \mathbb{R}^{q \times k}$  and the vector  $b \in \mathbb{R}^q$ , which define the polytope  $P \triangleq \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^k \mid Cx + Dy \leq b\}$ . The goal is to compute a matrix  $G \in \mathbb{R}^{p \times d}$  and a vector  $g \in \mathbb{R}^p$  that define the projection of  $P$  onto  $\mathbb{R}^d$ ,  $\pi_x P = \{x \in \mathbb{R}^d \mid Gx \leq g\}$ . Lemma 3.16 states that the projection of a polytope is a polytope, and hence can be expressed in the required form, as the intersection of a finite number of halfspaces.

**Lemma 3.16.** *[Zie95] If  $P \subset \mathbb{R}^d \times \mathbb{R}^k$  is a polytope, then the projection of  $P$  onto  $\mathbb{R}^d$  is a polytope.*

Lemma 3.17 allows the calculation of the dimension of a face of  $\pi_x P$ , given its defining equality set.

### 3.4 PROJECTION OF A POLYTOPE

---

**Lemma 3.17.** [BO98] *If  $E$  is an equality set of the polytope*

$P = \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^k \mid Cx + Dy \leq b\}$ , *then*

$$\begin{aligned} \dim \pi_x P_E &= \dim P_E - k + \text{rank } D_E \\ &= d + \text{rank } D_E - \text{rank} \begin{bmatrix} C_E & D_E \end{bmatrix} \end{aligned}$$

Theorem 3.18 implies that if the affine hulls of each of the facets of the projection can be found, and expressed in the form of linear equations, then we can directly find an appropriate matrix  $G$  and a vector  $g$  such that  $\pi_x P = \{x \in \mathbb{R}^d \mid Gx \leq g\}$ .

**Theorem 3.18.** [Web94, Theorem 3.2.1] *If  $F_1, \dots, F_t$  are the facets of a polytope  $P \in \mathbb{R}^n$ ,  $0 \in P$  and  $\text{aff } F_i = \{x \in \mathbb{R}^n \mid a_i^T x = b_i\}$ , where each  $a_i \in \mathbb{R}^n, b_i \in \mathbb{R}$ , then*

$$P = \text{aff } P \cap \left\{ x \in \mathbb{R}^n \mid \begin{bmatrix} a_1^T \\ \vdots \\ a_t^T \end{bmatrix} x \leq \begin{bmatrix} b_1 \\ \vdots \\ b_t \end{bmatrix} \right\},$$

where the sign of  $b_i$  is chosen so that  $b_i \geq 0$  for all  $i$ .

We now develop the theory required to compute the affine hulls of the facets of the projection as a function of the polytope  $P$ . Lemma 3.19 and the related Corollary 3.20, show that the faces of the projection  $\pi_x P$  are projections of faces of the polytope  $P$ , which can be expressed in terms of the equality sets of  $P$ .

**Lemma 3.19.** [Zie95, Lem. 7.10] *If  $P \subset \mathbb{R}^d \times \mathbb{R}^k$  is a polytope, then for every face  $F$  of  $\pi_x P$ , the preimage  $\pi_x^{-1} F = \{y \in P \mid \pi_x y \in F\}$  is a face of  $P$ .*

*Furthermore, if  $F$  and  $G$  are faces of  $\pi_x P$ , then  $F \subset G$  holds if and only if  $\pi_x^{-1} F \subset \pi_x^{-1} G$ .*

**Corollary 3.20.** *If  $P \subset \mathbb{R}^d \times \mathbb{R}^k$  is a polytope defined by the intersection of  $q$  halfspaces, then for every face  $F$  of  $\pi_x P$ , there exists a unique equality set  $E \subseteq \{1, \dots, q\}$  of  $P$  such that the preimage  $\pi_x^{-1} F = P_E$ .*

*Furthermore, if  $F$  and  $G$  are faces of  $\pi_x P$  such that  $\pi_x^{-1} F = P_E$  and  $\pi_x^{-1} G = P_B$ , where  $E$  and  $B$  are equality sets of  $P$ , then  $F \subset G$  holds if and only if  $E \supset B$ .*

*Proof.* Theorem 3.13 states that every face can be expressed as  $P_E$  for a unique equality set and therefore the first result follows from Proposition 3.19. The second result follows directly from Theorem 3.14 and Lemma 3.19.  $\square$

Lemma 3.21 shows that the projection of an affine hull is the affine hull of the projection.

### 3. POLYTOPIC PROJECTIONS AND EQUALITY SETS

---

**Lemma 3.21.** *If  $P \subset \mathbb{R}^d \times \mathbb{R}^k$  is a polytope then  $\text{aff } \pi_x P = \pi_x \text{aff } P$ .*

*Proof.* We first show the inclusion  $\text{aff } \pi_x P \subseteq \pi_x \text{aff } P$ . If  $\dim \pi_x P = n$ , then  $\text{aff } \pi_x P = \text{aff } \{x_1, \dots, x_{n+1}\}$ , for some  $n + 1$  affinely independent points,  $x_i \in \pi_x P$ . Since  $\pi_x P$  is the projection of  $P$ , for each  $x_i$ , there exists a  $y_i \in \mathbb{R}^k$  such that  $(x_i, y_i) \in P$ , and therefore  $x_i \in \{x \mid \exists y, (x, y) \in \text{aff } P\} = \pi_x \text{aff } P$ . It follows directly that the affine hull of  $\{x_1, \dots, x_{n+1}\}$  is a subset of the projection of the affine hull of  $P$ ,

$$\text{aff } \pi_x P \subseteq \pi_x \text{aff } P.$$

We now show the inclusion  $\text{aff } \pi_x P \supseteq \pi_x \text{aff } P$ . If  $\dim P = m$ , then  $\text{aff } P = \text{aff } \{(\xi_1, v_1), \dots, (\xi_{m+1}, v_{m+1})\}$ , for some  $m + 1$  affinely independent points  $(\xi_i, v_i) \in P$ . Hence,

$$\begin{aligned} \pi_x \text{aff } P &= \pi_x \text{aff } \{(\xi_1, v_1), \dots, (\xi_{m+1}, v_{m+1})\} \\ &= \text{aff } \left\{ x \mid \exists y, \begin{bmatrix} x \\ y \end{bmatrix} = \sum_{i=1}^{m+1} \lambda_i \begin{bmatrix} \xi_i \\ v_i \end{bmatrix}, \sum_{i=1}^{m+1} \lambda_i = 1 \right\} \\ &= \text{aff } \left\{ x \mid x = \sum_{i=1}^{m+1} \lambda_i \xi_i, \sum_{i=1}^{m+1} \lambda_i = 1, \exists y, y = \sum_{i=1}^{m+1} \lambda_i v_i \right\} \\ &= \text{aff } \{\xi_1, \dots, \xi_{m+1}\} \\ &\subseteq \text{aff } \pi_x P, \end{aligned}$$

where the inclusion follows because  $\xi_i \in \pi_x P$  for all  $i$ . □

Recall from Lemma 3.11 that equality sets define affine hulls, i.e.,  $\text{aff } P_{\mathbf{E}} = \{(x, y) \mid C_{\mathbf{E}}x + D_{\mathbf{E}}y = b_{\mathbf{E}}\}$ , if  $P \triangleq \{(x, y) \mid Cx + Dy \leq b\}$  and  $\mathbf{E}$  is an equality set. If  $\mathbf{E}$  also has the property that  $\pi_x P_{\mathbf{E}}$  is a facet of  $\pi_x P$ , then Lemma 3.21 allows the affine hull of the facet to be written as

$$\begin{aligned} \text{aff } \pi_x P_{\mathbf{E}} &= \pi_x \text{aff } P_{\mathbf{E}} \\ &= \pi_x \text{aff } \{(x, y) \mid C_{\mathbf{E}}x + D_{\mathbf{E}}y = b_{\mathbf{E}}\} \end{aligned}$$

Lemma 3.22 then provides a row of the matrix  $G$  and an element of the vector  $g$ :

$$\text{aff } \pi_x P_{\mathbf{E}} = \left\{ x \mid \mathbf{N}(D_{\mathbf{E}}^T)^T C_{\mathbf{E}}x = \mathbf{N}(D_{\mathbf{E}}^T)^T b_{\mathbf{E}} \right\}.$$

### 3.4 PROJECTION OF A POLYTOPE

---

**Lemma 3.22.** *If  $M \triangleq \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^k \mid C_{\mathbb{E}}x + D_{\mathbb{E}}y = b_{\mathbb{E}}\}$  is an affine set, then the projection of  $M$  onto  $\mathbb{R}^d$  is*

$$\pi_x M = \left\{ x \in \mathbb{R}^d \mid \mathbf{N}(D_{\mathbb{E}}^T)^T C_{\mathbb{E}}x = \mathbf{N}(D_{\mathbb{E}}^T)^T b_{\mathbb{E}} \right\}.$$

*Proof.* This statement can be derived directly as follows:

$$\begin{aligned} \pi_x M &= \left\{ x \in \mathbb{R}^d \mid \exists y \in \mathbb{R}^k, C_{\mathbb{E}}x + D_{\mathbb{E}}y = b_{\mathbb{E}} \right\} \\ &= \left\{ x \in \mathbb{R}^d \mid C_{\mathbb{E}}x - b_{\mathbb{E}} \in \text{range}(D_{\mathbb{E}}) \right\} \\ &= \left\{ x \in \mathbb{R}^d \mid \mathbf{N}(D_{\mathbb{E}}^T)^T (C_{\mathbb{E}}x - b_{\mathbb{E}}) = 0 \right\} \\ &= \left\{ x \in \mathbb{R}^d \mid \mathbf{N}(D_{\mathbb{E}}^T)^T C_{\mathbb{E}}x = \mathbf{N}(D_{\mathbb{E}}^T)^T b_{\mathbb{E}} \right\}, \end{aligned} \tag{3.7}$$

where (3.7) follows because the column space of  $D_{\mathbb{E}}$  is orthogonal to its left nullspace. □

The tools are now all in place to compute a matrix  $G$  and a vector  $g$  such that  $\pi_x P = \{x \in \mathbb{R}^d \mid Gx \leq g\}$ , if all of the equality sets of the polytope  $P$ , which define faces  $P_{\mathbb{E}}$  that project to facets of  $\pi_x P$ , can be found. The algorithm that is presented in the remainder of this part is an enumeration method that will find precisely these sets.





# Equality Set Projection

## 4.1 Algorithm Outline

The Equality Set Projection (ESP) algorithm computes the projection of polytopes that are described in halfspace form and also expresses the projection in the same form. It is an output sensitive algorithm, with a constant number of linear programs required per facet of the projection in the absence of degeneracy, although, like most geometric algorithms, it is worst-case exponential in its presence. It is therefore most suited for low facet-count, high vertex-count polytopes. This section will outline the basic procedure for computing the projection, while those following will present the details.

The input to the algorithm is the (bounded) polytope  $P$ , which is described by the intersection of  $q$  halfspaces. The data given to the algorithm are the matrices  $C \in \mathbb{R}^{q \times d}$ ,  $D \in \mathbb{R}^{q \times k}$  and  $b \in \mathbb{R}^q$  that describe  $P$  as follows:

$$P \triangleq \left\{ (x, y) \in \mathbb{R}^d \times \mathbb{R}^k \mid Cx + Dy \leq b \right\}.$$

The assumption is not made that  $P$  is irredundant, although it generally reduces computation time to remove redundancies before beginning. The goal is to compute the matrix  $G$  and the vector  $g$  such that the axis-aligned projection of  $P$  onto the first  $d$  coordinates is given by the irredundant description

$$\begin{aligned} \pi_x P &\triangleq \left\{ x \in \mathbb{R}^d \mid \exists y, (x, y) \in P \right\} \\ &= \left\{ x \in \mathbb{R}^d \mid Gx \leq g \right\}. \end{aligned}$$

The assumption is made that  $\pi_x P$  is full-dimensional in  $\mathbb{R}^d$  and that the origin is in its

interior. Note that this is guaranteed to be true if  $P$  is full-dimensional in  $\mathbb{R}^d \times \mathbb{R}^k$  and the origin is interior to it. The remainder of this section discusses the algorithm under these assumptions and Sections 5.3 and 5.4 show how to apply the algorithm when a polytope does not satisfy these requirements.

As discussed in Chapter 3, the matrix  $G$  and the vector  $g$  are formed from the affine hulls of the facets of the projection. Each facet  $F$  of the projection  $\pi_x P$  is defined by a unique equality set  $E$  of  $P$  such that  $F = \pi_x P_E$ . Therefore, the algorithm is a search procedure for finding these unique equality sets.

The ESP search procedure exploits the relationship between facets and ridges that is captured by the Diamond Property (Theorem 3.7). The algorithm is initialized by discovering a random facet  $F$  of  $\pi_x P$  (Shooting Oracle). The equality sets that define faces of  $P$  that project to the ridges of  $\pi_x P$  that are subsets of  $F$  can then be computed (Ridge Oracle).

A list is maintained that has one element for every ridge discovered. This element consists of two equality sets of  $P$  that define a ridge of the projection and one of its containing facets. In each iteration, a ridge is selected from the list, along with its containing facet. Recall that by the Diamond Property, each ridge is contained in exactly two facets. Therefore, given the selected ridge-facet list element, the equality set of the second containing facet can be computed (Adjacency Oracle). As before, the equality sets of all ridges of this new facet are computed and inserted into the list.

Note that because of the Diamond Property, each ridge is visited exactly twice. Therefore, by removing every ridge that appears in the list twice at the end of each iteration, we can guarantee that the algorithm has no cycles and has a finite execution time. The iterative step is repeated until the list is empty, at which time all facets will have been found.

Once all of the equality sets that define faces of  $P$  that project to facets of  $\pi_x P$  have been collected, the matrix  $G$  and the vector  $g$  can be computed as shown in Propositions 3.18 and 3.22. Theorem 4.1 proves that this procedure finds all facets of the projection and that it terminates in finite time. It will be seen in the following sections that this statement can be strengthened to polynomial time in the absence of degeneracy.

**Theorem 4.1.** *The ESP algorithm returns all facets of the projection and terminates in finite time.*

*Proof.* Beginning from a random facet  $F$  of  $\pi_x P$ , the ESP algorithm iteratively computes all adjacent facets without re-visiting any facet. Every polytope has a finite number of faces [Roc70, Thm. 19.1] and therefore ESP terminates in finite time.

It remains to be shown that all facets are visited. Given a facet  $F_0$ , we define the set  $\mathcal{F}(F_0)$  to be all facets  $F_n$  such that there exists a sequence of facets from  $F_0$  to  $F_n$  and  $F_i$

## 4.2 ADJACENCY ORACLE

---

is adjacent to  $F_{i+1}$ ,  $i = 0, \dots, n - 1$ . Given an initial starting facet  $F_0$ , the ESP algorithm clearly computes  $\mathcal{F}(F_0)$ . Therefore, if  $\mathcal{F}(F_0)$  covers all facets for any choice of  $F_0$ , then the ESP algorithm will return all facets of the projection.

We define a graph  $H(\mathcal{F}(F_0))$  whose vertices represent the facets of  $\mathcal{F}(F_0)$  and two vertices of the graph are connected by an edge if the corresponding facets are adjacent. [Zie95, Cor. 2.14] states that there exists a polytope  $\tilde{P}$  such that  $H(\mathcal{F}(F_0))$  is combinatorially equivalent to the graph  $G(\tilde{P})$  formed from the vertices and edges of  $\tilde{P}$ . Balinski's Theorem [Bal61] states that  $G(\tilde{P})$  is connected and therefore it follows that  $H(\mathcal{F}(F_0))$  is connected and the ESP algorithm returns all facets of the projection. □

Three oracles are needed for this search procedure:

**Adjacency Oracle**  $(E^{\text{adj}}, a_{\text{adj}}, b_{\text{adj}}) = \text{ADJ}((E_r, a_r, b_r), (E, a_f, b_f))$

This oracle takes two equality sets  $E_r$  and  $E$  of  $P$  where  $\pi_x P_E = \{x \mid a_f^T x = b_f\} \cap \pi_x P$  is a facet of  $\pi_x P$  and  $\pi_x P_{E_r} = \{x \mid a_r^T x = b_r\} \cap \pi_x P$  is a ridge with  $\pi_x P_E \supset \pi_x P_{E_r}$ . The oracle returns the unique equality set  $E^{\text{adj}}$  such that  $\pi_x P_{E^{\text{adj}}} \supset \pi_x P_{E_r}$ ,  $E^{\text{adj}} \neq E$ .

**Ridge Oracle**  $(E_r^1, \dots, E_r^m) = \text{RDG}(E, a_f, b_f)$

This oracle takes an equality set  $E$  of  $P$  where  $\pi_x P_E = \{x \mid a_f^T x = b_f\} \cap \pi_x P$  is a facet of  $\pi_x P$  and returns all equality sets  $E_r^i$  of  $P$  such that  $\pi_x P_{E_r^i}$  is a ridge of  $\pi_x P$  and  $\pi_x P_{E_r^i} \subset \pi_x P_E$ .

**Ray-shooting Oracle**  $(E_0, a_f, b_f) = \text{SHOOT}(P)$

This oracle returns a random equality set  $E_0$  of  $P$  such that  $\pi_x P_{E_0} = \{x \mid a_f^T x = b_f\} \cap \pi_x P$  is a facet of the projection.

These oracles are discussed in Sections 4.2, 4.3 and 4.4 respectively. ESP is described in procedural form in Algorithm 4.1.

The algorithm is illustrated by the example of a cube in  $\mathbb{R}^3$  being projected to  $\mathbb{R}^2$  as shown in Figure 4.1. The sequence shown in Figure 4.2 demonstrates the construction of the top two levels of the lattice of the projection as the algorithm proceeds. The path that the algorithm takes through the 3D cube lattice is shown in Figure 4.3.

ESP Procedure: Projection of a Cube

**Step 1** corresponds to the beginning of the **while** loop in Algorithm 4.1

## 4.2 Adjacency Oracle

The goal of this section is to build the adjacency oracle

$(E^{\text{adj}}, a_{\text{adj}}, b_{\text{adj}}) = \text{ADJ}((E_r, a_r, b_r), (E, a_f, b_f))$  introduced in Section 4.1. The oracle takes

---

**Algorithm 4.1** Equality Set Projection (ESP)

---

**Input:** Polytope  $P \triangleq \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^k \mid Cx + Dy \leq b\}$  whose projection is full-dimensional and contains the origin in its interior.

**Output:** Matrix  $G$  and vector  $g$  such that  $\{x \in \mathbb{R}^d \mid Gx \leq g\}$  is an irredundant description of  $\pi_x P$ .

List  $\mathbf{E}$  of all equality sets  $E$  of  $P$  such that  $\pi_x P_E$  is a facet of  $\pi_x P$ .

*Initialize ridge-facet list  $\mathbf{L}$  with random facet.*

- 1:  $\mathbf{L} \leftarrow \emptyset$
- 2:  $(\mathbf{E}_0, a_f, b_f) = \text{SHOOT}(P)$  Section 4.4
- 3:  $\mathbf{E}_r \leftarrow \text{RDG}(\mathbf{E}_0, a_f, b_f)$  Section 4.3
- 4: **for each** element  $(\mathbf{E}_r, a_r, b_r)$  in list  $\mathbf{E}_r$  **do**
- 5:     Add element  $((\mathbf{E}_r, a_r, b_r), (\mathbf{E}_0, a_f, b_f))$  to list  $\mathbf{L}$
- 6: **end for**

*Initialize matrix  $G$ , vector  $g$  and list  $\mathbf{E}$ .*

- 7:  $G \leftarrow a_f^T, g \leftarrow b_f$
- 8:  $\mathbf{E} \leftarrow \mathbf{E}_0$

*Search for adjacent facets until the list  $\mathbf{L}$  is empty.*

- 9: **while**  $\mathbf{L} \neq \emptyset$  **do**
- 10:     Choose an element  $((\mathbf{E}_r, a_r, b_r), (\mathbf{E}, a_f, b_f))$  from list  $\mathbf{L}$
- 11:      $(\mathbf{E}^{\text{adj}}, a_{\text{adj}}, b_{\text{adj}}) \leftarrow \text{ADJ}((\mathbf{E}_r, a_r, b_r), (\mathbf{E}, a_f, b_f))$  Section 4.2
- 12:      $\mathbf{E}_r \leftarrow \text{RDG}(\mathbf{E}^{\text{adj}}, a_{\text{adj}}, b_{\text{adj}})$  Section 4.3
- 13:     **for each** element  $(\mathbf{E}_r, a_r, b_r)$  in list  $\mathbf{E}_r$  **do**
- 14:         **if** there exists an element  $((\mathbf{A}_r, a_1, b_1), (\mathbf{A}, a_2, b_2))$  in list  $\mathbf{L}$  such that  $\mathbf{A}_r = \mathbf{E}_r$  **then**
- 15:             Remove element  $((\mathbf{A}_r, a_1, b_1), (\mathbf{A}, a_2, b_2))$  from list  $\mathbf{L}$
- 16:         **else**
- 17:             Add element  $((\mathbf{E}_r, a_r, b_r), (\mathbf{E}^{\text{adj}}, a_{\text{adj}}, b_{\text{adj}}))$  to list  $\mathbf{L}$
- 18:         **end if**
- 19:     **end for**
- 20:      $G \leftarrow \begin{bmatrix} G \\ a_{\text{adj}}^T \end{bmatrix}, g \leftarrow \begin{bmatrix} g \\ b_{\text{adj}} \end{bmatrix}$
- 21:      $\mathbf{E} \leftarrow (\mathbf{E}, \mathbf{E}^{\text{adj}})$
- 22: **end while**

*Report projection.*

- 23: Report  $G, g$  and  $\mathbf{E}$
-

## 4.2 ADJACENCY ORACLE

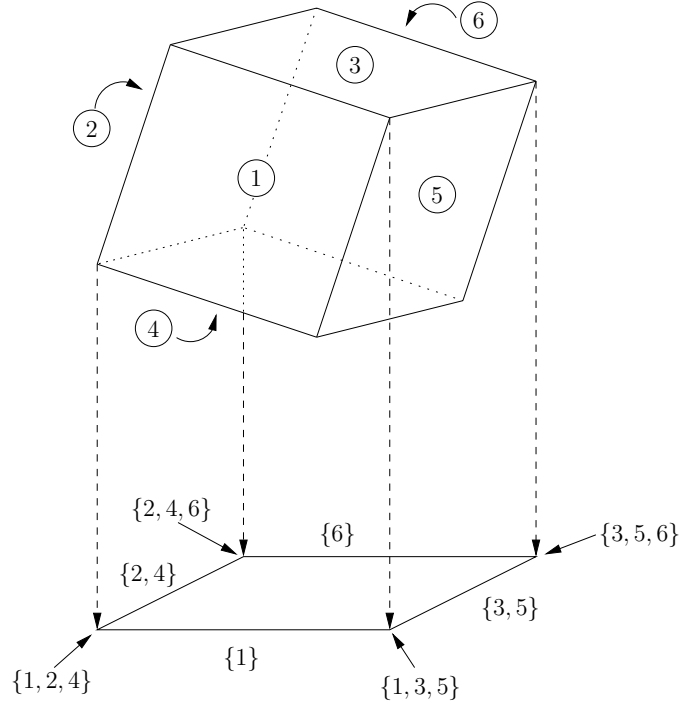


Figure 4.1: Example Projection of a Cube

two equality sets  $E$  and  $E_r \supset E$  that define a facet  $\pi_x P_E$  and a ridge  $\pi_x P_{E_r}$  of the projection  $\pi_x P$ . The unit normals  $a_r$  and  $a_f$  are orthogonal and have the property that:

$$\begin{aligned}\pi_x P_E &= \{x \mid a_f^T x = b_f\} \cap \pi_x P, \\ \pi_x P_{E_r} &= \{x \mid a_r^T x = b_r\} \cap \pi_x P_E.\end{aligned}$$

The goal is to compute an equality set  $E^{\text{adj}} \subset E_r$  such that  $\pi_x P_{E^{\text{adj}}}$  is the facet of the projection that is adjacent to the given facet  $\pi_x P_E$  and that contains the given ridge  $\pi_x P_{E_r}$ . Note that this facet is guaranteed to exist and to be unique by the Diamond Property (Theorem 3.7).

In this section we will show that the equality set of the adjacent facet is given by  $E^{\text{adj}} \triangleq \{i \in E_r \mid C_i x^* + D_i y^* = b_i\}$ , where  $(x^*, y^*)$  is the optimum of the LP

$$\begin{aligned}(x^*, y^*) &= \underset{x, y}{\operatorname{argmax}} && a_r^T x \\ &\text{subject to} && C_{E_r} x + D_{E_r} y \leq b_{E_r} \\ &&& a_f^T x = b_f(1 - \delta),\end{aligned}$$

where  $\delta$  is a positive number. The remainder of this section is dedicated to proving this claim.

We will first show that as the adjacent facet must contain the given ridge, its affine hull

## 4. EQUALITY SET PROJECTION

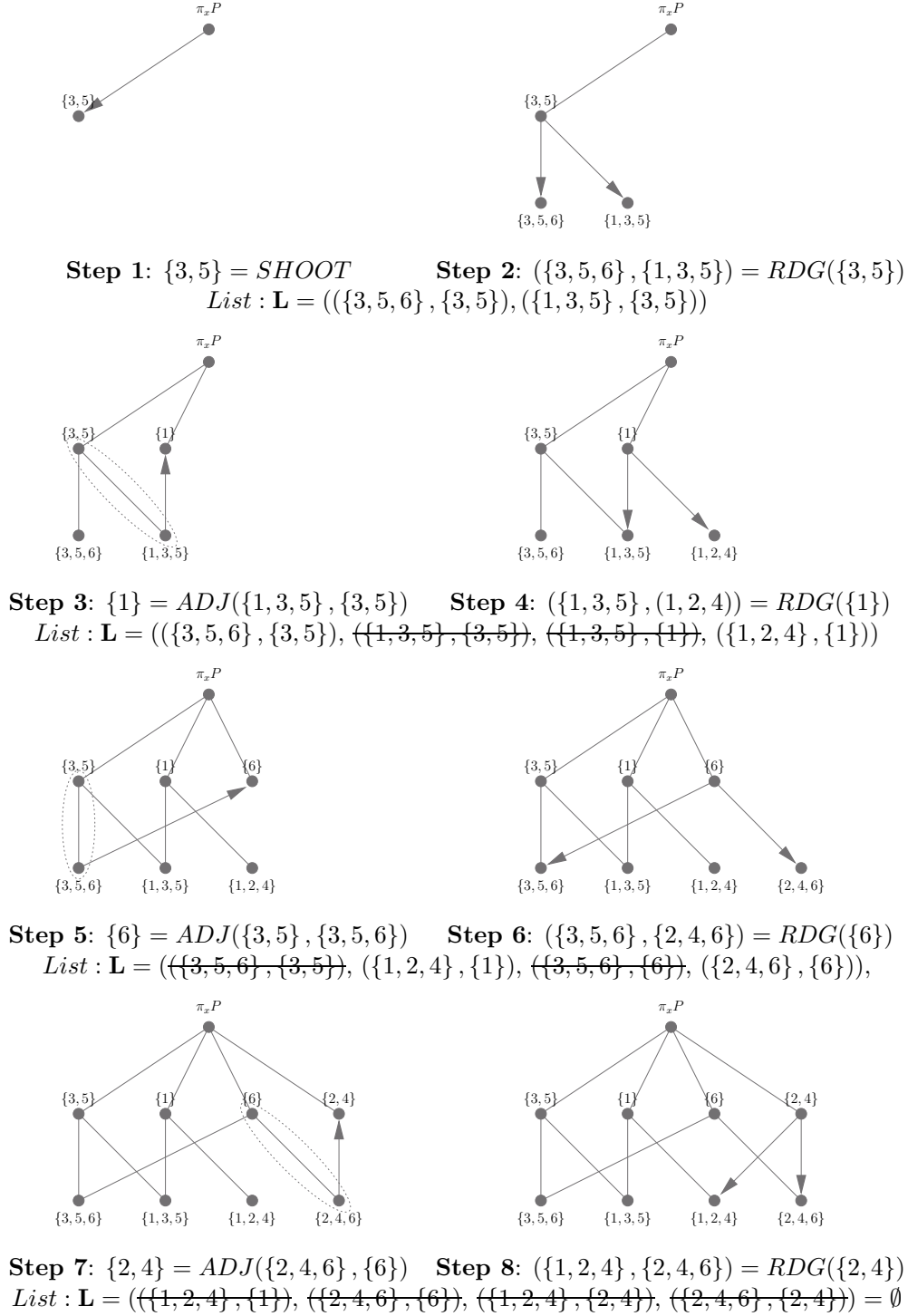


Figure 4.2: ESP Procedure: Projection of a Cube

**Step 1:** corresponds to the beginning of the **while** loop in Algorithm 4.1

## 4.2 ADJACENCY ORACLE

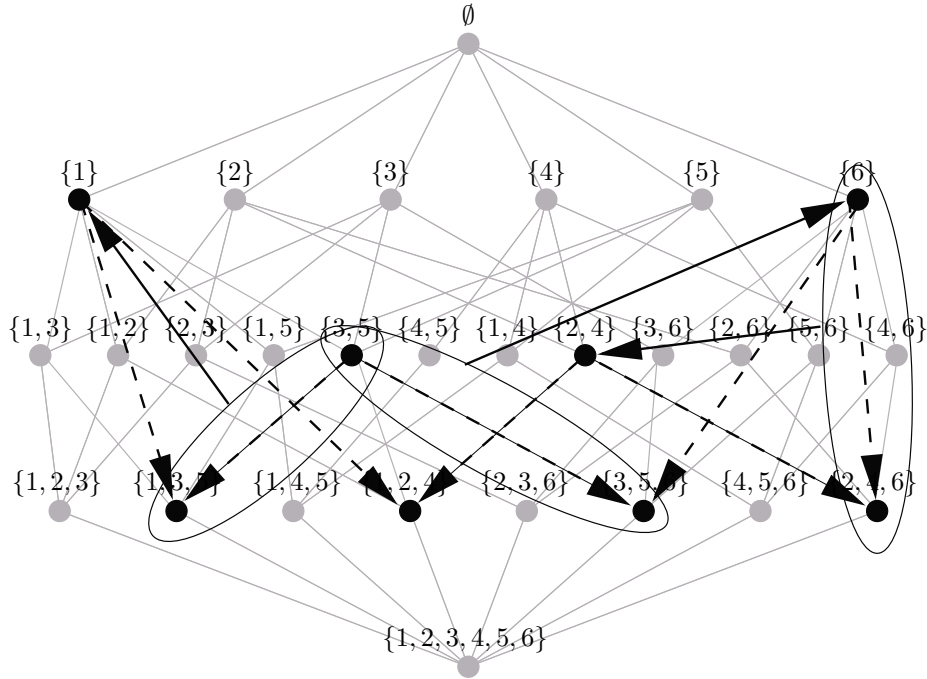


Figure 4.3: Example Projection of a Cube: Search Path in Cube Face Lattice

can differ in only one degree of freedom from that of the given facet. We will introduce a linear mapping that will allow us to formulate the search over this degree of freedom as a linear program in order to determine a point on the affine hull of the adjacent facet. From this point we will then compute the equality set of the adjacent facet.

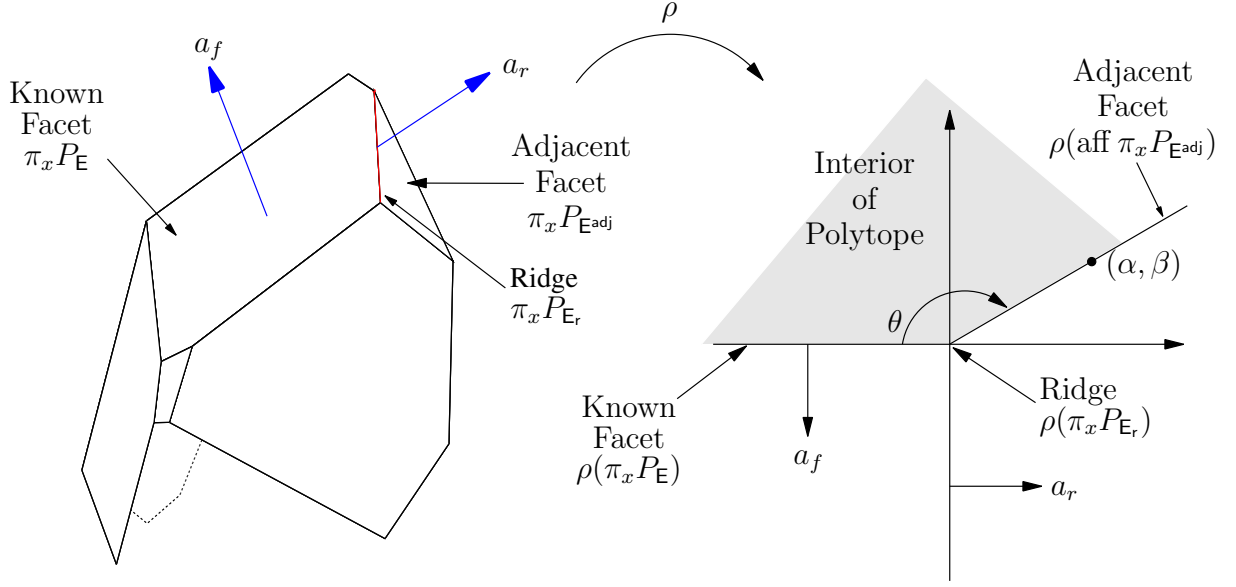
Similarly to [FLL00], we define an affine transformation  $\rho$  that maps  $\mathbb{R}^d$  to  $\mathbb{R}^2$ :

$$\rho : \mathbb{R}^d \longrightarrow \mathbb{R}^2, x \longmapsto \begin{bmatrix} a_r^T \\ -a_f^T \end{bmatrix} (x - x_0), \quad (4.1)$$

where  $x_0$  is any point in the affine hull of the given ridge  $\text{aff } \pi_x P_{E_r}$ .

The mapping  $\rho$  takes every point in the polytope  $\pi_x P$  to a two dimensional space such that all points in the affine hull of the facet  $\pi_x P_E$  are mapped to the first axis and all points in the affine hull of the ridge  $\text{aff } \pi_x P_{E_r}$  are mapped to the origin. The map is depicted in Figure 4.4 for a polytope that is projected to  $\mathbb{R}^3$ . For convenience, we will refer to the first axis as  $\alpha$  and the second as  $\beta$ .

**Remark 4.2.** *It is assumed that the normal  $a_f$  is given such that the ray  $\lambda a_f$  will intersect the affine hull of  $\pi_x P_E$  for a positive value of  $\lambda$  and  $a_r$  is outward facing from the facet  $\pi_x P_E$ . These requirements are satisfied if  $a_f$  and  $a_r$  are obtained from the ESP algorithm.*


 Figure 4.4: Example of the Wrapping Map  $\rho$ 

Under these assumptions, the facet  $\pi_x P_E$  is mapped to the negative  $\alpha$ -axis and the interior of the polytope is strictly above the  $\alpha$ -axis as shown in Figure 4.4. Note that because the polytope  $\pi_x P$  is convex, the angle that the adjacent facet makes with  $\pi_x P_E$  must be less than  $180^\circ$ , or equivalently all points  $(\alpha, \beta) \in \rho(\pi_x P_{E^{\text{adj}}})$  must have  $\beta \geq 0$ .

We will now show that the affine hull of the adjacent facet  $\text{aff } \pi_x P_{E^{\text{adj}}}$  will form a line going through the origin under the mapping  $\rho$  and furthermore, all points in the polytope are mapped to the left of this line. First, the following lemma is needed.

**Lemma 4.3.** *If  $R$  is a ridge,  $F \supset R$  is a facet and  $\text{aff } R = \left\{ x \mid \begin{bmatrix} a_f & a_r \end{bmatrix}^T (x - x_0) = 0 \right\}$ , then there exists a  $\gamma \in \mathbb{R}$  such that  $\text{aff } F = \left\{ x \mid \begin{bmatrix} 1 & \gamma \end{bmatrix} \begin{bmatrix} a_r^T \\ -a_f^T \end{bmatrix} (x - x_0) = 0 \right\}$ .*

*Proof.*  $F$  is a facet and therefore only one equality is needed to describe its affine hull:

$$\text{aff } F = \{x \mid a^T(x - x_0) = 0\}, \quad \text{for some } a \in \mathbb{R}^d.$$

The ridge  $R$  is a subset of  $F$  and therefore for every  $x$  in the affine hull of  $R$ ,  $x$  must be in the affine hull of  $F$ :

$$\begin{bmatrix} a_f & a_r \end{bmatrix}^T (x - x_0) = 0 \Rightarrow a^T(x - x_0) = 0.$$



## 4.2 ADJACENCY ORACLE

---

This is equivalent to  $x$  being in the affine hull of the facet if  $x - x_0$  is in the left-nullspace of  $\begin{bmatrix} a_f & a_r \end{bmatrix}$ :

$$x - x_0 \in \mathbf{N}\left(\begin{bmatrix} a_f & a_r \end{bmatrix}^T\right) \Rightarrow a^T(x - x_0) = 0.$$

Finally, we can see that  $a$  must be perpendicular to the left-nullspace of  $\begin{bmatrix} a_f & a_r \end{bmatrix}$ , or equivalently, in its rowspace:

$$a = a_r - \gamma a_f.$$

□

From Lemma 4.3 and (4.1) we can see that

$$\begin{aligned} \rho(\text{aff } F) &= \left\{ \rho(x) \mid \begin{bmatrix} 1 & \gamma \end{bmatrix} \rho(x) = 0 \right\} \\ &= \{(\alpha, \beta) \mid \alpha + \gamma\beta = 0\} \end{aligned}$$

and therefore under the mapping  $\rho$ , all points on the affine hull of the adjacent facet will be mapped to a line through the origin. The goal is now to determine which equality set  $E^{\text{adj}} \subset E_r$  defines this line.

Each subset  $B$  of  $E_r$  defines a polytope  $\pi_x P_B$  that, by Corollary 3.20, is a superset of the given ridge  $\pi_x P_B \supseteq \pi_x P_{E_r}$  and correspondingly, the affine hull contains the affine hull of the given ridge  $\text{aff } \pi_x P_B \supseteq \text{aff } \pi_x P_{E_r}$ . Therefore, under  $\rho$ , the affine hull of  $\pi_x P_B$  will either map to the origin, or to a line through the origin. Each of these lines forms an angle with the negative  $\alpha$ -axis as shown in Figure 4.5. Since the portion of each of these lines that lies above the  $\alpha$ -axis must be internal to the projection of the polytope, the largest angle  $\theta$  made with the negative  $\alpha$ -axis must define the adjacent facet.

**Remark 4.4.** *The simplest approach to finding the adjacent facet would be to compute the projection  $\pi_x \text{aff } P_B$  for each  $B \subset E_r$  and then calculate the angle that it makes with  $\pi_x \text{aff } P_{E_r}$ . While this would be linear for a non-degenerate facet, it becomes combinatorial in the case of degeneracy and so the LP approach presented here is preferred.*

**Remark 4.5.** *The adjacent facet can now be described in terms of the angle  $\theta \in (0, \pi)$  between the adjacent facet and the given facet  $\pi_x P_{E_r}$  or by a point  $(\alpha, \beta) \in \mathbb{R} \times \mathbb{R}_{>0}$  on the adjacent facet under the mapping  $\rho$ . Note that the affine mapping  $\rho$  is conformal and therefore the angle  $\theta$  defined under the mapping  $\rho$  is the true angle between the two facets.*

The search for the largest angle is formulated as a maximisation over  $\alpha$  while fixing  $\beta$  to be a positive value; for convenience, we here choose  $\beta = \delta b_f$ , where  $\delta$  is a small positive number.

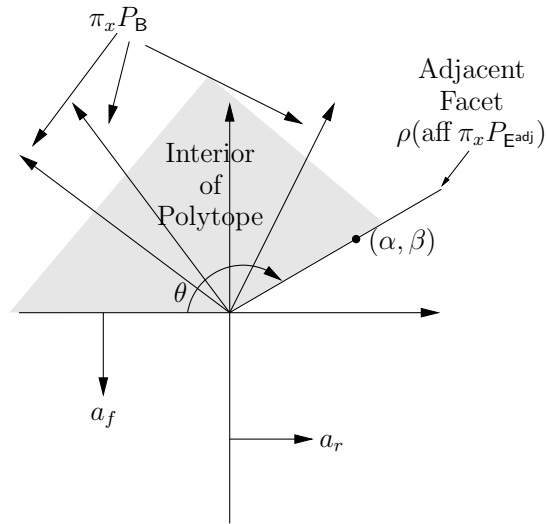


Figure 4.5: Subsets of  $E_r$  under the Wrapping Map  $\rho$

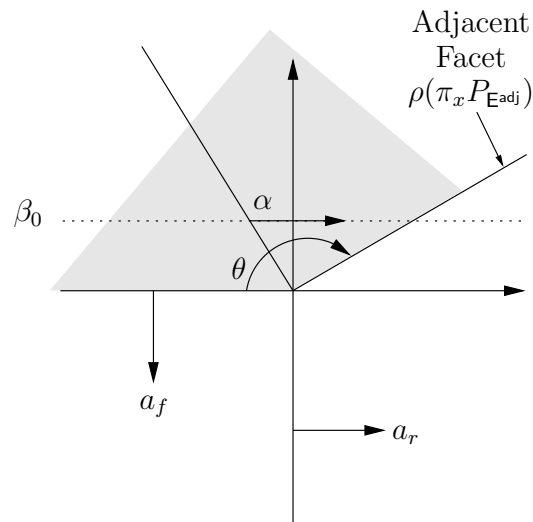


Figure 4.6: Linear Optimisation Under the Wrapping Map  $\rho$

## 4.2 ADJACENCY ORACLE

---

From Figure 4.6 this is clearly equivalent to maximizing  $\theta$ . In the following linear program we search for a point  $(x, y)$  in the polytope  $P$  and define the projection of this point (which is just  $x$ ) under the mapping  $\rho$  as  $(\alpha, \beta) \triangleq \rho(x)$ . We can then maximize  $\alpha$  and constrain  $\beta$  to be  $\delta b_f$ . The appropriate maximization is:

$$\begin{aligned} & \underset{x,y}{\text{maximise}} && \alpha \\ & \text{subject to} && (x, y) \in P \\ & && \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \rho(x) \\ & && \beta = \delta b_f. \end{aligned} \tag{4.2}$$

From the definition of the mapping  $\rho$  (4.1) and recalling that  $a_f^T x_0 = b_f$  we can re-write LP (4.2) as follows:

$$\begin{aligned} J^* = & \underset{x,y}{\text{maximise}} && a_r^T x \\ & \text{subject to} && C_{E_r} x + D_{E_r} y \leq b_{E_r} \\ & && a_f^T x = b_f(1 - \delta). \end{aligned} \tag{4.3}$$

**Remark 4.6.** Notice that in LP (4.3) only the constraints  $E_r$  are used to represent  $(x, y) \in P$ . From Corollary 3.20 we know that the equality set of the adjacent facet must be a subset of  $E_r$  and therefore for efficiency we include only these constraints. As all constraints that are not active on the ridge have been removed from the system, the inequality system in LP (4.3) becomes an unbounded polyhedron and therefore any positive value of  $\delta$  is acceptable.

We now show how to compute the equality set of the adjacent facet from the optimizer of LP (4.3). First, the following lemma is needed.

**Lemma 4.7.** *If  $(x^*, y^*)$  is an optimizer of LP (4.3) and  $E^* = \{i \in E_r \mid C_i x^* + D_i y^* = b_i\}$ , then  $E^*$  is an equality set of  $P$ ,  $\pi_x P_{E^*} \supset \pi_x P_{E_r}$  and  $\dim \pi_x P_{E^*} = d - 1$ .*

*Proof.*  $E^*$  is clearly an equality set by Definition 3.9.

We first show that  $E^* \subset E_r$ . Since LP (4.3) contains only the constraints  $E_r$ , we have  $E^* \subseteq E_r$ . To show proper inclusion we recall that the ridge  $\pi_x P_{E_r}$  is mapped to the origin under the map  $\rho$ , however the constraint  $\beta > 0$  implies that  $(x^*, y^*)$  is not zero and therefore not on the affine hull of  $P_{E_r}$ . It follows that  $E_r \neq E^*$ .

From Theorem 3.14,  $E^* \subset E_r$  implies that  $P_{E^*} \supset P_{E_r}$  and therefore  $\pi_x P_{E^*} \supseteq \pi_x P_{E_r}$ . By construction,  $x^* \in \text{aff } \pi_x P_{E^*}$  and  $x^* \notin \pi_x P_{E_r}$  and therefore  $\pi_x P_{E^*} \supset \pi_x P_{E_r}$ .

It follows that  $\dim \pi_x P_{E^*} > \dim \pi_x P_{E_r} = d - 2$ .  $\dim \pi_x P_{E^*}$  is  $d$  if and only if  $E^* = \emptyset$ . From the constraints in LP (4.3), we can see that this is the case only if  $x^* = \infty$ . However, the

polytope  $P$  is bounded and the angle between any two faces is less than  $180^\circ$  by convexity, and therefore  $x^* < \infty$  and  $\dim \pi_x P_{E^*} = d - 1$ .  $\square$

Recall that  $P$  is bounded, and therefore the primal optimizer exists and is finite, although it may be non-unique. If the optimizer of LP (4.3) is unique, then we can compute the equality set of the adjacent facet directly using the following proposition. Note that a primal degenerate solution still provides a unique optimizer.

**Theorem 4.8.** *If  $(x^*, y^*)$  is a unique optimal point of LP (4.3) and  $E^* = \{i \in E_r \mid C_i x^* + D_i y^* = b_i\}$ , then  $E^*$  is an equality set,  $\dim P_{E^*} = d - 1$  and  $\pi_x P_{E^*} \supset \pi_x P_{E_r}$  is a facet of  $\pi_x P$ .*

*Proof.* By construction, the affine hull of  $\pi_x P_{E^*}$  is the affine hull of the adjacent facet. We now have that the adjacent facet is given by  $F = (\text{aff } \pi_x P_{E^*}) \cap \pi_x P$ . Clearly, if  $F = \pi_x P_{E^*}$ , then  $\pi_x P_{E^*}$  is a facet.

By construction, the point  $x^*$  is in the interior of  $F$ . Since the optimal point is unique, there is only one  $y = y^*$  such that  $(x^*, y)$  is in the interior of  $\pi_x^{-1} F$  and therefore the equality set of  $\pi_x^{-1} F$  is given by  $B = \{i \mid C_i x^* + D_i y^* = b_i\} = E^*$ . It follows that  $F = \pi_x P_B = \pi_x P_{E^*}$  and  $\pi_x P_{E^*}$  is a facet of  $\pi_x P$ .  $\square$

The solution is said to be dual-degenerate if there are multiple optimizers. In this situation Theorem 4.8 does not apply and more work is needed to compute the equality set of the adjacent facet. Recall that by construction, the projection of any point  $(x^*, y^*)$  that is an optimizer of LP (4.3) is on the affine hull of the adjacent facet,  $x^* \in \text{aff } \pi_x P_{E^{\text{adj}}}$ . Therefore, given any optimizer  $(x^*, y^*)$  we can compute the affine hull of the adjacent facet  $\text{aff } \pi_x P_{E^{\text{adj}}}$  from Lemma 4.3, which states that it has the form:

$$\begin{aligned} \text{aff } \pi_x P_{E^{\text{adj}}} &= \{x \mid (a_r - \gamma a_f)^T (x - x_0) = 0\} \\ &= \{x \mid (a_r - \gamma a_f)^T x = b_r - \gamma b_f\}, \end{aligned}$$

for some  $\gamma \in \mathbb{R}$ . The optimal point  $x^*$  is on the affine hull of the adjacent facet and therefore we can compute  $\gamma$  from the above equation as:

$$\gamma = \frac{a_r^T x^* - b_r}{a_f^T x^* - b_f}. \quad (4.4)$$

**Remark 4.9.** *Note that most commercial LP solvers will return an arbitrary optimizer  $(x^*, y^*)$  in the case of dual-degeneracy.*

### 4.3 RIDGE ORACLE

---

**Remark 4.10.** *A test for the recognition of dual-degeneracy using an LP is given in [Mur83, Theorem 4.14].*

Finally, the equality set that defines the adjacent facet must be derived from the equation for its affine hull. The face  $P_{\mathbf{E}^{\text{adj}}}$  can be written as the pre-image of  $\pi_x P_{\mathbf{E}^{\text{adj}}}$ :

$$P_{\mathbf{E}^{\text{adj}}} = \pi_x^{-1} \pi_x P_{\mathbf{E}^{\text{adj}}} = P \cap \{x \mid (a_r - \gamma a_f)^T x = b_r - \gamma b_f\}. \quad (4.5)$$

By definition the constraints of  $P$  that are everywhere active in  $P_{\mathbf{E}^{\text{adj}}}$  form the equality set  $\mathbf{E}^{\text{adj}}$ . A method for computing these constraints for a given polytope is given in Section 5.2, which will provide the desired set,  $\mathbf{E}^{\text{adj}}$ .

Algorithm 4.2 summarizes the procedure discussed in this section.

### 4.3 Ridge Oracle

The oracle  $\mathbf{E}_r = \text{RDG}(\mathbf{E}, a_f, b_f)$  introduced in Section 4.1 will be developed here. The oracle takes as input an equality set  $\mathbf{E}$  that defines a facet  $\pi_x P_{\mathbf{E}}$  of the projection  $\pi_x P$ . The unit vector  $a_f$  and the scalar  $b_f$  also define the facet as  $\pi_x P_{\mathbf{E}} = \{x \mid a_f^T x = b_f\} \cap \pi_x P$ . The oracle returns the equality sets of the ridges of the projection  $\pi_x P$  that are subsets of the given facet  $\pi_x P_{\mathbf{E}}$ .

Recall that every face of a polytope is itself a polytope and therefore  $\pi_x P_{\mathbf{E}}$  can be written in the form  $\pi_x P_{\mathbf{E}} = \{x \mid a_f^T x = b_f\} \cap \{x \mid \Upsilon x \leq v\}$  for some  $\Upsilon \in \mathbb{R}^{p \times d}$  and  $v \in \mathbb{R}^p$ . The ridges of the projection that are subsets of the given facet  $\pi_x P_{\mathbf{E}}$  are then given by the facets of the polytope  $\pi_x P_{\mathbf{E}}$ , which can be written as  $\pi_x P_{\mathbf{E}} \cap \{x \mid \Upsilon_i x = v_i\}$ , for  $i = 1, \dots, p$ . In this section we will show that if the dimension of the face  $P_{\mathbf{E}}$  is  $d - 1$ , then such an  $\Upsilon$  and  $v$  can be found directly, otherwise a low-dimensional projection is required.

We begin by computing expressions for the given face of the polytope  $P_{\mathbf{E}}$  and its projection  $\pi_x P_{\mathbf{E}}$ . In Section 4.3.1, we show how to compute the equality sets of the ridges directly if the dimension of the face  $P_{\mathbf{E}}$  is  $d - 1$  and then Section 4.3.2 handles the general case. The general case requires a recursive call to the ESP algorithm that reduces the dimension to which we are projecting at each step. Therefore in order to ensure that this recursion terminates, in Section 4.3.3 we present a method of computing the ridges directly when projecting to 1D.

**Lemma 4.11.** *If  $\mathbf{E}$  is an equality set of  $P$  such that  $\pi_x P_{\mathbf{E}}$  is a facet of  $\pi_x P$  and  $\dim P_{\mathbf{E}} =$*

---

**Algorithm 4.2** Adjacency oracle (ESP)

---

**Input:** Polytope  $P$  and equality sets  $E_r$  and  $E$  such that  $\pi_x P_{E_r}$  is a ridge and  $\pi_x P_E$  is a facet of  $\pi_x P$  and  $E_r \supset E$ .

Orthogonal unit vectors  $a_f$  and  $a_r$  and scalars  $b_f$  and  $b_r$  such that  $\text{aff } \pi_x P_E = \{x \mid a_f^T x = b_f\}$  and  $\text{aff } \pi_x P_{E_r} = \{x \mid a_r^T x = b_r\} \cap \text{aff } \pi_x P_E$ .

**Output:** Equality set  $E^{\text{adj}}$  such that  $\pi_x P_{E^{\text{adj}}}$  is a facet of  $\pi_x P$  and  $\pi_x P_{E_r} \subset \pi_x P_{E^{\text{adj}}}$

*Compute a point on the affine hull of the adjacent facet.*

1: Compute

$$\begin{aligned} (x^*, y^*) = & \arg \max_{x,y} a_r^T x \\ \text{subject to } & C_{E_r} x + D_{E_r} y \leq b_{E_r} \\ & a_f^T x = b_f(1 - \delta) \end{aligned}$$

*Compute the equality set of the adjacent facet.*

2: **if** LP is *not* dual degenerate **then**

3:  $E^{\text{adj}} \leftarrow \{i \in E_r \mid C_i x^* + D_i y^* = b_i\}$

4: **else**

5:  $\gamma \leftarrow \frac{a_r^T x^* - b_r}{a_f^T x^* - b_f}$  (4.4)

6: Compute the equality set  $E^{\text{adj}}$  of Section 5.2

$$\{(x, y) \mid C_{E_r} x + D_{E_r} y \leq b_{E_r}, (a_r - \gamma a_f)^T x = b_r - \gamma b_f\}.$$

7: **end if**

*Compute affine hull of adjacent facet.*

8:  $[a_{\text{adj}}^T \ b_{\text{adj}}] \leftarrow N(D_{E^{\text{adj}}}^T)^T [C_{E^{\text{adj}}} \ b_{E^{\text{adj}}}]$  Lemma 3.22

*Normalise and ensure halfspace contains origin.*

9:  $[a_{\text{adj}}^T \ b_{\text{adj}}] \leftarrow \frac{\text{sign } b_{\text{adj}}}{\|a_{\text{adj}}\|_2} [a_{\text{adj}}^T \ b_{\text{adj}}]$

*Report adjacent facet.*

10: Report  $(E^{\text{adj}}, a_{\text{adj}}, b_{\text{adj}})$

---

### 4.3 RIDGE ORACLE

---

$d - 1 + n$ , then the rank of  $D_E$  is  $k - n$  and

$$P_E = \left\{ (x, y) \left| \begin{array}{l} a_f^T x = b_f, \\ Sx + L\tilde{y} \leq t, \\ y = N(D_E)\tilde{y} + D_E^\dagger(b_E - C_E x) \end{array} \right. \right\},$$

where  $E^c \triangleq \{1, \dots, q\} \setminus E$ , and

$$S \triangleq C_{E^c} - D_{E^c} D_E^\dagger C_E, \quad L \triangleq D_{E^c} N(D_E), \quad t \triangleq b_{E^c} - D_{E^c} D_E^\dagger b_E,$$

where  $D_E^\dagger$  is the Moore-Penrose pseudo-inverse of  $D_E$ .

*Proof.* The definition of  $P_E$  is

$$P_E = \left\{ (x, y) \left| \begin{array}{l} C_E x + D_E y = b_E, \\ C_{E^c} x + D_{E^c} y \leq b_{E^c} \end{array} \right. \right\}. \quad (4.6)$$

Let  $\left[ \begin{array}{cc} \hat{U} & \tilde{U} \end{array} \right] \left[ \begin{array}{cc} \Sigma & 0 \\ 0 & 0 \end{array} \right] \left[ \begin{array}{cc} \hat{V} & \tilde{V} \end{array} \right]^T \triangleq D_E$  be the singular value decomposition of  $D_E$  and introduce the change of variables  $\hat{V}\hat{y} + \tilde{V}\tilde{y} \triangleq y$ . Multiplying by  $\left[ \begin{array}{cc} \hat{U} & \tilde{U} \end{array} \right]^T$ , the affine hull of  $P_E$  can be written as the two equations:

$$\hat{U}^T C_E x + \Sigma \hat{y} = \hat{U}^T b_E \quad (4.7)$$

$$\tilde{U}^T C_{E^c} x = \tilde{U}^T b_{E^c} \quad (4.8)$$

Recalling that  $\tilde{U}$  is equal to  $N(D_E^T)$  we see from Lemma 3.22 that (4.8) defines the affine hull of the projection:  $a_f^T x = b_f$ . Solving (4.7) for  $\hat{y}$  and substituting into (4.6) gives the desired result:

$$P_E = \left\{ (x, y) \left| \begin{array}{l} a_f^T x = b_f, \\ (C_{E^c} - D_{E^c} D_E^\dagger C_E)x + D_{E^c} \tilde{V} \tilde{y} \leq b_{E^c} - D_{E^c} D_E^\dagger b_E, \\ y = \tilde{V} \tilde{y} + D_E^\dagger (b_E - C_E x) \end{array} \right. \right\},$$

where we notice that the pseudo-inverse  $D_E^\dagger$  is given by  $\hat{V} \Sigma^{-1} \hat{U}^T$ .

It remains to be shown that the rank of  $D_E$  is  $k - n$ . Lemma 3.17 gives the dimension of the projection  $\pi_x P_E$  as:

$$\dim \pi_x P_E = \dim P_E - k + \text{rank } D_E. \quad (4.9)$$

$\pi_x P_{\mathbf{E}}$  is a facet and therefore the dimension of  $\pi_x P_{\mathbf{E}}$  is  $d - 1$  and the dimension of  $P_{\mathbf{E}}$  has been assumed to be  $d - 1 + n$ , and therefore solving (4.9) provides the desired relation:  $\text{rank } D_{\mathbf{E}} = k - n$ .

□

**Corollary 4.12.** *If  $\mathbf{E}$  is an equality set of  $P$  such that  $\pi_x P_{\mathbf{E}}$  is a facet of  $\pi_x P$  and  $\dim P_{\mathbf{E}} = d - 1 + n$ , then*

$$\pi_x P_{\mathbf{E}} = \left\{ x \mid \exists \tilde{y} \in \mathbb{R}^n, Sx + L\tilde{y} \leq t, a_f^T x = b_f \right\},$$

where  $a_f, b_f, S, L$ , and  $t$  are as defined in Lemma 4.11.

*Proof.* Follows directly from Lemma 4.11. □

Note that if  $n = 0$ , Corollary 4.12 gives an explicit expression for  $\pi_x P_{\mathbf{E}}$  since  $\tilde{y} \in \mathbb{R}^0$ . We now distinguish two cases and the following sections present methods for computing the ridges for  $n = 0$  and  $n > 0$  respectively.

**Remark 4.13.** *From Theorem 4.8, the dimension of the face  $P_{\mathbf{E}}$  is  $d - 1$  if and only if the LP in (4.3) is not dual-degenerate.*

### 4.3.1 Case 1: $\dim P_{\mathbf{E}} = d - 1$

In this section the assumption is made that the dimension of the face  $P_{\mathbf{E}}$  is  $d - 1$ . From Corollary 4.12 we can see that the facet is given by  $\pi_x P_{\mathbf{E}} = \left\{ x \mid Sx \leq t, a_f^T x = b_f \right\}$ . We first prove that all facets of  $\pi_x P_{\mathbf{E}}$  (ridges of  $\pi_x P$ ) can be written as  $\pi_x P_{\mathbf{E} \cup \{i\}}$  for some  $i$  in  $\mathbf{E}^c$ . We then show how to select only those indices  $i$  that correspond to facets of  $\pi_x P_{\mathbf{E}}$  (ridges of  $\pi_x P$ ) and finally, how to construct the equality sets for each.

**Lemma 4.14.** *If  $\mathbf{E}$  is an equality set of  $P$  such that  $\pi_x P_{\mathbf{E}}$  is a facet of  $\pi_x P$  and  $\dim P_{\mathbf{E}} = d - 1$ , then for all  $i \in \mathbf{E}^c$ ,*

$$\pi_x P_{\mathbf{E} \cup \{i\}} = \left\{ x \mid \begin{bmatrix} a_f^T \\ S_i \end{bmatrix} x = \begin{bmatrix} b_f \\ t_i \end{bmatrix}, Sx \leq t \right\},$$

where  $a_f, b_f, S$  and  $t$  are as defined in Lemma 4.11.

*Proof.* The result follows directly from Corollary 4.12 by noting that if  $n$  is zero then  $\tilde{y} \in \mathbb{R}^0$ . □



### 4.3 RIDGE ORACLE

---

From Corollary 4.12 and Lemma 4.14, we see that for each facet  $F$  of  $\pi_x P_E$ , there exists an  $i \in E^c$  such that  $F = \pi_x P_{E \cup \{i\}}$ . It is clear from Lemma 4.14 and the definition of faces (Definition 3.5) that  $\pi_x P_{E \cup \{i\}}$  is a face of  $\pi_x P_E$  for all  $i \in E^c$ . Therefore,  $\pi_x P_{E \cup \{i\}}$  is a facet of  $\pi_x P_E$  (a ridge of  $\pi_x P$ ) if and only if it is of the appropriate dimension:  $\dim \pi_x P_{E \cup \{i\}} = \dim \pi_x P_E - 1 = d - 2$ .

The dimension of  $\pi_x P_{E \cup \{i\}}$  is  $d - 2$  if and only if the dimension of its affine hull is  $d - 2$ , or equivalently the rank of  $\begin{bmatrix} a_f^T & b_f \\ S_{Q(i)} & t_{Q(i)} \end{bmatrix}$  is two where  $Q(i)$  is the equality set of  $\pi_x P_{E \cup \{i\}}$ . Therefore, we can see that  $\pi_x P_{E \cup \{i\}}$  is a ridge if and only if its equality set is given by:

$$Q(i) \triangleq \left\{ j \in E^c \mid \text{rank} \begin{bmatrix} a_f^T & b_f \\ S_{\{i,j\}} & t_{\{i,j\}} \end{bmatrix} = 2 \right\}. \quad (4.10)$$

This idea is formalized in Proposition 4.15 below.

**Proposition 4.15.** *If  $E$  is an equality set of  $P$  such that  $\pi_x P_E$  is a facet of  $\pi_x P$  and  $\dim P_E = d - 1$ , then for all  $i \in E^c$ ,  $\pi_x P_{E \cup \{i\}}$  is a facet of  $\pi_x P_E$  if and only if  $\text{rank} \begin{bmatrix} a_f^T & b_f \\ S_i & t_i \end{bmatrix} = 2$  and there exists an  $x_0 \in \pi_x P_{E \cup \{i\}}$  such that  $S_j x_0 < t_j$  for all  $j \in E^c \setminus Q(i)$ , where*

$$Q(i) \triangleq \left\{ j \in E^c \mid \text{rank} \begin{bmatrix} a_f^T & b_f \\ S_{\{i,j\}} & t_{\{i,j\}} \end{bmatrix} = 2 \right\}.$$

*Proof.* Assume that  $\pi_x P_{E \cup \{i\}}$  is a facet of  $\pi_x P_E$ . We prove the existence of an  $x_0$  such that  $S_j x_0 < t_j$  for all  $j \in E^c \setminus Q(i)$  by contradiction. Assume such an  $x_0$  does not exist. Then there exists a  $j \in E^c \setminus Q(i)$  such that  $S_j x = t_j$  for all  $x \in \pi_x P_E$ . Since  $j \notin E^c \setminus Q(i)$  we have that

$$\begin{aligned} \dim \pi_x P_{E \cup \{i\}} &= \dim \text{aff } \pi_x P_{E \cup \{i\}} \\ &\leq \dim \left\{ x \mid \begin{bmatrix} x_f^T \\ S_{\{i,j\}} \end{bmatrix} x = \begin{bmatrix} b_f \\ t_{\{i,j\}} \end{bmatrix} \right\} \\ &= d - 3 \end{aligned}$$

and therefore an appropriate  $x_0$  exists by contradiction.

Assume that there exists an  $x_0$  such that  $S_j x_0 < t_j$  for all  $j \in E^c \setminus Q(i)$ . We define the set

$$B(\epsilon, x_0) \triangleq \left\{ x \mid \|x - x_0\|_2 < \epsilon, \begin{bmatrix} a_f^T \\ S_i \end{bmatrix} (x - x_0) = \begin{bmatrix} b_f \\ t_i \end{bmatrix} \right\}.$$

Note that  $\dim B(\epsilon, x_0) = d - 2$  for all  $\epsilon > 0$  and all  $x_0$ . Clearly, there exists an  $\epsilon > 0$  such that

$(x_0 - x) \in \pi_x P_{E \cup \{i\}}$  for all  $x \in B(\epsilon, x_0)$ . It follows that  $\pi_x P_{E \cup \{i\}}$  has a subset of dimension  $d - 2$  and therefore  $\dim \pi_x P_{E \cup \{i\}} \geq d - 2$ . The rank of  $\begin{bmatrix} a_f^T & b_f \\ S_i & t_i \end{bmatrix}$  is two and therefore  $\dim \pi_x P_{E \cup \{i\}} \leq d - 2$ . It follows that  $\pi_x P_{E \cup \{i\}}$  is a  $(d - 2)$ -face of  $\pi_x P_E$  and is therefore a facet of  $\pi_x P_E$ .

□

We now propose a linear program that will test if  $Q(i)$  as defined in Proposition 4.15 is an equality set of  $\pi_x P$ :

$$\begin{aligned}
 & \underset{(\tau, x)}{\text{minimize}} && \tau \\
 & \text{subject to} && S_{\setminus Q(i)} x \leq t_{\setminus Q(i)} + \tau \\
 & && a_f^T x = b_f \\
 & && S_i x = t_i \\
 & && \tau \geq -c, \quad \text{for some } c \in \mathbb{R}_{>0}
 \end{aligned} \tag{4.11}$$

If there exists an  $x$  and a strictly negative  $\tau$  that satisfies LP (4.11), then clearly we have found an  $x$  such that  $S_j x < t_j$  for all  $j \in E^c \setminus Q(i)$  and the requirements of Proposition 4.15 are satisfied. Therefore  $\pi_x P_{E \cup \{i\}}$  is a facet of  $\pi_x P_E$  if and only if there exists a feasible solution to LP (4.11) such that  $\tau$  is strictly less than zero. Note that  $x$  is bounded in LP (4.11) because the polytope  $P$  is assumed bounded and the cost is bounded below. The equality set of the ridge  $\pi_x P_{E \cup \{i\}}$  is then given by the set  $Q(i)$ .

### 4.3.2 Case 2: $\dim P_E = d - 1 + n$ , $n > 0$

Here we make the assumption that the dimension of the face  $P_E$  is larger than  $d - 1$ . As a result,  $n$  is strictly greater than zero and, from Corollary 4.12, we can write  $\pi_x P_E$  as a projection from  $\mathbb{R}^d \times \mathbb{R}^n$  to  $\mathbb{R}^d$ . Recall that each facet of  $\pi_x P_E$  is a ridge of  $\pi_x P$  and therefore computing this projection is equivalent to finding all the ridges that are subsets of  $\pi_x P_E$ . In this section we show how to compute this projection using ESP, which requires some slight work as it does not contain the origin and is not full-dimensional.

The polytope  $\pi_x P_E$  can be written as the projection from  $\mathbb{R}^d \times \mathbb{R}^n$  to  $\mathbb{R}^d$ :

$$\pi_x P_E = \pi_x \left\{ (x, \tilde{y}) \in \mathbb{R}^d \times \mathbb{R}^n \mid Sx + L\tilde{y} \leq t, a_f^T x = b_f \right\}, \tag{4.12}$$

where  $S$ ,  $L$  and  $t$  are as defined in Corollary 4.12. Recall from Corollary 3.20 that every facet

### 4.3 RIDGE ORACLE

---

$F$  of  $\pi_x P_{\mathbf{E}}$  can be written as

$$\begin{aligned} F &= (\pi_x P_{\mathbf{E}})_{\mathbf{R}} \\ &= \pi_x \left\{ (x, \tilde{y}) \in \mathbb{R}^d \times \mathbb{R}^n \mid S_{\mathbf{R}}x + L_{\mathbf{R}}\tilde{y} = t_{\mathbf{R}}, a_f^T x = b_f, Sx + L\tilde{y} \leq t \right\}, \end{aligned}$$

for some equality set  $\mathbf{R}$  of  $\pi_x P_{\mathbf{E}}$ . From Lemma 4.11 and Corollary 4.12 we can see that  $\pi_x P_{\mathbf{E} \cup \mathbf{R}} = (\pi_x P_{\mathbf{E}})_{\mathbf{R}}$  and therefore  $\mathbf{E} \cup \mathbf{R}$  is a ridge-defining equality set of  $P$ . All such facet-defining equality sets  $\mathbf{R}$  of  $\pi_x P_{\mathbf{E}}$  can be computed via a call to the ESP algorithm, which therefore also gives all of the ridge-defining equality sets  $\mathbf{E}_r = \mathbf{E} \cup \mathbf{R}$  of  $P$  such that  $\pi_x P_{\mathbf{E}_r} \subset \pi_x P_{\mathbf{E}}$ .

Note that (4.12) does not satisfy the assumptions that its projection is full dimensional or that the origin is contained in its interior, which is required if this projection is to be computed via ESP. Taking the singular value decomposition  $a_f^T = \begin{bmatrix} \sigma & 0 & \dots & 0 \end{bmatrix} V^T$ , we can re-write (4.12) as:

$$\begin{aligned} \pi_x P_{\mathbf{E}} &= \left\{ x \in \mathbb{R}^d \mid \exists \tilde{y} \in \mathbb{R}^n, \tilde{x} \in \mathbb{R}^{d-1}, Sx + L\tilde{y} \leq t, x = V \begin{bmatrix} b_f/\sigma \\ \tilde{x} \end{bmatrix} \right\} \\ &= \left\{ x \in \mathbb{R}^d \mid x = V \begin{bmatrix} b_f/\sigma \\ \tilde{x} \end{bmatrix}, \tilde{x} \in \pi_{d-1} \tilde{P} \right\}, \end{aligned} \quad (4.13)$$

where

$$\tilde{P} = \left\{ (\tilde{x}, \tilde{y}) \in \mathbb{R}^{d-1} \times \mathbb{R}^n \mid S\tilde{V}\tilde{x} + L\tilde{y} \leq t - S\hat{V}b_f/\sigma \right\} \quad (4.14)$$

and  $V$  is partitioned appropriately as  $V = \begin{bmatrix} \hat{V} & \tilde{V} \end{bmatrix}$ .

Note that as the dimension of  $\pi_x P_{\mathbf{E}}$  is  $d - 1$ ,  $\tilde{P}$  is guaranteed to be full-dimensional. The reader is referred to Section 5.4 for a simple procedure to ensure that the origin is interior to  $\tilde{P}$ . The projection  $\pi_{d-1} \tilde{P}$  can now be computed using a recursive call to the ESP algorithm.

Finally, for each facet  $\left\{ \tilde{x} \mid \tilde{a}_f \tilde{x} = \tilde{b}_f \right\} \cap \pi_{d-1} \tilde{P}$  of  $\pi_{d-1} \tilde{P}$ , the corresponding ridge of  $\pi_x P$  can be computed from (4.13) as:

$$\left\{ x \mid \tilde{a}_f \tilde{V}^T x = \tilde{b}_f \right\} \cap \pi_x P_{\mathbf{E}} \quad (4.15)$$

**Remark 4.16.** *The adjacency oracle requires that the equation describing each ridge has a unit normal and is orthogonal to the facet. Once the ridge  $\left\{ x \mid a_r^T x = b_r \right\} \cap \pi_x P_{\mathbf{E}}$  has been*

computed, an equivalent representation that has the required properties can be computed as:

$$\begin{aligned} \begin{bmatrix} a_r^T & b_r \end{bmatrix} &\leftarrow \begin{bmatrix} a_r^T & b_r \end{bmatrix} - a_f^T a_r \begin{bmatrix} a_f & b_f \end{bmatrix} \\ \begin{bmatrix} a_r^T & b_r \end{bmatrix} &\leftarrow \frac{\text{sign } b_r}{\|a_r\|_2} \begin{bmatrix} a_r^T & b_r \end{bmatrix} \end{aligned} \quad (4.16)$$

### 4.3.3 Projection to 1D

This section is included in order to guarantee that the recursion introduced when the dimension of  $P_{\mathbb{E}}$  is larger than  $d - 1$  terminates.

If  $d$  is one, then the facets of the projection are vertices and there are clearly only two of them. They can be computed by maximizing and minimizing along the  $x$ -axis as follows

$$\begin{aligned} (x_{\min}^*, y_{\min}^*) / (x_{\max}^*, y_{\max}^*) &\triangleq \min / \max_{(x,y)} x \\ &\text{subject to } Cx + Dy \leq b \end{aligned} \quad (4.17)$$

If LP (4.17) is not dual degenerate, then the equality sets of the ridges are given by

$$\begin{aligned} \mathbb{E}_{\min} &\triangleq \{i \mid Cx_{\min}^* + Dy_{\min}^* = b\}, \\ \mathbb{E}_{\max} &\triangleq \{i \mid Cx_{\max}^* + Dy_{\max}^* = b\}. \end{aligned}$$

If LP (4.17) is dual degenerate, then the equality sets must be computed as in Section 4.2. The pre-image of the facets  $x_{\min}^*$  and  $x_{\max}^*$  are given by

$$\begin{aligned} \pi_x^{-1} x_{\min}^* &= \pi_x^{-1} P_{\mathbb{E}_{\min}} = P \cap \{(x_{\min}^*, y)\}, \\ \pi_x^{-1} x_{\max}^* &= \pi_x^{-1} P_{\mathbb{E}_{\max}} = P \cap \{(x_{\max}^*, y)\}, \end{aligned}$$

where the equality sets  $\mathbb{E}_{\min}$  and  $\mathbb{E}_{\max}$  can be computed as in Section 5.2.

Algorithm 4.3 describes the procedures developed in this section in algorithmic form.

## 4.4 Shooting Oracle

This section discusses the oracle  $(\mathbb{E}_0, a_f, b_f) = \text{SHOOT}(P)$  introduced in Section 4.1. The goal of the oracle is to initialise the ESP algorithm by finding a random equality set  $\mathbb{E}_0$  such that  $\pi_x P_{\mathbb{E}_0}$  is a facet of the projection  $\pi_x P$ .

This oracle operates by randomly choosing a direction  $\gamma \in \mathbb{R}^d$  and then solving an LP in order to move along the ray  $\gamma r$ ,  $r \geq 0$  until a point is found on the boundary of  $\pi_x P$ . If the point is not on a facet of  $\pi_x P$  then the ray is randomly perturbed until it is. The active

#### 4.4 SHOOTING ORACLE

---



---

**Algorithm 4.3** Ridge oracle:  $\mathbf{E}_r = RDG(\mathbf{E}, a_f, b_f)$

---

**Input:** Polytope  $P$  and equality set  $\mathbf{E}$  such that  $\pi_x P_{\mathbf{E}}$  is a facet of  $\pi_x P$ .

**Output:** List  $\mathbf{E}_r$  whose elements are all equality sets  $E_r$  such that  $\pi_x P_{E_r}$  is a facet of  $\pi_x P_{\mathbf{E}}$ .

*Initialize variables.*

- 1:  $\mathbf{E}_r \leftarrow \emptyset$
- 2: Compute  $S$ ,  $L$  and  $t$  as in Lemma 4.11

*Compute the dimension of  $P_{\mathbf{E}}$ .*

- 3: **if**  $\text{rank} \begin{bmatrix} C_{\mathbf{E}} & D_{\mathbf{E}} \end{bmatrix} < k + 1$  **then**

*Call ESP recursively to compute the ridges.*

- 4:  $\tilde{\mathbf{E}}_r \leftarrow \text{ESP}(\tilde{P})$  (4.13)

- 5: Convert the facets of  $\pi_{d-1} \tilde{P}$  into ridges of  $\pi_x P$  and add to  $\mathbf{E}_r$ . (4.15)

- 6: **else**

*Test each  $i \in E^c$  to see if  $\mathbf{E} \cup \{i\}$  defines a ridge.*

- 7: **for each**  $i$  in  $E^c$  **do**
  - 8:     Compute  $\tau^*$  from LP (4.11)
  - 9:     **if**  $\tau^* < 0$  **then** Proposition 4.15
  - 10:         Compute equality set  $Q(i)$
  - 11:          $\begin{bmatrix} a_r^T & b_r \end{bmatrix} \leftarrow \begin{bmatrix} S_i & t_i \end{bmatrix}$
  - 12:          $\mathbf{E}_r \leftarrow ((Q(i), a_r, b_r), \mathbf{E}_r)$
  - 13:     **end if**
  - 14: **end for**
  - 15: **end if**
  - 16: Normalize all equations and make orthogonal to the facet  $\pi_x P_{\mathbf{E}}$  Remark 4.16
-

constraints that define the facet can then be calculated from the constraints of  $P$  that are satisfied with equality at this point.

The following LP will be used to search in the direction of the ray  $\gamma r$  until a facet of the projection is found:

$$\begin{aligned} (r^*, y^*) \triangleq & \operatorname{argmax}_{(r,y) \in \mathbb{R} \times \mathbb{R}^k} r \\ & \text{subject to } C\gamma r + Dy \leq b \end{aligned} \tag{4.18}$$

The point  $(\gamma r^*, y^*)$  will be on the face of  $P$  that projects to the face of  $\pi_x P$  intersecting the ray  $\gamma r$ , where  $r$  is a positive real number. This is equivalent to LP (4.3) where a point was computed on the adjacent facet. The method of computing the equality set of the face of  $P$  that contains  $(\gamma r^*, y^*)$  is the same as that presented in Section 4.2. Namely, if the optimiser of the LP is unique, then the equality set is given by  $E_0 \triangleq \{i \mid C_i \gamma r^* + D_i y^* = b_i\}$ . However, if the LP is dual-degenerate, then the equality set can be computed by calculating the affine hull of the pre-image:

$$P_{E_0} = \pi_x^{-1} \pi_x P_{E_0} = P \cap \{x \mid a_f^T x = b_f\},$$

where  $a_f$  and  $b_f$  are given by Lemma 3.22.

**Remark 4.17.** *The ESP algorithm assumes that the polytope  $P$  contains the origin. This ensures that the ray  $\gamma r$  will intersect the polytope for some value of  $r \geq 0$ . If an initial equality set is already known, then the interiority assumption can be relaxed.*

## 4.5 Complexity Analysis

The complexity of the ESP algorithm is best described in terms of the number of linear programs needed. Let  $LP(n, q)$  be the time complexity of an LP of dimension  $n$  with  $q$  constraints. For the simplex approach, the average complexity is linear in  $q$  and polynomial (approximately to the power three) in  $n$ , while the worst case is exponential. For interior point methods the worst case is polynomial to within a given  $\epsilon$ -bound of the optimum, however, there is a large constant multiplier on the interior point approach and as a result it is often slower for ‘small’ problems (less than a few hundred dimensions).

First, assume that none of the linear programs computed during the adjacency oracle are dual-degenerate. If the polytope  $P$  is in  $\mathbb{R}^d \times \mathbb{R}^k$  and has  $q$  constraints, then one LP in  $d + k$  dimensions with  $q$  constraints must be solved per output facet in order to compute its equality set. Computing the ridges of each facet requires a redundancy removal operation.

## 4.5 COMPLEXITY ANALYSIS

---



---

**Algorithm 4.4** Shooting oracle: Calculation of a random facet of  $\pi_x P$

---

**Input:** Polytope  $P = \{(x, y) \in \mathbb{R}^d \times \mathbb{R}^k \mid Cx + Dy \leq b\}$  that contains the origin in its interior and whose projection is full-dimensional.

**Output:** A randomly selected equality set  $E_0$  of  $P$  such that  $\pi_x P_{E_0} \triangleq \{x \mid a_f^T x = b_f\} \cap \pi_x P$  is a facet of  $\pi_x P$ .

*Find a face of  $P$  that projects to a facet of  $\pi_x P$*

1: **repeat**

2:   Choose a random vector  $\gamma \in \mathbb{R}^d$

3:   Compute

$$(r^*, y^*) \triangleq \underset{(r, y) \in \mathbb{R} \times \mathbb{R}^k}{\operatorname{argmax}} \quad r$$

subject to  $C\gamma r + Dy \leq b$

4:    $E_0 \leftarrow \{i \mid C_i \gamma r^* + D_i y^* = b_i\}$

5: **until**  $\dim \pi_x P_{E_0} = d - \operatorname{rank} \mathbf{N}(D_{E_0}^T)^T C_{E_0} = d - 1$

*Compute affine hull of facet*

6:  $\begin{bmatrix} a_f^T & b_f \end{bmatrix} \leftarrow \mathbf{N}(D_{E_0}^T)^T \begin{bmatrix} C_{E_0} & b_{E_0} \end{bmatrix}$

Lemma 3.22

7:  $\begin{bmatrix} a_f^T & b_f \end{bmatrix} \leftarrow \frac{\operatorname{sign} b_f}{\|a_f\|_2} \begin{bmatrix} a_f^T & b_f \end{bmatrix}$

Remark 4.16

*Handle dual-degeneracy in LP*

Section 4.2

8: **if** LP is dual-degenerate in 3 **then**

9:   Compute equality set  $E_0$  such that  $P_{E_0} = \{(x, y) \mid a_f^T x = b_f\} \cap P$

Section 5.2

10: **end if**

*Report facet*

11: Report  $(E_0, a_f, b_f)$ .

---

This can be done in  $q - |\mathbf{E}_r|$ ,  $d$ -dimensional LPs with  $q - |\mathbf{E}_r|$  constraints. If  $n_f$  is the number of inequalities in the projection  $\pi_x P$ , then the worst-case time complexity is

$$\mathcal{O}(n_f(LP(d+k, q) + (q - |\mathbf{E}_r|)LP(d, q - |\mathbf{E}_r|))).$$

If the projection polytope is in general position, then  $|\mathbf{E}_r|$  is  $k$  for all facets and the complexity becomes:

$$\mathcal{O}(n_f(LP(d+k, q) + (q - k)LP(d, q - k))). \quad (4.19)$$

Clearly, the complexity is linear in the number of output facets.

It is interesting to note that if the dimension that we are projecting to and the number of constraints are held constant, then the complexity becomes  $\mathcal{O}(n_f LP(k, q))$ . Finally, if the size of the polytope  $P$  (dimension and number of constraints) is fixed, then the complexity is linear in the number of output facets,  $\mathcal{O}(n_f)$ .

#### 4.5.1 Degeneracy

If degeneracy is encountered, then ESP will need to be called recursively and the algorithm will no longer be output sensitive. This is not surprising as there currently exist no output sensitive algorithms for any basic geometric operations in the degenerate case.

The worst-case is clearly if every facet of the projection is degenerate and their preimages are  $(d+k-1)$ -dimensional faces of  $P$ . ESP would then be called recursively, projecting these faces from  $\mathbb{R}^{d+k-1}$  to  $\mathbb{R}^{d-1}$ . Clearly, this could continue recursively until the projection is to  $\mathbb{R}^1$ . The result is that in the worst-case, the complexity of ESP is a function of the number of faces in  $P$ , rather than the number of facets of  $\pi_x P$ . As it is possible for the number of faces of  $P$  to be exponentially larger than the number of facets in  $\pi_x P$ , in the worst-case ESP is an exponential algorithm.

Despite this worst-case behaviour, simulations in Chapter 6 demonstrate that ESP is very well suited to many types of polytopes encountered in control problems, which are both degenerate and non-degenerate.



## Extensions and Implementation Details

### 5.1 Degeneracy

In this section we will discuss a second method of dealing with degeneracy. A facet of the projection is called *degenerate* if its pre-image is of dimension larger than  $d - 1$ , where  $d$  is the dimension of the projection. A method to handle this problem was introduced in Section 4.3, which involved recursive calls to the ESP algorithm. We here present a second method that perturbs the cost function of the linear program used in the adjacency oracle such that it cannot be dual-degenerate. We will refer to these two methods as the recursive and perturbation methods respectively.

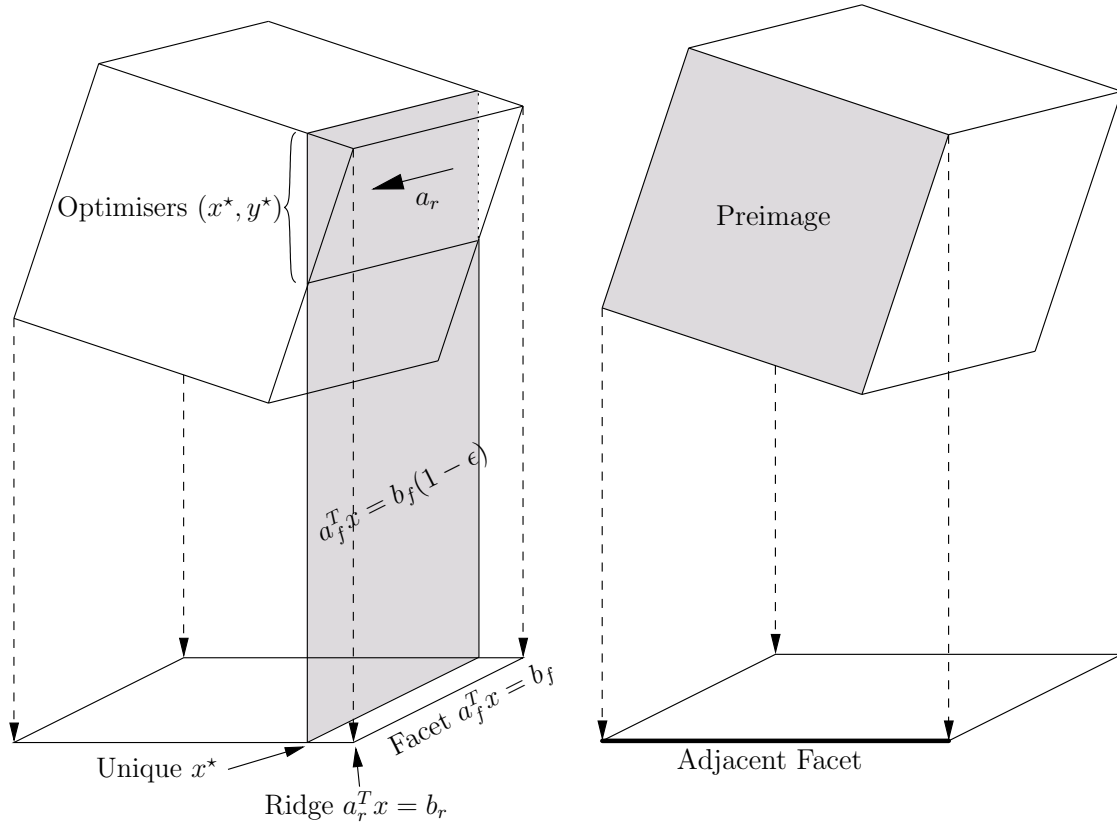
Consider the following modification of LP (4.3):

$$\begin{aligned}
 (x^*, y^*) = \operatorname{argmax}_{x, y} & \quad \left( \begin{bmatrix} a_r \\ 0 \end{bmatrix} + \epsilon \right)^T \begin{pmatrix} x \\ y \end{pmatrix} \\
 \text{subject to} & \quad C_{E_r} x + D_{E_r} y \leq b_{E_r} \\
 & \quad a_f^T x = b_f(1 - \delta),
 \end{aligned} \tag{5.1}$$

where  $\epsilon = \left[ \epsilon_0 \quad \epsilon_0^2 \quad \dots \quad \epsilon_0^{d+k} \right]^T$  and  $\epsilon_0$  is a sufficiently small, positive number. This *lexicographic perturbation* will be discussed in detail in Section 8.5, but here we will simply state the main results: LP (5.1) is not dual-degenerate for sufficiently small  $\epsilon$  and the (unique) optimiser  $(x^*, y^*)$  of LP (5.1) is also an optimiser of LP (4.3).

The benefit of using LP (5.1) for the adjacency oracle is clearly that the pre-image of each facet will be of dimension  $d - 1$  and therefore the ridges can be computed without any recursion. The cost is that a degenerate facet may be rediscovered several times. This is illustrated through an example in Figures 5.1 and 5.2 where the adjacent facet is degenerate.

In Figure 5.1 we can see that the recursive method will choose an optimiser in the interior of the preimage of the adjacent degenerate facet and will therefore return the equality set of the preimage of the entire face. As the ridges cannot be computed directly from this 2-dimensional face, a recursive call is needed to ESP. Contrast this to Figure 5.2, where the perturbation method causes the adjacency oracle to choose a particular 1-dimensional face of  $P$  to be the preimage.



(a) Multiple optimisers: ESP will choose one in the strict interior of the preimage of  $x^*$

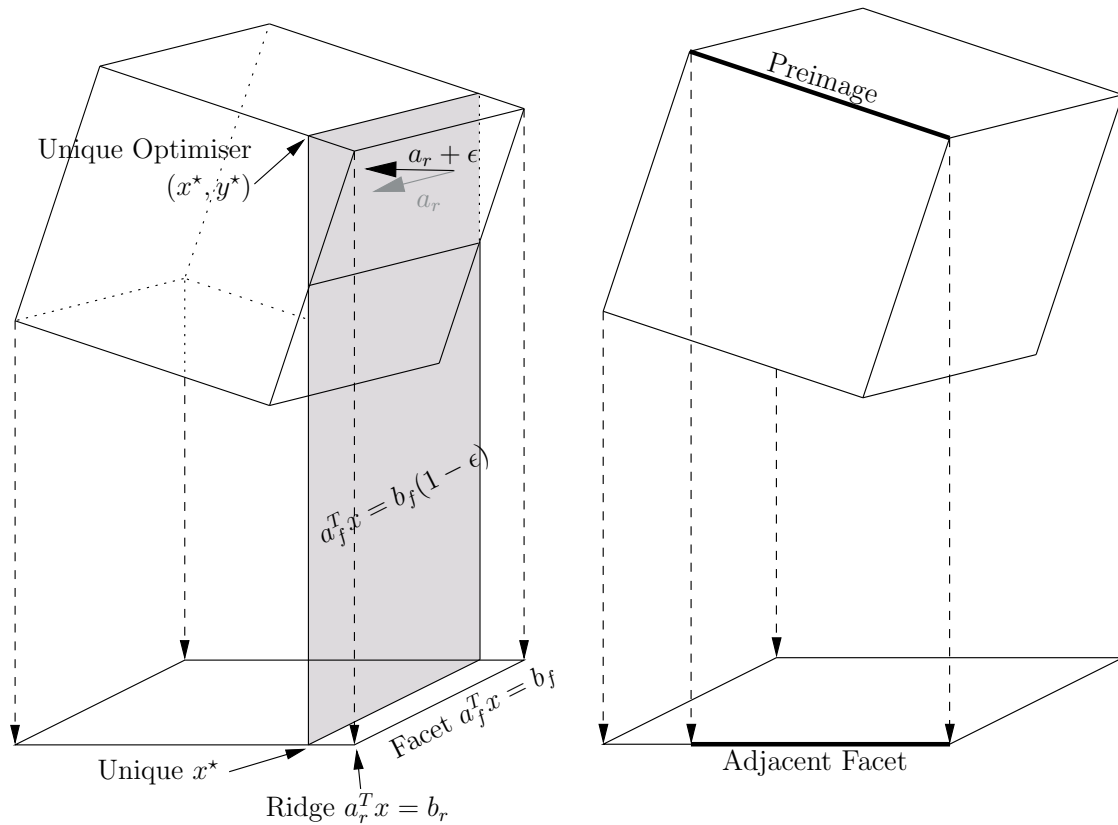
(b) Preimage is not of dimension  $d - 1$  and a recursive call to ESP must be made to compute the ridges.

Figure 5.1: Degenerate Projection Example: Recursive Method

If there are degenerate facets in the projection, then the ESP algorithm will no longer be output sensitive. The complexity of the perturbation method is clearly a function of the number of  $(d - 1)$ -dimensional faces of the polytope  $P$  that project into facets of  $\pi_x P$ . As this number of faces is known to be worst-case exponential, the ESP algorithm is clearly worst-case exponential under both degeneracy handling methods.

## 5.1 DEGENERACY

---



(a) Perturbation of the cost makes the optimiser unique.

(b) Ridges can be computed directly, but ESP will find the facet more than once.

Figure 5.2: Degenerate Projection Example: Perturbation Method

It is difficult to compare the complexity of the two approaches, as there exist polytopes for which each is superior. Consider Figure 5.3, where two polytopes are projected from  $R^3$  to  $R^2$ . It is possible for one method to be exponentially better than the other, as demonstrated by the polytopes in Figure 5.3, if they were taken to higher dimensions.

This analysis makes it difficult to choose between the two approaches as it is unknown how to tell which will be better *a priori*. Computational experience has demonstrated that the perturbation method of handling degeneracy is often more numerically robust than the recursive approach. However, as many geometers have found, using rational arithmetic can greatly alleviate such problems. It has also been seen that for most problems of interest to control, both methods are approximately equal. For these reasons, we generally choose the perturbation method over the recursive.

## 5.2 Calculation of the Affine Hull

This section presents a well known algorithm for the computation of the equality set  $E$  of a polytope  $P$  such that  $P_E = P$  [Bor02, Alg. 1.3.1]. The input to the algorithm is the matrix  $A \in \mathbb{R}^{q \times n}$  and the vector  $b \in \mathbb{R}^q$  that defines the polytope  $P \triangleq \{z \in \mathbb{R}^n \mid Az \leq b\}$ . The output is an equality set  $E$  such that  $P_E = P$  and the affine hull is given by  $\text{aff } P = \{z \mid A_E z = b_E\}$ .

Recall that the equality set of  $P$  consists of all constraints that are active at every point in the polytope. Therefore, if a point can be found in the polytope for which a given constraint is not active, then that constraint is clearly not in the equality set. Given a constraint  $i \in \{1, \dots, q\}$ , a point exists for which it is not active if  $A_i z - b_i$  is strictly less than zero for some  $z$ . We can search for such a point by minimizing the value of  $A_i z - b_i$ :

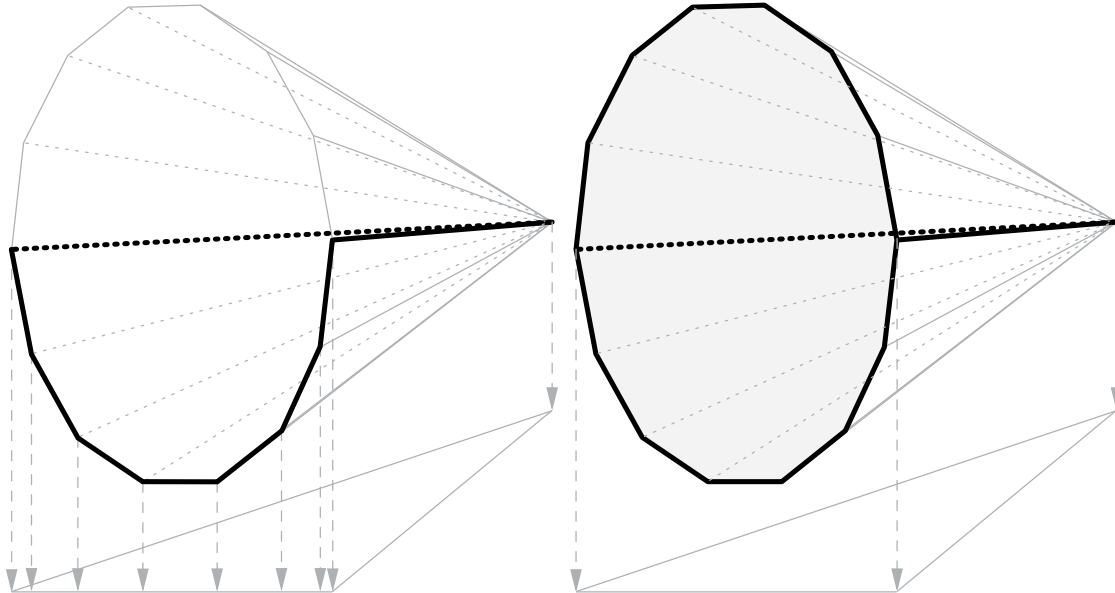
$$J(i)^* \triangleq \begin{array}{ll} \underset{z}{\text{minimise}} & A_i z - b_i \\ \text{subject to} & Az \leq b. \end{array}$$

If  $J(i)^*$  is strictly negative, then constraint  $i$  is not in the equality set. If it is positive, then the constraint is redundant and if it is equal to zero then it is in the equality set.

## 5.3 Projection of non Full-Dimensional Polytopes

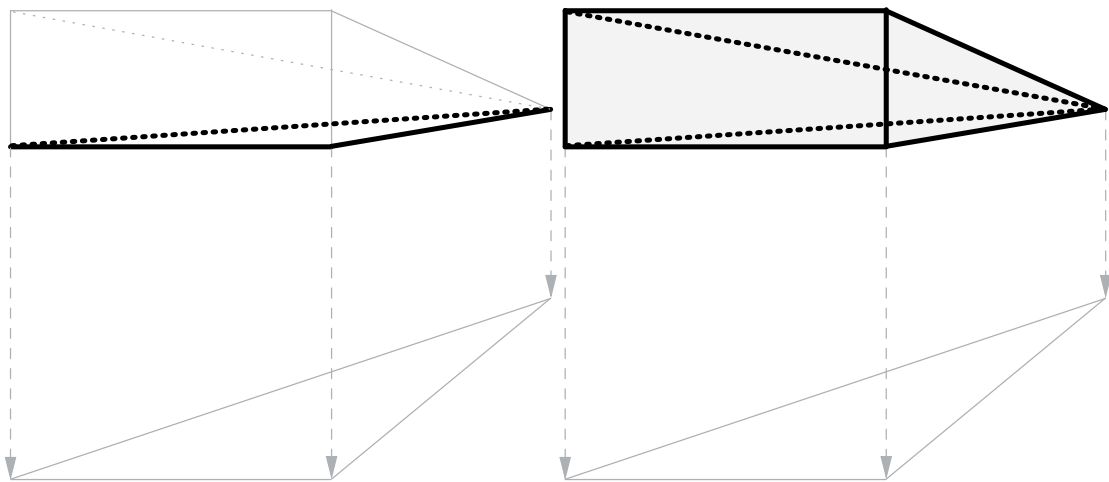
If  $\pi_x P$  is not full-dimensional, then  $P$  must have a non-trivial affine hull and there exists an equality set  $A$  such that  $P_A = P$ . An algorithm for computing the affine hull of a polytope, and the equality set of the constraints that determine it was given in Section 5.2. If  $A$  is the

### 5.3 PROJECTION OF NON FULL-DIMENSIONAL POLYTOPES



(a) The perturbation method causes the front facet to be re-discovered seven times.

(b) The recursive method needs to be called exactly once for the same polytope.



(c) The perturbation method visits each of the degenerate facets exactly once.

(d) The recursive method must recurse for each of the three facets.

Figure 5.3: Example Projections that are Good/Bad for both Degeneracy Methods.

equality set of the affine hull, then from Lemma 3.22 we can write the projection  $\pi_x P$  as

$$\pi_x P = \{x \in \mathbb{R}^n \mid \exists y, Fx = f, Cx + Dy \leq b\}, \quad (5.2)$$

where  $F \triangleq \mathbf{N}(D_A^T)^T C_A$  and  $f \triangleq \mathbf{N}(D_A^T)^T b_A$ . We now form a polytope  $\tilde{P}$  that satisfies the assumption that its projection is full-dimensional and from which we can recover the projection  $\pi_x P$ .

The equation  $Fx = f$  can be equivalently written as  $x = \mathbf{N}(F)\tilde{x} + F^\dagger f$ , for  $\tilde{x} \in \mathbb{R}^{\dim \pi_x P}$ , where we note that  $\dim \pi_x P = n - \text{rank } F$ . Defining  $\tilde{P}$  as

$$\tilde{P} \triangleq \left\{ (\tilde{x}, y) \mid C\mathbf{N}(F)\tilde{x} + Dy \leq b - CF^\dagger f \right\},$$

the polytope  $\pi_x P$  becomes

$$\begin{aligned} \pi_x P &= \left\{ x \in \mathbb{R}^n \mid \exists y, x = \mathbf{N}(F)\tilde{x} + F^\dagger f, (\tilde{x}, y) \in \tilde{P} \right\} \\ &= \left\{ x \in \mathbb{R}^n \mid x = \mathbf{N}(F)\tilde{x} + F^\dagger f, \tilde{x} \in \pi_{\tilde{x}} \tilde{P} \right\}. \end{aligned}$$

We now show that  $\tilde{P}$  satisfies the assumption that  $\pi_{\tilde{x}} \tilde{P}$  is full-dimensional.

$$\begin{aligned} &\pi_{\tilde{x}} \text{aff } \tilde{P} \\ &= \pi_{\tilde{x}} \left\{ (\tilde{x}, y) \mid C_A \mathbf{N}(F)\tilde{x} + D_A y = b_A - C_A F^\dagger f \right\} \\ &= \left\{ \tilde{x} \mid \mathbf{N}(D_A^T)^T C_A \mathbf{N}(F)\tilde{x} = \mathbf{N}(D_A^T)^T (b_A - C_A F^\dagger f) \right\} \\ &= \mathbb{R}^{\dim \pi_x P} \end{aligned} \quad (5.3)$$

where (5.3) follows because  $F$  is defined as  $\mathbf{N}(D_A^T)^T C_A$ .

Once the projection of  $\tilde{P}$  is computed using ESP, an expression is needed to recover  $\pi_x P$ .  $\tilde{x}$  can be written as a function of  $x$  as  $\mathbf{N}(F)^T x = \tilde{x}$ . If  $\pi_{\tilde{x}} \tilde{P}$  is given by  $\{\tilde{x} \mid G\tilde{x} \leq g\}$  then

$$\begin{aligned} \pi_x P &= \left\{ x \mid x = \mathbf{N}(F)\tilde{x} + F^\dagger f, G\tilde{x} \leq g \right\} \\ &= \left\{ x \mid Fx = f, G\mathbf{N}(F)^T x \leq g \right\}. \end{aligned}$$

## 5.4 Projection of Polytopes that do not Contain the Origin

In this section we consider a polytope  $P$  that does not contain the origin in its interior. The solution presented here is to compute a strictly interior point  $(x_0, y_0)$  and then translate the polytope such that this point gets mapped to the origin. After running ESP on the translated

## 5.4 PROJECTION OF POLYTOPES THAT DO NOT CONTAIN THE ORIGIN

---

polytope, the projection is then translated back by  $-x_0$  to restore the origin placement.

The following linear program will compute an interior point of polytope  $P$  whose equality set is  $A$ :

$$\begin{aligned}
 (x_0, y_0) \triangleq \underset{(\tau, x, y)}{\operatorname{argmin}} \quad & \tau \\
 \text{subject to} \quad & C_{A^c}x + D_{A^c}y \leq b_{A^c} + \tau \\
 & C_Ax + D_Ay = b_A
 \end{aligned} \tag{5.4}$$

Translating  $P$  by  $(x_0, y_0)$  gives  $\hat{P} \triangleq \{(x, y) \mid Cx + Dy \leq b + Cx_0 + Dy_0\}$ . If the projection of  $\hat{P}$  is  $\pi_x \hat{P} = \{x \mid \hat{G}x \leq \hat{g}\}$ , then  $\pi_x P = \{x \mid \hat{G}x \leq \hat{g} - \hat{G}x_0\}$ .





## Projection Examples

In this chapter we will present various comparative results on projections that are of interest to control. While there exists a class of polytopes for which each projection method is optimal, we will see that ESP is particularly and uniquely well suited to the types of problems of interest in control.

All simulations presented in this section were carried out on a Pentium 4, 3GHz machine with 2GB of RAM.

### 6.1 Random Polytopes

This section shows comparative simulations of ESP against other commonly used projection algorithms. Each algorithm is presented with an irredundant halfspace description of a polytope  $P$  and the output of the algorithm is an irredundant halfspace description of the projection  $\pi_x P$ . The polytopes to be projected have been chosen randomly, and therefore are very likely to be non-degenerate. A study of the degeneracy properties of ESP will be seen in a later section.

Four algorithms have been compared:

1. Fourier Elimination: This algorithm was implemented by the author of this thesis in C as described in [SH95].
2. Projection Cone: If the projection cone is defined as  $W \triangleq \{v \mid vD = 0, v \geq 0\}$ , then the projection is given by  $\pi_x P = \{x \mid (vC)x \leq vb, \forall v \in \text{extr } W\}$ . The extreme rays of  $W$  were computed via the double description method CDD, which is implemented in C [FP96] (version 0.93b, floating point arithmetic).

3. Vertex Enumeration: In this approach, the vertices of the polytope  $P$  are enumerated, projected and then the convex hull of the projection is calculated. All computations are done via the double description method CDD [FP96].
4. ESP: The ESP algorithm is implemented in MATLAB and all linear programs are computed via the Stanford Systems Optimization Laboratory (SOL) toolbox [MS] (Tomlab release 4.0.0)

As we require the output to be irredundant, each inequality generated by the first two approaches is tested for redundancy by a call to a linear program. For a fair comparison, the LP code used is the Stanford Systems Optimization Laboratory (SOL) toolbox [MS].

**Remark 6.1.** *Note that although at first glance, enumerating the vertices of the polytope as in method 3 above may seem slow in all cases, there are classes of polytopes for which it is fast. Furthermore, it is advocated in leading software packages, such as the Geometric Bounding Toolbox (GBT) [Sys].*

Several approaches are available to compute the extreme rays in 2 and the vertices in 3, including [BDH96, Avi00, FP96, CL97, Cla, BFM98a]. As with most polytopical algorithms, the efficiency of these methods varies widely as a function of the structure of the polytope: a given algorithm will be fast for a certain class of polytopes and slow for a different one. It was found, however, that for the class of polytopes considered here, CDD [FP96] was generally faster than, or similar speed to the methods [BDH96, Avi00, BFM98a]. Therefore, we have chosen to compare ESP with CDD.

The polytopes that are used in this comparison are randomly generated, bounded by a hypersphere of radius 10 and all inequalities in their description are irredundant. Figures 6.1 and 6.1 shows the results of the simulations where the  $y$ -axis is a logarithmic scale of the time taken per facet of the projection and the  $x$ -axis is the variable  $\xi$ , which is defined in the following list. Let  $q$  be the number of halfspaces in the description of  $P$ . We define four scenarios:

a.  $\mathbb{R}^{10} \longrightarrow \mathbb{R}^4, \quad q = \xi, \quad \xi = 10, \dots, 100$

This test demonstrates the complexity of the algorithms as a function of the number of halfspaces in the polytope. From (4.19) and recalling that an LP has an average linear complexity for fixed dimension, the complexity per facet of the projection of ESP is  $\mathcal{O}(q^2)$ ; this tendency can be seen in Figure 6.1(a).

b.  $\mathbb{R}^\xi \longrightarrow \mathbb{R}^4, \quad q = 3\xi, \quad \xi = 5, \dots, 30$

In this test we project to a fixed dimension while increasing the dimension of the polytope  $P$ . From (4.19), the complexity of ESP should increase as  $\mathcal{O}(LP(\xi, 3\xi))$ . It is clear

## 6.2 FEASIBILITY

---

from Figure 6.1(b) that for these polytopes, ESP is capable of projecting from much higher dimensions than the other approaches. The main limitation as the dimension increases is the rapid growth of the number of facets in the projection.

- c.  $\mathbb{R}^\xi \longrightarrow \mathbb{R}^4$ ,  $q = 2\xi$ ,  $\xi = 5, \dots, 70$   $P$  is a hypercube

This scenario computes the projection of rotated hypercubes to  $\mathbb{R}^4$ . Note that while a  $n$ -dimensional hypercube has  $2n$  facets, it contains  $2^n$  vertices. Therefore, the projection cone and vertex enumeration approaches are ill-suited in this case. However, as can be seen from Figure 6.1(c), ESP can handle very high dimensions.

- d.  $\mathbb{R}^\xi \longrightarrow \mathbb{R}^2$ ,  $q = 3\xi$ ,  $\xi = 5, \dots, 50$

The final test shows the projection of randomly generated polytopes from high dimensions to  $\mathbb{R}^2$ . The procedure described in Section 4.3.3 for the calculation of ridges in 1D makes ESP particularly suited to this task, as can be readily seen from Figure 6.1(d).

**Remark 6.2.** *Note that difference in speed between ESP and the other algorithms for small problems is likely to be due in large part to the overhead involved in the implementation of ESP in MATLAB.*

## 6.2 Feasibility

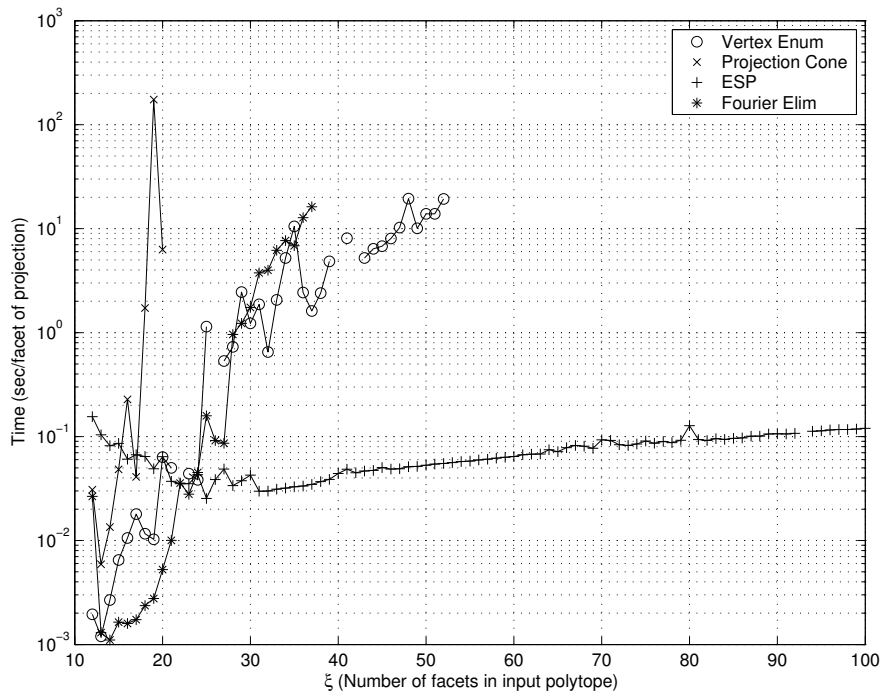
We now examine one of the main uses of projection in a control context: the calculation of the region of feasibility. In constrained control, polytopes arise naturally as input and/or state constraints. For example, sets of halfspaces can be used to describe the physical position or rate limits of a valve or the maximum safe pressure of a tank. Once constraints are placed on the operation of a system, it is no longer necessarily the case that a control action can be found for every state that satisfies all constraints. Determining the states of a linear system with polytopic constraints for which there exists a feasible control action is called the feasibility problem and requires a projection of polytopes.

Consider the following linear system:

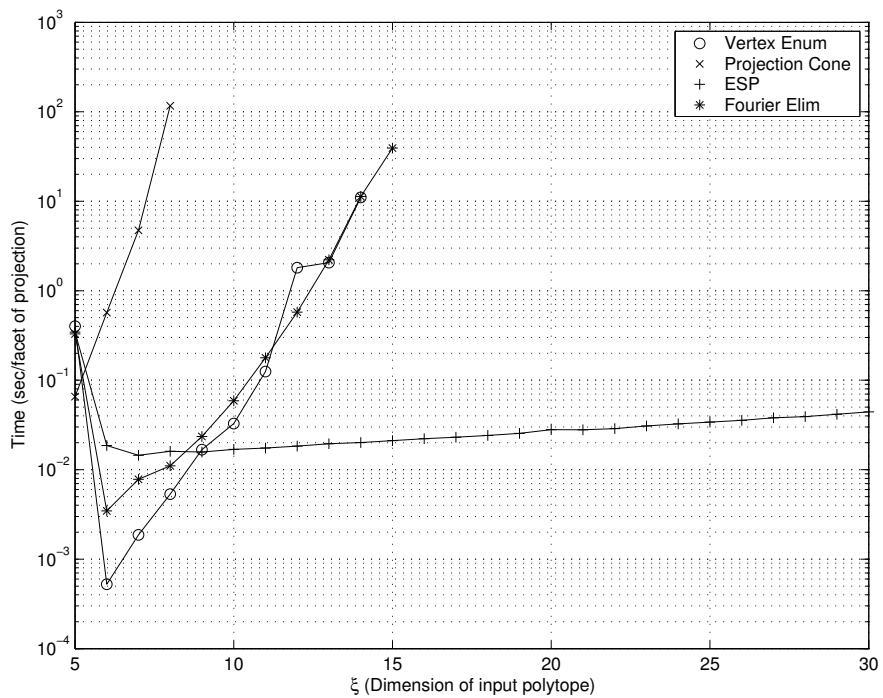
$$x^+ = Ax + Bu,$$

where  $x \in \mathbb{R}^n$  is the state,  $x^+$  is the successor state and  $u \in \mathbb{R}^m$  is the input. Consider the

## 6. PROJECTION EXAMPLES



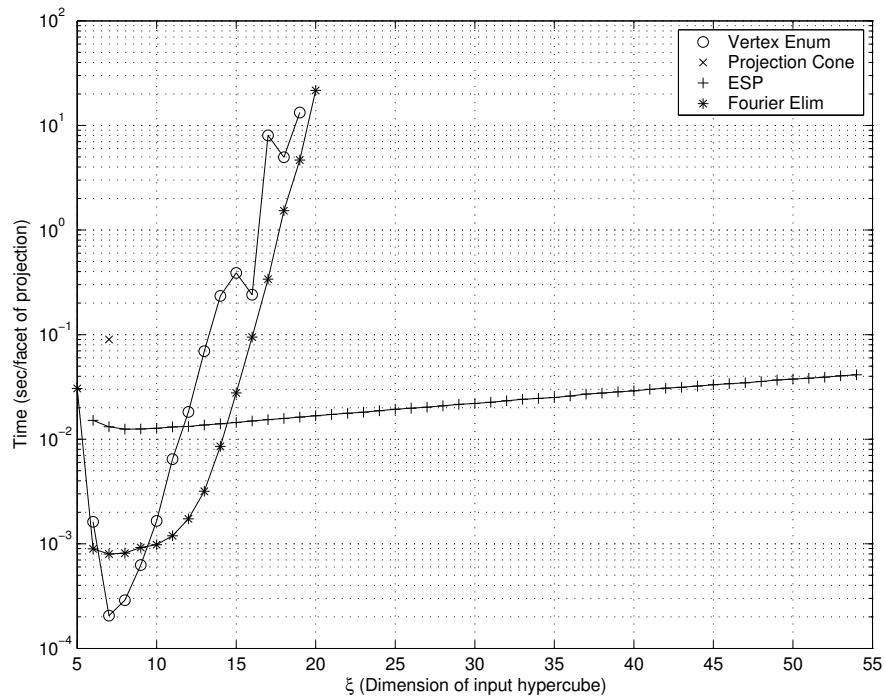
(a)  $\mathbb{R}^{10} \rightarrow \mathbb{R}^4$ ;  $\xi =$  number of facets in  $P$



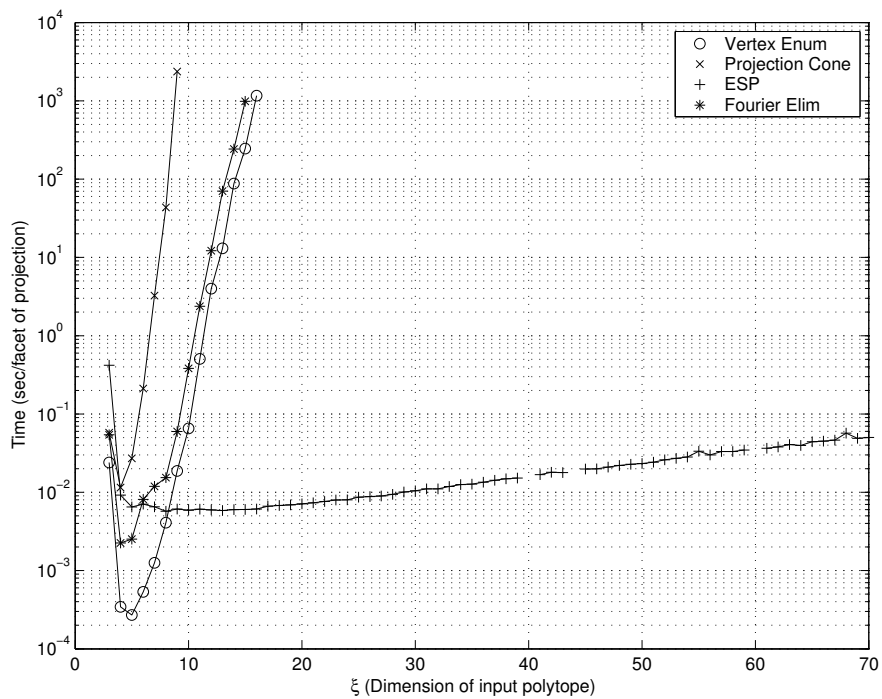
(b)  $\mathbb{R}^{\xi} \rightarrow \mathbb{R}^4$ ;  $3\xi =$  number of facets in  $P$

Figure 6.1: Comparative Simulation Results for Randomly Generated Polytopes (a-b)

## 6.2 FEASIBILITY



(c)  $\mathbb{R}^\xi \rightarrow \mathbb{R}^4$ ;  $P$  is a rotated hypercube



(d)  $\mathbb{R}^\xi \rightarrow \mathbb{R}^2$ ;  $3\xi =$  number of facets in  $P$

Figure 6.1: Comparative Simulation Results for Randomly Generated Polytopes (c-d)

following constraints:

$$\begin{aligned}
x_0 &= x \\
x_{i+1} &= Ax_i + Bu_i, \quad i = 0, \dots, N-1 \\
x_i &\in \mathcal{X}, \quad i = 1, \dots, N-1 \\
x_N &\in \mathcal{X}_F, \\
u_i &\in \mathcal{U}, \quad i = 0, \dots, N-1,
\end{aligned} \tag{6.1}$$

where  $x_i$  and  $u_i$  are future states and inputs respectively, which are constrained to be in the polytopes  $\mathcal{X}$  and  $\mathcal{U}$ , with the state at the end of the horizon  $N$  required to lie in the terminal set  $\mathcal{X}_F$ .

The feasible set is defined as all initial states for which there exists a sequence of inputs and states that satisfy the constraints in (6.1). For convenience, we define the vectorised sets  $X \triangleq [x_1^T \ \dots \ x_N^T]^T$  and  $U \triangleq [u_0^T \ \dots \ u_{N-1}^T]$  and the matrices  $\mathcal{A} \triangleq I_N \otimes [A \ -I_n]$ ,  $\mathcal{A}_x \triangleq \mathcal{A}_{*,\{1,\dots,n\}}$ ,  $\mathcal{A}_X \triangleq \mathcal{A}_{*,\{n+1,\dots,nN\}}$  and  $\mathcal{B} \triangleq I_{N-1} \otimes B$ , where  $I_j \in \mathbb{R}^{j \times j}$  is the identity and  $\otimes$  is the Kronecker product. The feasible set can now be written as the following projection:

$$\mathbb{X}_F \triangleq \pi_x \left\{ (x, X, U) \left| \begin{array}{l} \mathcal{A}_x x + \mathcal{A}_X X + \mathcal{B}U = 0, \\ X \in \mathcal{X}^{N-1} \times \mathcal{X}_F, \\ U \in \mathcal{U}^{N-1} \end{array} \right. \right\} \tag{6.2}$$

$$= \pi_x \left\{ (x, X, U) \left| \begin{array}{l} -\mathcal{A}_X^{-1}(\mathcal{A}_x x + \mathcal{B}U) \in \mathcal{X}^{N-1} \times \mathcal{X}_F, \\ U \in \mathcal{U}^{N-1} \end{array} \right. \right\} \tag{6.3}$$

In this thesis we are primarily interested in the computational complexity of calculating the feasible set  $\mathbb{X}_F$  and therefore, we here present some illustrative examples. The first is a toy example that is used in many papers to illustrate such computations as it can be computed quickly using any approach and, being two-dimensional, can be drawn on paper. The second and third examples demonstrate how quickly the computation time of existing methods grows with problem size, while ESP is still fast enough to be practical.

### 6.2.1 Double Integrator

Consider the single input, two-state double integrator:

$$x(k+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u(k)$$

## 6.2 FEASIBILITY

---

The following input and state constraints must be met at each point in time:

$$\begin{aligned} -1 &\leq u(k) \leq 1, & \forall k \geq 0 \\ \begin{pmatrix} -5 \\ -5 \end{pmatrix} &\leq x(k) \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix}, & \forall k \geq 1 \end{aligned}$$

If a horizon of  $N = 2$  is assumed, then the projection (6.2) is:

$$\begin{aligned} \mathbb{X}_F &= \pi_x \left\{ (x, U) \mid - \begin{pmatrix} 5 \\ 5 \\ 1 \\ 1 \end{pmatrix} \leq \begin{bmatrix} A & B & 0 \\ A^2 & AB & B \\ 0 & I_m & 0 \\ 0 & 0 & I_m \end{bmatrix} \begin{pmatrix} x \\ U \end{pmatrix} \leq \begin{pmatrix} 5 \\ 5 \\ 1 \\ 1 \end{pmatrix} \right\} \\ &= \pi_x \left\{ (x, U) \mid - \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 2 \\ 2 \end{pmatrix} \leq \begin{bmatrix} 2 & 2 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 2 & 4 & 3 & 2 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{pmatrix} x \\ U \end{pmatrix} \leq \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 2 \\ 2 \end{pmatrix} \right\} \end{aligned}$$

This projection from  $\mathbb{R}^4$  to  $\mathbb{R}^2$  is small enough to be computed very quickly by any available method:

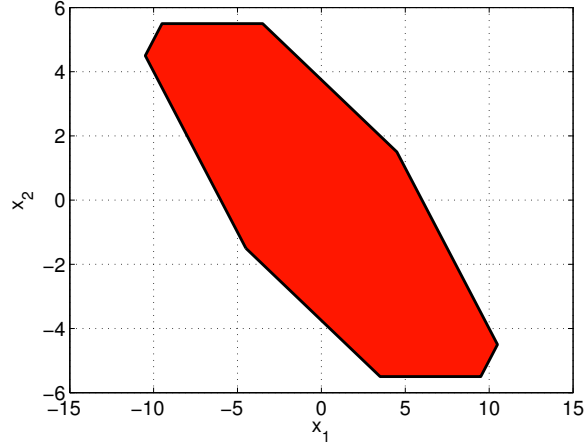
$$\mathbb{X}_F = \left\{ x \mid - \begin{pmatrix} 6 \\ 15 \\ 15 \\ 11 \end{pmatrix} \leq \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 2 & 4 \\ 0 & 2 \end{bmatrix} x \leq \begin{pmatrix} 6 \\ 15 \\ 15 \\ 11 \end{pmatrix} \right\}$$

This feasible set is shown as Figure 6.2

### 6.2.2 Random Three-Dimensional System

We now consider a randomly generated three-dimensional system and compare the computation time of the various projection methods for this problem. The following system was randomly selected using MATLAB's `drss` function:

$$x(k+1) = \begin{bmatrix} -0.0489 & 0.4040 & 0.2306 \\ 0.0682 & 0.3540 & -0.4081 \\ -0.4602 & -0.0895 & -0.0368 \end{bmatrix} x + \begin{bmatrix} -0.4326 & 0.2877 \\ 0 & -1.1465 \\ 0.1253 & 1.1909 \end{bmatrix} u \quad (6.4)$$


 Figure 6.2: Feasible Region of the Double Integrator for  $N = 2$ 

subject to the stage constraints

$$\begin{aligned}
 - \begin{bmatrix} 1 \\ 1 \end{bmatrix} &\leq u(k) \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}, & \quad \forall k \geq 0 \\
 - \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix} &\leq x(k) \leq \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}, & \quad \forall k \geq 1
 \end{aligned} \tag{6.5}$$

As is common, the terminal constraint  $\mathcal{X}_F$  is taken to be the minimally invariant set generated using a linear LQG controller with identity state and cost weights (see, for example [RKKM05] for details on minimally invariant sets and their uses). The resulting set  $\mathcal{X}_F$  contains 30 constraints.

The feasibility projection is from a polytope in  $\mathbb{R}^{13}$  containing 80 constraints, of which 30 are from the terminal set  $\mathcal{X}_F$  and the remainder are the input and state constraints (6.5). The resulting set  $\mathbb{X}_F$  is in  $\mathbb{R}^3$  and consists of only 16 halfspaces. While this would seem to be a simple problem, the upper and lower bounds on the variables in (6.5) result in a polytope that is related to a hypercube and therefore has a very large number of vertices: 166,276. As a result, most projection algorithms take a very long time to compute the feasible set. A comparison is shown in Table 6.1 and the feasible region is shown in Figure 6.3.

**Remark 6.3.** *Constraints of form (6.5) are very common in control and lead to a hypercube in the states and inputs. Intersecting this hypercube with the subspace (6.4), which forms the linear dynamics of the system may increase or decrease the number of vertices. While a hypercube has an exponential number of vertices, experience has shown that it is common for the constraint polytope to have many more than this. As was seen in the previous paragraph,*



## 6.2 FEASIBILITY

---

*the polytope has 166,276 vertices while a 13-dimensional hypercube has only 8,192.*

As in Example 6.2.1, four methods were compared, but now that the polytopes have structure, there is a significant difference in computation time depending on the vertex enumeration algorithm used. The following codes were tried: qhull [BDH96], porta [CL97], pd [BFM98a], CDD [FP96] and lrs [Avi00], but only CDD, qhull and lrs are reported as numerical errors caused the rest to fail without returning a result. In Table 6.1, the method labeled VpH refers to first enumerating the vertices of the original polytope, projecting the vertices and then enumerating the facets of the projection.

It should be noted that the performance of both Fourier elimination and the double description method (CDD) are extremely dependant on the order in which the variables are eliminated. Various orderings were tried and while, for this example, a significant difference was not seen in CDD, Fourier elimination varied between 30 seconds and almost 6 hours. It is not known if 30 seconds is the optimal ordering, or if another exists which could solve this problem faster.

Method	Time (secs)
ESP	0.22
Fourier Elimination	30.2 – 20,739.0
VpH	
CDD	4,186.7
lrs	455.51
qhull	391.0
Projection Cone	
CDD	9,327.7
lrs	> 12 hours
qhull	> 2GB RAM

Table 6.1: Comparison of Projection Methods for the Calculation of the Feasible Region for Example 6.2.2

### 6.2.3 Large Random System

In the final feasibility example we examine the performance of ESP on a larger problem. Comparative results cannot be given as none of the existing methods are able to compute the required projection.

Consider the following system with four states and ten inputs and a prediction horizon of  $N = 10$ :

$$x(k+1) = Ax(k) + Bu(k)$$

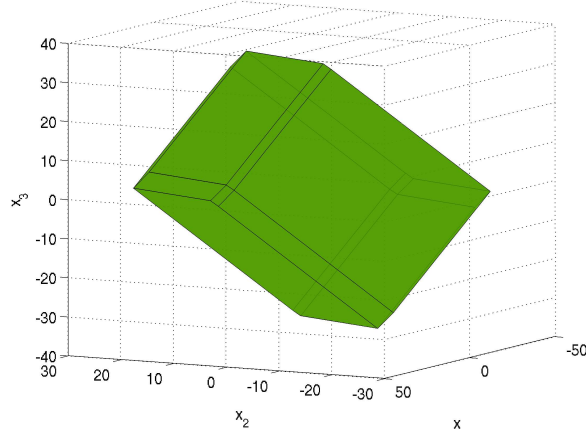


Figure 6.3: Feasible Region for Example 6.2.2

where

$$A = \begin{bmatrix} 0.1867 & 0.1104 & 0.4882 & 0.0127 \\ 0.1104 & -0.4556 & -0.1107 & 0.2971 \\ 0.4882 & -0.1107 & 0.2308 & 0.2476 \\ 0.0127 & 0.2971 & 0.2476 & 0.5222 \end{bmatrix}$$

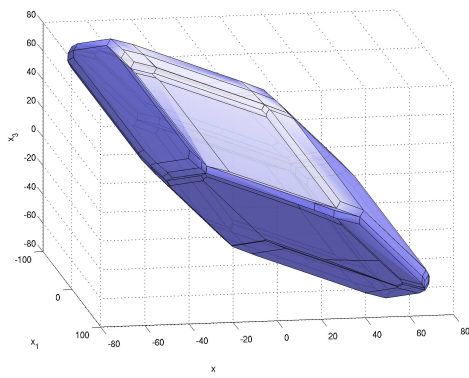
$$B = \begin{bmatrix} -0.4326 & 0 & 0 & 0 & 1.0668 & 0.2944 & -0.6918 & -1.4410 & 0 & 1.1908 \\ 0 & 1.1909 & 0.1746 & 2.1832 & 0.0593 & -1.3362 & 0 & 0 & 0.7119 & -1.2025 \\ 0.1253 & 1.1892 & -0.1867 & -0.1364 & -0.0956 & 0.7143 & 0 & 0 & 1.2902 & -0.0198 \\ 0.2877 & -0.0376 & 0.7258 & 0.1139 & -0.8323 & 1.6236 & -1.5937 & 0.6900 & 0.6686 & -0.1567 \end{bmatrix}$$

The computation of the feasible set required a projection of a 104-dimensional polytope with 208 constraints to  $\mathbb{R}^4$ . This projection was computed using ESP and the resulting feasible set contains 542 constraints. As the feasible region  $\mathbb{X}_F$  is four dimensional it cannot be drawn directly, but the projections onto each of the sets of three axes is shown as Figure 6.4.

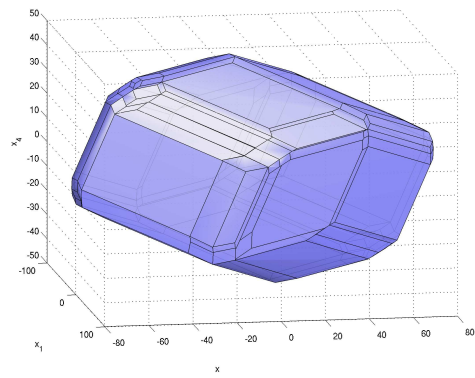
The calculation of a feasible region tends to be very degenerate if the number of inputs exceeds the number of states. As this is clearly the case for this example, a comparison of the degeneracy handling methods for ESP described in Sections 4.3 and 5.1 is in order. For this problem it was found that the vast majority of the facets were degenerate and that the recursive (Section 4.3) method took 1865.4 seconds, while the perturbation method (Section 5.1) took 249.4 seconds, spending fully 87% of its effort dealing with degeneracy. Computational experience has shown that for control problems the degeneracy handling method of Section 5.1 is often better both in speed and in numerical stability. However, as demonstrated in Figure 5.3, this is not a general rule. Despite the loss of the output sensitivity property of the algorithm for these degenerate projections, ESP is clearly drastically superior for this type of

## 6.2 FEASIBILITY

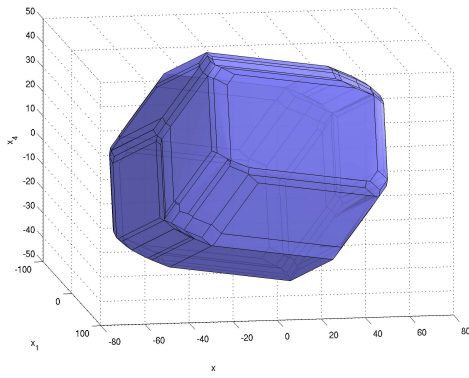
---



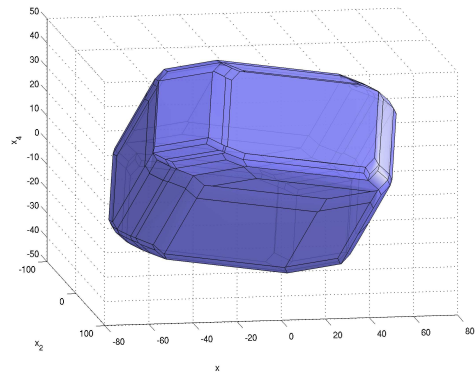
(a) Projection onto  $x_2, x_1, x_3$



(b) Projection onto  $x_2, x_1, x_4$



(c) Projection onto  $x_3, x_1, x_4$



(d) Projection onto  $x_3, x_2, x_4$

Figure 6.4: Feasible Set  $X_F$  of Example 6.2.3

problem.

It should be noted that while the computation of the feasible set in this example took only 3 minutes, it is at the limits of the current implementation of ESP. The reason for this is the numerical stability of the implementation in MATLAB, which does all calculations in floating-point arithmetic. As many geometers have found, the round-off errors involved put a sharp limit on any algorithm not using exact arithmetic. Future versions of ESP will be implemented using rational arithmetic in order to remove all problems with these errors.

## Part II

# Parametric Linear Programming



## Introduction

In this part we will consider the following multi-parametric linear program (mpLP) in the (vector) parameter  $\theta \in \mathbb{R}^d$ :

$$\begin{aligned}
 f(\theta) \triangleq \underset{x}{\text{minimise}} \quad & (E\theta + c)^T x \\
 \text{subject to} \quad & x \in D,
 \end{aligned}
 \tag{7.1}$$

where  $D$  is a polytope. The goal in ‘solving’ an mpLP is to pre-compute the solution for every possible value of the parameter.

**Remark 7.1.** *Note that we here consider only parameters in the cost or in the polytope  $D$ , but not both. There are two reasons for this: First, the problems of interest to control have this structure and second, as was shown in [SKJ<sup>+</sup>04] if there are parameters in both the cost and the polyhedron  $D$ , then the proofs of continuity, completeness and correctness of the algorithms in this part are invalid.*

The single-parameter linear parametric program has been studied for decades in various fields [Gal95]. The ability to determine what happens to the optimal solution if a parameter is varied is essential whenever there are uncertain parameters. While the single-parametric linear program has been thoroughly investigated, there has been less research into the *multi-parametric* linear program. There are two contributing factors: First, the primary use of parametric programming would appear to be as an aid to decision making. In this role, it is primarily a visualisation tool and as such one must be able to plot the solution on a 2D piece of paper. Second, as will be seen in this part, the complexity of the solution to an mpLP grows extremely quickly with the dimension of the parameter and many problems of interest outside control have an extremely large number of parameters. As such, analysing them one at a time seems to be the best approach.

In recent years there has been a surge of interest in the control community in multi-parametric programming. It has been determined [BBM00] that certain constrained, finite-horizon optimal control problems, commonly referred to as model predictive controllers (MPC) can be posed as mpLPs or mpQPs (multi-parametric quadratic programs) with the measured state as the parameter. The benefits of this are two-fold: First, before this point, the relationship between the measured state of a system and the input of the MPC controller was obscured, as the link between the two was an online optimisation. However, it can now be said with certainty that an MPC controller is in fact a piecewise linear controller, a very common and well-understood type of control. Second, one of the primary limitations of MPC, which has prevented its use in many areas, is the amount of time taken to compute the control action, as this requires the solution to a linear or quadratic program. By pre-computing the control for every possible state, this requirement of online optimisation can be removed and as a result, MPC can now be applied to a much wider range of systems, in which a high sampling rate is required.

This part will investigate the structure of the solution to multi-parametric linear programs. It is known [BBM00] that the solution to an mpLP is piecewise-affine and is defined over a union of polyhedral ‘critical regions’. We will show that if the problem is non-degenerate, then the union of these critical regions makes up a so-called complex. This property allows the proof that the solution methods proposed in Section 9.4 are correct and complete in the non-degenerate case. Furthermore, this property will enable the ability to search the solution in logarithmic time in Part IV. Clearly, it would be desirable if this property is maintained when the problem is degenerate and in Section 9.2 we do just this, introducing an algebraic perturbation of the data that will guarantee non-degeneracy for all problems. We will also prove that this perturbation ensures that the dual optimiser of (7.1) is continuous, a critical requirement for control, where a discontinuous optimiser will cause chattering. (Note that the parameter enters only in the constraints for the dual of (7.1)). Four methods will be proposed for the enumeration, each suited to a different type or size of problem. Finally, an analysis of the complexity of the algorithms will be reported and it will be seen that, in the absence of degeneracy, the enumeration methods are output sensitive.

## 7.1 Related Work

We briefly review the salient properties of existing proposals for computing multi-parametric linear programs. There have been only a few methods published:



## 7.1 RELATED WORK

---

### Full enumeration

The simplest approach to multiparametric programming is to enumerate all bases of the constraint polytope  $D$ . Each basis then needs to be tested to determine if there exists a parameter  $\theta$  satisfying the optimality conditions at that vertex. This approach is clearly not output sensitive as there can be exponentially more vertices in the polytope than there are critical regions. While there are bound to be problems with a particular structure that make this scheme efficient, in general it is a very costly approach.

### Region Complement Method (RCM)

With [BBM00], Borrelli sparked a large interest in mpLPs and mpQPs in recent years. There has been a significant amount published since the first paper in 2000, but the main references are [BBM03] and [BBM00]. An implementation of this approach has been made available as part of the Hybrid Toolbox [Bem03].

This approach stores two lists of non-overlapping polytopes, whose union covers the parameter space. One is a list of discovered critical regions, and the other is a list of polytopes whose union makes up the complement of the discovered regions. At each iteration of the algorithm, a polytope is selected from the list of unexplored polytopes and an interior point is computed. The critical region containing this point is calculated and added to the first list. Its complement is taken and intersected with the known regions, before being added to the list of unexplored space. Iterations continue until the list of unexplored regions is empty.

The algorithm has the important property that its solution is guaranteed to cover the entire feasible region, whether the problem is degenerate or not. However, if there is degeneracy, then it is possible for the primal optimiser (control input) to be discontinuous and therefore for chattering to occur. Furthermore, degeneracy may cause artificial cuts to be made in the parameter space and therefore a very large number of critical regions to be returned.

The size of the list of unexplored space is not a function of the number of critical regions. It follows that the algorithm is not output sensitive and computational experience has shown that it is not generally as efficient for control problems as other approaches.

### Facet Traversal Method (FTM)

The method reported in [Bao02, GBTM04] takes a distinctly different approach to computing the solution. An implementation of this algorithm is available in the multi-parametric toolbox (MPT) [KGBM04].

This solver is similar to the basis enumeration method that will be introduced in Section 9.4.1 and if there is no degeneracy, the algorithms will return the same result. However,

there are several key differences.

First, adjacent regions are computed by finding a point in the interior of a given facet using an LP. A small amount is added to this point in the direction normal to the facet and then a high-dimensional LP is used to compute the active constraints in the adjacent region. There are three differences between this and what is proposed in this thesis. First, the computation of an explicit point interior to the facet both requires an additional LP and can cause numerical error. Second, moving the given point in the direction normal to the facet requires one to choose a stepsize and if this stepsize is too large then small regions will be missed; too small and numerical error will interfere with the algorithm. The difficulty comes in that it is impossible to know if a stepsize is too big or too small before the algorithm is run. Finally, the calculation of the active constraints in the adjacent region requires a full LP in high dimension to be solved, rather than the LP of Theorem 9.11, which requires exactly one pivot in the absence of degeneracy. The resulting difference in the number of high-dimensional pivots will be seen in simulation in Section 10.

This method was primarily developed for solving multi-parametric *quadratic* programs, in which requiring the cost to be positive definite will guarantee a unique and continuous primal optimiser. As a result the degeneracy handling for linear problems has not yet been fully investigated. While they have not yet been published, there are ad-hoc approaches implemented in the MPT code: if dual-degeneracy is detected then a small random perturbation is made to the primal cost vector. As a different random perturbation is used each time, the solution will not be a complex and the input will not be continuous if the problem is degenerate. Furthermore, as this approach requires the assumption that the critical regions form a complex, no guarantee can be made that the entire feasible region will be covered, or that a given section of the feasible region will be explored only once. This behaviour can be seen in Example 10.4.

### [Gal95]

In [Gal95] the effect of degeneracy on parametric programming was studied and the useful notion of the *degeneracy graph* was introduced. While only sketches of algorithms were proposed in the work, an appropriate implementation would provide the optimal solution for a non-degenerate problem. When the problem is degenerate, the purpose of the author of [Gal95] is rather different than the goal here. Rather than finding a single basis for each parameter  $\theta$ , the goal is to enumerate *all* bases. While this may show interesting properties of the problem, it is unnecessary for control applications. Furthermore, as seen in Figure 8.4 the number of bases that need to be enumerated increases exponentially with the degree of

## 7.2 OUTLINE

---

degeneracy, making this approach unsuited to control problems.

### [STJ05a]

The paper [STJ05a] introduces an approach to ensuring continuity of the optimiser of an mpLP. The method identifies the degenerate regions and then solves an mpQP with a positive definite cost in each. There are two reasons why one may prefer the perturbation approach proposed in this thesis. First, as it is based only on mpLPs, a separate mpQP solver is not needed and second, the critical regions will no longer form a complex and therefore the solution cannot be searched in logarithmic time as shown in Part IV.

**Remark 7.2.** *Note that the names Facet Traversal Method (FTM) and Region Complement Method (RCM) have been created by the author of this thesis so that they could be referred to conveniently in later sections. No names were given to these approaches in the papers in which they were published.*

## 7.2 Outline

Multi-parametric linear programs are obviously very closely linked to standard LPs. Chapter 8 will provide the required background on the geometric aspects of linear programming. Also presented is a modification of the well-known simplex algorithm that ensures a unique optimal basis is obtained, whether the LP is degenerate or not, which is an essential ingredient for the mpLP algorithm to follow. Chapter 9 will define what is meant by a ‘solution’ to an mpLP and investigate its structure. It will be proven that if the modified simplex method of the previous chapter is used, then the solution will have two important properties: the optimiser is continuous and the solution is defined over a complex. The first property prevents chattering in the control input and the second is a requirement for the proposed algorithms to be correct. Finally, Chapter 9 will present four enumeration methods, each with their pros and cons. Finally, examples and simulation results will be presented in Chapter 10, with a focus on control problems.



## Geometry of the Simplex Method

In this chapter we will review the needed details of linear programs (LPs), specifically: the simplex method and its geometry, degeneracy and lexicographic perturbation and single-parameter linear programming. A new enhancement of the standard simplex approach will be introduced that will guarantee a unique optimal basis; an unnecessary detail for most applications, but a useful tool for computational geometry and a requirement for the algorithms presented in this thesis. Much of the material for this section is derived from [Mur83, BT97].

In this chapter we will consider the following linear program:

$$\begin{aligned} J \triangleq \quad & \text{minimise} \quad c^T x \\ & \text{subject to} \quad x \in P \end{aligned} \tag{8.1}$$

where the polyhedron  $P \subset \mathbb{R}^n$  is defined in what is referred to as *standard form*:

$$P \triangleq \{x \mid Ax = b, x \geq 0\} \tag{8.2}$$

for some matrix  $A \in \mathbb{R}^{m \times n}$  and some vector  $b \in \mathbb{R}^m$ .

**Remark 8.1.** *Note that any polyhedron can be written in the form (8.2). See, for example, [Mur83] for details.*

We begin by motivating our interest in the vertices of the polyhedron  $P$  with the Fundamental Theorem of Linear Programming:

**Theorem 8.2 (Fundamental Theorem of Linear Programming).** *If an LP has an optimal solution, then there exists an extreme point of the feasible region that is optimal.*

## 8.1 Representation of Vertices: The Basis

We will now discuss the standard method of representing vertices in simplex-based algorithms: the basis.

Although the form of the polyhedron used here is slightly different than that of Part I, a face of the polyhedron  $P$  is still given by setting some of the inequality constraints to equality. If  $A$  is in  $\mathbb{R}^{m \times n}$  and  $N$  is a subset of  $\{1, \dots, n\}$ , then we will slightly abuse the notation introduced in Part I and define  $P_N$  to be the face:

$$\begin{aligned} P_N &\triangleq \{x \mid Ax = b, x \geq 0, x_N = 0\} \\ &= \{x \mid A_{\star, \setminus N} x_{\setminus N} = b, x_{\setminus N} \geq 0, x_N = 0\}, \end{aligned} \tag{8.3}$$

where the notation  $A_{\star, E}$  selects the columns of  $A$  in  $E$ . By definition, a face  $P_N$  is a vertex of  $P$  if and only if (8.3) contains exactly one point. Clearly, this will be the case if and only if the rank of  $A_{\star, \setminus N}$  is  $m$  (where we make the standing assumption that  $A \neq 0$ ). A standard result of linear algebra states that if the rank of  $A_{\star, \setminus N}$  is  $m$ , then there will exist a subset  $B$  of  $\{1, \dots, n\} \setminus N$  such that  $\text{rank}(A_{\star, B}) = m$  and  $|B| = m$ . Any set  $B \subseteq \{1, \dots, n\}$ , such that  $|B| = m$  is called a *basis* of  $P$  and if  $B$  also has the property that  $P_{\setminus B}$  is a vertex of  $P$ , then it is called a *feasible basis*. Clearly, for every vertex  $v$  of  $P$ , there exists a basis such that  $P_{\setminus B} = \{v\}$ .

In the remainder of this thesis, we will adopt the standard notation used in simplex-based methods as follows. If  $B$  is a basis, then  $N \triangleq \{1, \dots, n\} \setminus B$  is defined as the non-basic variables. The matrix  $A$  will often be partitioned into basic  $A_{\star, B}$  and non-basic columns  $A_{\star, N}$ , but where it is clear from the context, we will use the standard abuse of notation and refer to  $A_{\star, B}$  as  $B$  and  $A_{\star, N}$  as  $N$ . Furthermore, we will make the assumption that  $N$  is an equality set (Definition 3.9). The case where this is not true is referred to as *primal degeneracy* and will be discussed in Section 8.5.1 below.

## 8.2 Pivoting and Polyhedral Skeletons

Two vertices are called *adjacent* if there exists an edge of the polyhedron  $P$  that contains both. Note that this adjacency of vertices differs from the adjacency of facets discussed in Part I. As the dimension of an edge is one and that of a vertex is zero, we know from Theorem 3.14 that each of the equality sets of the edges of  $P$  that contain  $P_N$  are given by  $P_{N \setminus \{e\}}$  for some  $e$  in  $N$ .

If  $B$  is a feasible basis of  $P$  and we have what is called an *entering variable*  $e \in N$ , then

### 8.3 OPTIMALITY CONDITIONS

---

we have the following relations for the edges that contain the vertex  $P_N$ :

$$\begin{aligned} P_{N \setminus \{e\}} &= \{x \mid Bx_B + N_{\star, \{e\}}x_e = b, x_{N \setminus \{e\}} = 0, x_B \geq 0, x_e \geq 0\} \\ &= \{x \mid x_B = B^{-1}b - B^{-1}N_{\star, \{e\}}x_e \geq 0, x_{N \setminus \{e\}} = 0\} \end{aligned} \quad (8.4)$$

If all the rows of the vector  $B^{-1}N_{\star, \{e\}}$  are less than or equal to zero,  $x_e$  can be increased forever and (8.4) will remain non-empty. It follows that  $P_{N \setminus \{e\}}$  is a ray of  $P$  in this case. If, however, any of the rows of  $B^{-1}N_{\star, \{e\}}$  are positive, then there is clearly an upper limit to how large  $x_e$  can grow before (8.4) becomes empty. As  $x_e$  is increased away from zero, one of the inequality constraints on the basic variables in (8.4) will become an equality. The index  $l(e)$  of this constraint is called the *leaving variable* and is given by what is called the *ratio test*, which follows directly from (8.4):

$$l(e) \triangleq \operatorname{argmin}_{j \in B} \left\{ \frac{(B^{-1}b)_j}{(B^{-1}N_{\star, \{e\}})_j} \mid (B^{-1}N_{\star, \{e\}})_j \geq 0 \right\} \quad (8.5)$$

**Remark 8.3.** *Note that  $l(e)$  is not necessarily unique. We will here assume that it is and discuss the case where it is not in Section 8.5.2.*

From the above arguments, we have that  $B' \triangleq B \setminus l(e) \cup \{e\}$  is a feasible basis of the polyhedron  $P$ . The basis  $B'$  is said to be *adjacent* to the basis  $B$  and we have that their associated vertices are also adjacent. The act of choosing an entering variable  $e \in N$  and computing  $B'$  is referred to as a simplex *pivot*.

Balinski's Theorem [Bal61] states that the *skeleton* of a polyhedron, which is the graph formed by its vertices and edges, is connected and therefore, beginning from any basis we can make a series of pivots and arrive at any other basis in  $P$ .

### 8.3 Optimality Conditions

Given a vector  $x$  in the polyhedron  $P$ , a feasible direction (or tangent vector) is defined as all directions away from  $x$  that remain in  $P$ .

**Definition 8.4** ([BT97]). *Let  $x$  be an element of the polyhedron  $P \subseteq \mathbb{R}^n$ . A vector  $\gamma \in \mathbb{R}^n$  is said to be a feasible direction at  $x$  if there exists a strictly positive scalar  $\alpha$  for which  $x + \alpha\gamma \in P$ .*

The first order necessary condition for optimality states that a point  $x$  is locally optimal only if the cost increases in all feasible directions away from  $x$ . In a linear program a convex

function is being optimised over a convex set, and therefore this condition is also sufficient and implies global optimality.

**Theorem 8.5.** *Let  $x$  be an element of the polyhedron  $P \subseteq \mathbb{R}^n$ . A necessary and sufficient condition for  $x$  to be a global minimum of the linear program (8.1) is*

$$c^T \gamma \geq 0$$

for all feasible directions  $\gamma$  at  $x$ .

From Theorem 8.2, the Fundamental Theorem of Linear Programming, we know that if an optimum of LP (8.1) exists, then it will occur at a vertex. Although there may be other optimal points interior to higher dimensional faces, this allow us to focus on finding an optimal vertex and therefore we aim to compute the change in cost as we move away from a given vertex to its neighbouring vertices.

Let  $B$  be a basis of LP (8.1) and  $x$  its associated vertex. We wish to consider the cost at the point  $x + \alpha\gamma$  for all vectors  $\gamma$  such that  $x + \alpha\gamma$  remains feasible. As discussed above, all of the feasible directions that lead to another vertex of the polyhedron involve increasing one of the variables  $x_N$  from zero. To this end, we select a non-basic variable  $j \in N$  and set  $\gamma_j = 1$ , while keeping all the other non-basic variables at zero:  $\gamma_i = 0, \forall i \in N, i \neq j$ . As all points  $x + \alpha\gamma$  must remain feasible, we have that  $A(x + \alpha\gamma) = b$ , which implies that  $A\gamma = 0$ , since  $x$  is already feasible. From this we can compute:

$$\begin{aligned} 0 &= A\gamma \\ &= B\gamma_B + N\gamma_N \\ &= B\gamma_B + A_{\star,j} \\ \Leftrightarrow \gamma_B &= -B^{-1}A_{\star,j} \end{aligned} \tag{8.6}$$

It is now a simple matter to test the condition of Theorem 8.5:

$$\begin{aligned} c^T \gamma &\geq 0 && \text{for all feasible directions } \gamma \\ \Leftrightarrow c_j^T - c_B^T B^{-1} A_{\star,j} &\geq 0, && \forall j \in \{1, \dots, n\} \\ \Leftrightarrow c^T - c_B^T B^{-1} A &\geq 0 \end{aligned} \tag{8.7}$$

The quantity in (8.7) is called the *reduced cost* and is commonly denoted by  $\bar{c}^T \triangleq c^T - c_B^T B^{-1} A$ . The following Theorem is the common optimality condition used in linear programming.



## 8.4 SIMPLEX ALGORITHM

---

**Theorem 8.6.** [BT97] *If  $B$  is a feasible basis of LP (8.1), and  $x$  and  $\bar{c}$  are the vertex and associated reduced cost respectively, then  $x$  is an optimal solution if and only if  $\bar{c} \geq 0$ .*

**Remark 8.7.** *Note that the reduced cost is often defined as  $c_N^T - c_B^T B^{-1} N$ , as  $\bar{c}_B$  is clearly zero. We include the zero elements in the definition here so that the indices are preserved, i.e.  $\bar{c}_i$  is related to the  $i^{\text{th}}$  variable  $x_i$ , rather than to  $x_{N_i}$ .*

### 8.4 Simplex Algorithm

We can now outline a rudimentary version of the (primal) simplex algorithm. The input to the algorithm is any feasible basis of the polyhedron  $P$  and the output is a feasible basis that corresponds to an optimal extreme point. At each step of the algorithm, one of the non-basic variables  $e \in N$  is chosen to ‘enter the basis’, or become inactive. A pivot is performed on this entering variable, where the leaving variable is chosen via the ratio test (8.5). By only selecting entering variables whose reduced cost is negative, we can be sure to decrease the cost in each iteration until an optimal basis is reached. This simple procedure is outlined in Algorithm 8.1 below.

**Remark 8.8.** *An initial feasible basis can be computed via a linear program over a modified version of the polyhedron  $P$ . See, for example, [Mur83] for details.*

The connectivity of the polyhedral skeleton discussed above implies that we will reach the optimal basis from any initial feasible basis. See any text on linear programming for a detailed proof (e.g. [Mur83]).

There are two steps in the simplex algorithm that involve choice: choosing the entering variable (Step 3) and choosing the leaving variable in the case of a tie in (8.5) (Step 4). The rule used to make these choices drastically affects the number of pivots that the algorithm will have to make before arriving at the optimum. There has been an enormous amount of research done on finding an optimal rule in terms of the number of pivots, while still ensuring that the optimal basis will be reached. Despite this, the original, proposed by Dantzig [Dan48, Dan51], is still often the best. See [TZ91] for a partial, although extensive, survey. In the following sections we will develop a rule that is ideal for our purposes as it deals efficiently with the degeneracy that plagues geometric algorithms.

### 8.5 Degeneracy

Up until now we have implicitly assumed that the complement of each basis is an equality set, which implies that each vertex is associated with a single basis, and we have not considered

---

**Algorithm 8.1** Simplex Algorithm

---

**Input:** Feasible basis  $B$

**Output:** Optimal basis  $B^*$  or unbounded

- 1: Compute reduced cost  $\bar{c}$  for the basis  $B$  (8.7)
  - 2: **while** there exists an  $e \in N$  such that  $\bar{c}_e < 0$  **do**
  - 3:   Choose an entering variable to reduce the cost:  $\{e \in N \mid \bar{c}_e < 0\}$
  - 4:   Compute the leaving variable via the ratio test:  $l(e)$  (8.5)
  - 5:   **if**  $l(e)$  is empty **then**
  - 6:     Return *unbounded*      The optimal cost is  $-\infty$  and  $B \cup \{e\}$  defines an extreme ray
  - 7:   **end if**
  - 8:   Perform the pivot:  $B \leftarrow B \setminus l(e) \cup \{e\}$
  - 9:   Compute the reduced cost  $\bar{c}$  for the basis  $B$
  - 10: **end while**
  - 11: Return  $B^* \leftarrow B$
- 

the case where the optimiser is non-unique. The violation of these assumptions is called primal and dual degeneracy respectively and the following sections will discuss them in detail as well as introduce a minor modification of the simplex algorithm that will compute a unique optimiser in the presence of degeneracy.

### 8.5.1 Primal Degeneracy

Primal degeneracy occurs when more than  $n - m$  constraints are active at a vertex. By definition, a basis must contain exactly  $m$  inactive constraints and therefore, when primal degeneracy occurs some of the basic variables  $x_B$  will be equal to zero (i.e. active).

Consider the following example polyhedron, which is shown in Figure 8.1:

$$P \triangleq \left\{ x \in \mathbb{R}^5 \mid \begin{bmatrix} -1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \end{bmatrix} x = \begin{bmatrix} 1 \\ 0.6 \\ 1 \end{bmatrix}, x \geq 0 \right\} \quad (8.8)$$

With reference to Figure 8.1 we see that the vertex  $v_2$  is primal degenerate as all of the following three bases represent it:  $\{1, 2, 3\}$ ,  $\{1, 2, 4\}$ ,  $\{1, 2, 5\}$ .

**Remark 8.9.** Note that while two-dimensional polyhedra can only contain primal degenerate vertices if there are weakly redundant constraints as in Figure 8.1, higher dimensional polyhedra commonly have them as shown in Figure 8.2.

The main problem caused by primal degeneracy in linear programming is called *cycling*. Cycling can occur when the simplex algorithm has a choice of which basis to use in describing a vertex. For instance, in the above example if we began from basis  $\{1, 2, 3\}$  and were optimising

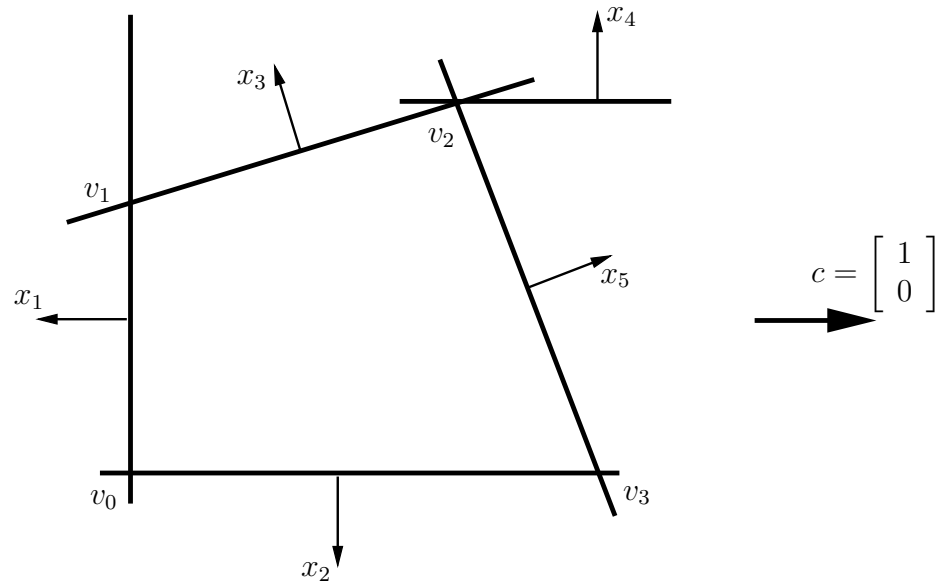


Figure 8.1: Example of Primal Degeneracy: Each of the variables  $x_i$  is shown as the orthogonal distance from hyperplane  $i$  (i.e.  $x_3$ ,  $x_4$  and  $x_5$  are slack variables)

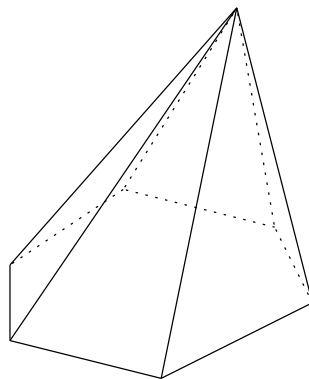


Figure 8.2: 3D Example of Primal Degeneracy

the cost function  $c^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$ , the simplex algorithm would choose to pivot to basis  $\{1, 2, 5\}$ . In the next iteration pivoting to either basis  $\{2, 4, 5\}$  or back to basis  $\{1, 2, 3\}$  is valid. If  $\{1, 2, 3\}$  is chosen then this cycle would continue indefinitely without making any progress towards the optimal.

While cycling is generally ignored in commercial codes as it is usually not problematic, there are several accepted methods in the literature to handle it [Mur83, Chap. 10]. For the purposes of geometric algorithms the main issue is the non-uniqueness of the optimal basis, rather than cycling. However, one of the approaches developed to deal with cycling will also solve this problem: lexicographic perturbation.

Primal degeneracy occurs when some of the basic variables  $x_B$  are equal to zero. Early methods of resolving primal degeneracy consisted of adding a small random vector to the right hand side of the constraints in (8.2) [Cha52]. If  $B$  is a primal-degenerate basis, then the basic variables are  $x_B = B^{-1}b$  of which some  $x_i$  are equal to zero. However, if a small random vector  $\epsilon$  was added to  $b$ , we would have  $x_B = B^{-1}b + B^{-1}\epsilon$ , and all  $x_i$  would be non-zero and therefore the basis could not be primal degenerate.

Lexicographic perturbation is a more elaborate version of this simple idea and was originally proposed in [DOW55]. Consider the perturbed constraints:

$$P^\epsilon \triangleq \{x \mid Ax = b + \epsilon, x \geq 0\},$$

where  $\epsilon^T \triangleq \begin{bmatrix} \epsilon_0 & \epsilon_0^2 & \dots & \epsilon_0^m \end{bmatrix}$  and  $\epsilon_0 > 0$  is a *sufficiently small* positive number.

The key result of this section is the following: there exists a sufficiently small  $\hat{\epsilon}$  such that for all  $\epsilon_0 < \hat{\epsilon}$ , the polyhedron  $P^\epsilon$  is not primal degenerate.

**Theorem 8.10** ([Mur83]). *Given any polyhedron  $P = \{x \mid Ax = b, x \geq 0\}$ , there exists a positive number  $\epsilon_1 > 0$ , such that whenever  $0 < \epsilon_0 < \epsilon_1$ , the following perturbed problem is primal non-degenerate:*

$$\begin{aligned} & \text{minimise} && c^T x \\ & \text{subject to} && x \in P^\epsilon, \end{aligned} \tag{8.9}$$

where  $P^\epsilon \triangleq \{x \mid Ax = b + \epsilon, x \geq 0\}$ , and  $\epsilon^T \triangleq \begin{bmatrix} \epsilon_0 & \epsilon_0^2 & \dots & \epsilon_0^m \end{bmatrix}$

Furthermore, the relationship between vertices of  $P^\epsilon$  and bases of  $P$  is given by the following theorem.

**Theorem 8.11** ([Mur83]). *If  $B$  is a feasible basis for  $P^\epsilon$  when  $\epsilon$  is sufficiently small, then  $B$  is a feasible basis for  $P$ .*

## 8.5 DEGENERACY

---

Theorems 8.10 and 8.11 allow us to solve the non-primal degenerate problem (8.9) rather than the original degenerate one. The remainder of this section discusses the mechanics of solving a linear program over the polyhedron  $P^\epsilon$  for a sufficiently small  $\epsilon$ .

Consider now a primal degenerate basis  $B$  and we have the basic vector:

$$\begin{aligned} (x_B)_i &= (B^{-1}b)_i + (B^{-1}\epsilon)_i \\ &= (B^{-1})_i b + (B^{-1})_{i,1}\epsilon_0 + (B^{-1})_{i,2}\epsilon_0^2 + \cdots + (B^{-1})_{i,m}\epsilon_0^m \end{aligned} \quad (8.10)$$

Clearly,  $(x_B)_i$  is feasible (i.e. positive) for all arbitrarily small  $\epsilon_0$  if and only if the first non-zero element of the vector  $(B^{-1})_i \begin{bmatrix} b & I_m \end{bmatrix}$  is positive. This condition is referred to as *lexico-positivity*.

**Definition 8.12 (Lexico Positive).** *If  $\gamma = (\gamma_1, \dots, \gamma_r)$  is a vector, then it is said to be lexico positive (or lex-positive) if  $\gamma \neq 0$  and if the first nonzero component of  $\gamma$  is positive. Lexico positivity will be denoted by the symbol  $\gamma \succ 0$ .*

*Given two vectors,  $v$  and  $u$ , we say that  $v \succ u$  if and only if  $v - u \succ 0$ . Given a set of vectors  $\{v_1, \dots, v_r\}$ , the lexicographical minimum, denoted  $\text{lexmin}$  is the element  $v_i$  such that  $v_j \succ v_i$  for all  $j \neq i$ .*

Clearly, not all feasible bases will satisfy the positivity condition of (8.10). The following definition introduces the bases of interest.

**Definition 8.13.** *A feasible basis  $B$  is called lexicographically feasible (lex-feasible) if and only if every row of the matrix*

$$\begin{bmatrix} B^{-1}b & B^{-1} \end{bmatrix}$$

*is lex-positive.*

Figure 8.3 illustrates the geometry of a lexicographic perturbation. One can see that each of the constraints in Figure 8.1 has been shifted outwards from the origin. The result is that the primal degenerate vertex  $v_2$  has been changed into the three non-degenerate vertices that correspond to the three bases that can describe it:  $v_{2a}$ ,  $v_{2b}$  and  $v_{2c}$ . Notice that only  $v_{2a}$  and  $v_{2b}$  are feasible vertices of  $P^\epsilon$ , while  $v_{2c}$  is not. This is the graphical interpretation of  $v_{2c}$  not being lex-feasible.

We saw in Figure 8.3 that the number of lex-feasible bases can be less than the feasible ones. We define the *degree of degeneracy*  $\sigma$  as the number of extra constraints that are active at a vertex i.e. there are  $n - m + \sigma$  constraints active. If a given vertex has a degree of

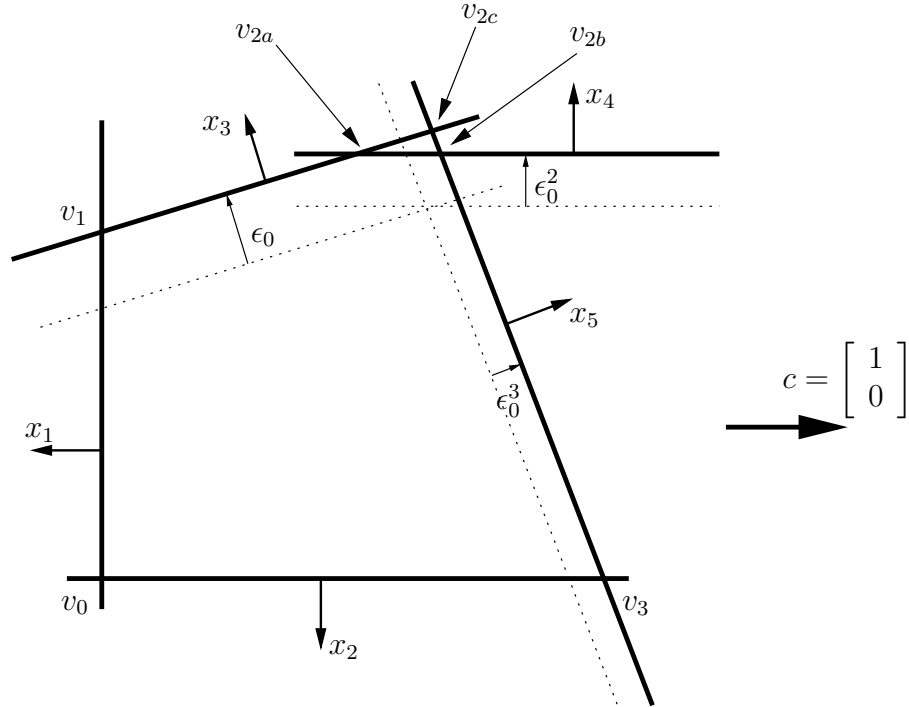


Figure 8.3: Lexicographic Perturbation  $P^\epsilon$

degeneracy of  $\sigma$ , then there are up to

$$\phi(n - m, \sigma) \triangleq \binom{n - m + \sigma}{\sigma}$$

possible feasible bases that describe the vertex [Mur83]. However, in [Arm93] it was shown that the number of lex-feasible basis is bounded by

$$\phi_{\text{lex}}(n - m, \sigma) \triangleq \binom{\lfloor \frac{n-m}{2} \rfloor + \sigma}{\sigma} + \binom{\lfloor \frac{n-m-1}{2} \rfloor + \sigma}{\sigma} - (\sigma + 1).$$

While  $\phi_{\text{lex}}$  is non-polynomial, it is still much smaller than  $\phi$  as shown in Figure 8.4.

**Remark 8.14.** *A little discussed topic in the literature is the effect of ordering on the complexity of lexicographic perturbation. In the above discussion, we have followed the standard convention of setting  $\epsilon$  equal to  $[\epsilon_0 \ \epsilon_0^2 \ \dots \ \epsilon_0^m]^T$ . However, all of the above results hold for any ordering of this vector, and the number of lex-feasible bases can change drastically with the chosen order. For example, if we were to use the ordering  $\epsilon = [\epsilon_0^2 \ \epsilon_0 \ \epsilon_0^3]^T$  in the example (8.8), then we would have only one lex-feasible basis as shown in Figure 8.5. This*

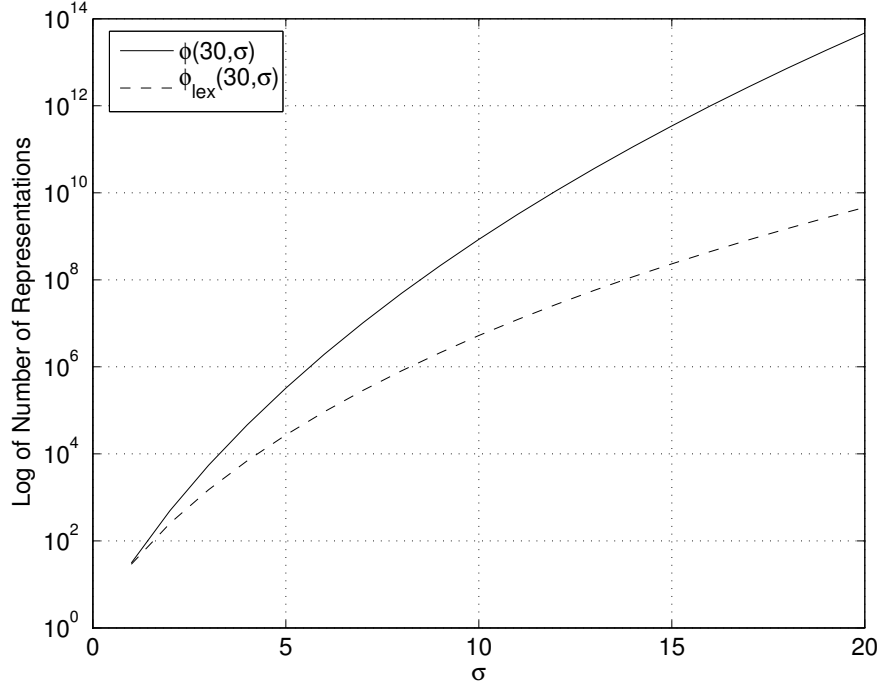


Figure 8.4: Maximum Number of Feasible and Lex-Feasible Representations of a Vertex with Degree of Degeneracy  $\sigma$

effect of ordering has largely been ignored in the linear programming literature as the computational effort of finding a good ordering would likely be large compared to the calculation of a single linear program. However, for enumeration tasks, a pre-processing stage may well be worth the effort and this would make an interesting topic for future research.

We now discuss a modification of the ratio test (8.5) that uses lexicographic perturbation. If  $B$  is a lex-feasible basis and  $e$  is the entering variable, then the lexicographic ratio test is given by:

$$l_{\text{lex}}(e) \triangleq \underset{j \in B}{\text{arglexmin}} \left\{ \frac{\left[ \begin{array}{cc} B^{-1}b & B^{-1} \end{array} \right]_j}{(B^{-1}N_{\star, \{e\}})_j} \mid (B^{-1}N_{\star, \{e\}})_j \geq 0 \right\} \quad (8.11)$$

Note that the choice made by (8.11) must be unique. A non-unique solution would exist if and only if two rows of  $\left[ \begin{array}{cc} B^{-1}b & B^{-1} \end{array} \right]$  were identical, and this clearly cannot happen as  $B$  is invertible.

The following three important theorems allow us to simply replace the ratio test (8.5) with the lexicographic ratio test (8.11) in the simplex algorithm and be sure that the optimal solution will be found and that it will not be primal degenerate. We will refer to a pivot made

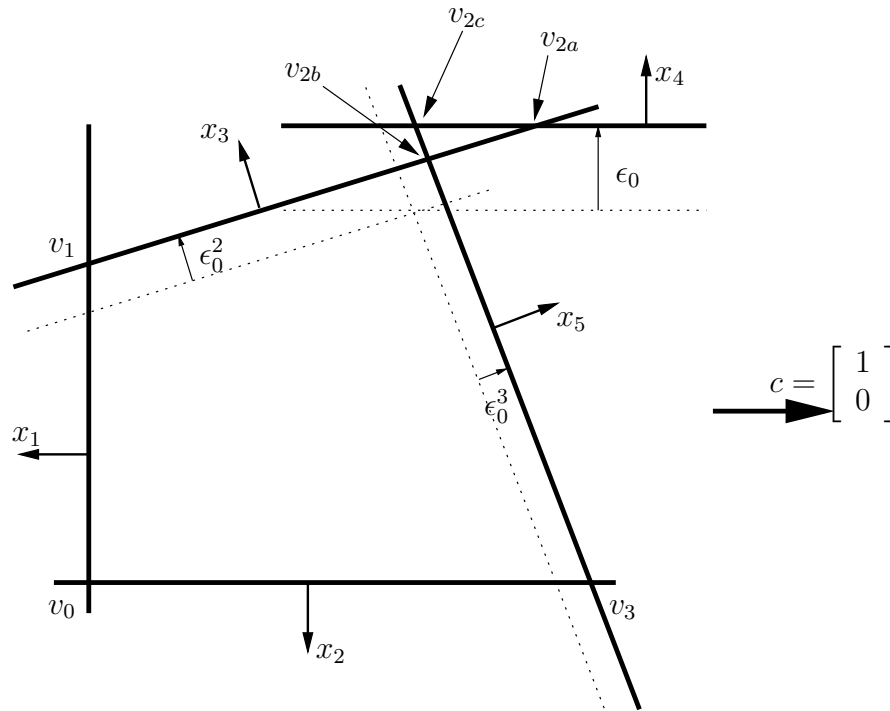


Figure 8.5: Lexicographic Perturbation: Effect of Ordering

using the lexicographic ratio test as a *lex-pivot*.

**Theorem 8.15** ([Mur83]). *If  $B$  is a lex-feasible basis of  $P$  and a lex-pivot is performed on  $B$ , the resulting basis will be lex-feasible.*

**Theorem 8.16** ([Mur83]). *For sufficiently small  $\epsilon$ , there is a one-to-one correspondence between the vertices of  $P^\epsilon$  and the lex-feasible bases of  $P$ .*

**Theorem 8.17** ([Mur83]). *Two vertices  $v_1$  and  $v_2$  of  $P$  are adjacent if and only if there are a lex-feasible basis  $B_1$  associated with  $v_1$  and a lex-feasible basis  $B_2$  associated with  $v_2$  such that one can move from  $B_1$  to  $B_2$  (and vice versa) via a single lexicographic pivot.*

**Remark 8.18.** *An initial feasible lex-positive basis can be found by solving a modified LP using a lexicographic pivoting rule. Details can be found in [Mur83].*

### 8.5.2 Dual Degeneracy

A linear program is called *dual degenerate* if there exists more than one vertex that achieves the minimal cost. Assuming that lexicographic perturbation is being used, this condition is equivalent to there being more than one optimal basis.



## 8.5 DEGENERACY

---

For many applications the existence of multiple optimisers is not a problem. However, for the purposes of the algorithm that will be presented in Chapter 9, a unique optimal basis is required. This section will describe a new method that will augment the original problem such that there is a unique optimal basis.

The first step in the algorithm is to use a standard simplex approach to find any lex-feasible and optimal basis for LP (8.1). It is well-known that the set of all optimisers will lie on a face of  $P^\epsilon$  [Zie95, Sec. 3.2]. Therefore, the second step is to identify this face using the known optimal basis. Finally, a second linear program is computed over the optimal face using a cost that is guaranteed to be non-dual degenerate.

Given a lex-optimal basis  $B$ , the following new theorem gives the face of  $P^\epsilon$  that contains all optimisers of LP (8.9).

**Theorem 8.19.** *If  $B$  is a lex-optimal basis of LP (8.1), then the set of all lex-optimisers are given by the face  $P_T$ , where  $T \triangleq \{i \mid \bar{c}_i > 0\}$ .*

*Proof.* The proof proceeds by first defining the set of all feasible directions  $\mathcal{F}$  at the vertex  $v$  associated with the basis  $B$  (i.e. the tangent set). We then parameterise each point  $x$  in  $P$  as  $x = v + f$  for some  $f \in \mathcal{F}$  and then show that the cost increases in the direction  $f$  unless  $x \in P_T$ .

Let  $F \in \mathbb{R}^{n \times n-m}$  be the matrix whose rows are defined as:

$$F_B \triangleq -B^{-1}N \qquad F_N \triangleq I \qquad (8.12)$$

From (8.6) we can see that the columns of  $F$  are precisely the directions along the edges that intersect at  $v$  and as  $v \in P^\epsilon$  is not primal-degenerate, there are exactly  $n - m$  of them and they are clearly linearly independent. The set of all feasible directions at  $v$  is then given by the set  $\mathcal{F}$ :

$$\mathcal{F} \triangleq \{\gamma \mid \exists \lambda \in \mathbb{R}^{n-m}, \gamma = F\lambda, \lambda \geq 0\}.$$

From [Zie95, Lemma 3.6] we have that the cone  $\mathcal{F}$  is a superset of  $P$ . As  $P$  is convex, it follows that for every  $x$  in  $P$  there exists a  $\gamma \in \mathcal{F}$  such that  $x = \gamma + v$ . Furthermore, as  $F$  is full rank, there exists a unique  $\lambda \geq 0 \in \mathbb{R}^{n-m}$  such that  $x = F\lambda + v$ .

The cost at  $x$  is then given by:

$$c^T x = c^T F\lambda + c^T v$$

and recalling from (8.7) that  $\bar{c}_N^T = c^T F$  and  $\bar{c}_B = 0$ :

$$c^T x = \bar{c}_N^T \lambda + c^T v$$

and the definition of  $T$  as  $\{i \mid \bar{c} > 0\}$  gives:

$$\begin{aligned} c^T x &= \bar{c}_T^T \lambda_T + \bar{c}_{\setminus T}^T \lambda_{\setminus T} + c^T v \\ &= \bar{c}_T^T \lambda_T + c^T v. \end{aligned}$$

Clearly, the cost at  $x$  is equal to the optimal cost of  $c^T v$  if and only if  $\lambda_T = 0$ , which completes our proof. □

**Corollary 8.20.** *An LP is not dual-degenerate if and only if there exists a lex-optimal basis  $B$  of LP (8.1) such that  $\bar{c} > 0$ .*

## 8.6 Simplex Algorithm with a Unique Optimal Basis

The tools are now in place to define a modified version of the simplex algorithm in which a unique optimal basis is guaranteed to be found. In order to ensure a unique optimal basis, the LP must be neither primal nor dual degenerate. In this section we will show that the following modified LP has this property:

$$\begin{aligned} &\text{minimise} && (c + \check{c}\delta)^T x \\ &\text{subject to} && x \in P^\epsilon \end{aligned} \tag{8.13}$$

for sufficiently small  $\epsilon$  and  $\delta$ , where  $\check{c}$  is any vector that is not perpendicular to any of the edges of  $P^\epsilon$ . The output in solving this LP is the optimal basis. If desired, the optimiser can then be easily derived from the basis.

**Remark 8.21.** *The simplest method to find an appropriate  $\check{c}$  is to select it randomly. This clearly does not guarantee that the assumption of non-degeneracy will be met, but one can then simply test for degeneracy at each step of the algorithm and start again with another random vector if it is detected.*

**Remark 8.22.** *Note that the cost  $\check{c}$  may also be treated as a symbolic vector of the form  $(\kappa, \kappa^2, \dots)$ , where  $\kappa$  is a small, unspecified, positive value. The reduced cost of LP (8.14) would then be computed as  $r = (\kappa, \kappa^2, \dots)F$ , where  $F$  is the matrix of feasible directions*

## 8.6 SIMPLEX ALGORITHM WITH A UNIQUE OPTIMAL BASIS

---

$F_N = I$ ,  $F_B = -B^{-1}N$ . Clearly, the  $i^{\text{th}}$  reduced cost  $r_i$  is positive if and only if the first non-zero element of  $F$  is positive. As with lexicographic perturbation of the primal, the reduced cost can clearly not be zero, and therefore not dual-degenerate. [FLN97]<sup>1</sup>

**Theorem 8.23.** *The lex-feasible basis  $B$  of  $P$  is the unique optimiser of LP (8.13) if and only if  $B$  is a lex-optimiser of LP (8.1) and of the LP:*

$$\begin{aligned} & \text{minimise} && \bar{c}^T x \\ & \text{subject to} && x \in P^\epsilon \\ & && x_T = 0 \end{aligned} \tag{8.14}$$

where  $T \triangleq \{i \mid \bar{c}_i > 0\}$

*Proof.* An optimal basis  $B$  of LP (8.13) is unique if and only if it is neither primal nor dual degenerate. Primal non-degeneracy follows directly for the lexicographically perturbed polyhedron  $P^\epsilon$  from the discussion in Section 8.5.1 and the remainder of this proof will show that  $B$  is not dual-degenerate either.

Let  $B$  be a lex-feasible basis of  $P$ . From Theorem 8.6,  $B$  is an optimal basis of LP (8.13) if and only if

$$\bar{c} + \delta \bar{c} \geq 0, \tag{8.15}$$

for  $\delta$  arbitrarily small. Hence, a necessary condition is for  $\bar{c} \geq 0$ , or in other words, for  $B$  to be a lex-optimal basis of LP (8.1).

Assume that  $B$  is a lex-optimal basis of LP (8.1) and let  $T = \{i \mid \bar{c}_i > 0\}$ . Equation 8.15 becomes:

$$\bar{c}_T + \delta \bar{c}_T \geq 0 \tag{8.16}$$

$$\bar{c}_{\setminus T} \geq 0. \tag{8.17}$$

The first condition is clearly satisfied for all sufficiently small  $\delta$ , as  $\bar{c}_T > 0$ . The second condition is satisfied if and only if  $B$  is an optimiser of LP (8.14).

Finally, we address the issue of uniqueness and show that the inequality in (8.17) is strict and therefore not dual-degenerate. Recall that  $\bar{c}_{\setminus T}^T = \bar{c}^T F_{\setminus T}$ , where the columns of  $F$  are the directions along the edges that intersect at the vertex associated with  $B$ . It follows that an element of  $\bar{c}$  can be equal to zero if and only if  $\bar{c}$  is perpendicular to one of the edges of  $P$ . By assumption, this is not true and therefore the optimiser of LP (8.13) is unique.  $\square$

---

<sup>1</sup>The author would like to thank his examiner Komei Fukuda for pointing this out.

The following two corollaries follow directly:

**Corollary 8.24.** *If  $B$  is a lex-optimal basis of LP (8.13), then it is a lex-optimal basis of LP (8.1).*

**Corollary 8.25.** *If  $B$  is a unique (non-degenerate) lex-optimal basis of LP (8.1), then it is a lex-optimal basis of LP (8.13).*

From the above theorem, it is clear that the following modified simplex algorithm (Algorithm 8.2) will compute the unique optimum of LP (8.13).

---

**Algorithm 8.2** Simplex Algorithm with a Unique Optimiser

---

**Input:** Lex-feasible basis  $B$ , costs  $c$  and  $\check{c}$

**Output:** Unique optimal basis  $B^*$  or unbounded

```

1:  $STAGE \leftarrow 1$ 
2:  $T \leftarrow \emptyset$ 
3: Goto 11 (8.7)
4: while there exists an  $e \in N$  such that  $\bar{c}_e < 0$  do
5:   Choose an entering variable to reduce the cost:  $\{e \in N \setminus T \mid \bar{c}_e < 0\}$ 
6:   Compute the leaving variable via the lex ratio test:  $l_{\text{lex}}(e)$  (8.11)
7:   if  $l_{\text{lex}}(e)$  is empty then
8:     Return unbounded
9:   end if
10:  Perform the pivot:  $B \leftarrow B \setminus l_{\text{lex}}(e) \cup \{e\}$ 
11:  Compute the reduced cost  $\bar{c}$ 
12:  if  $\bar{c} \geq 0$  then
13:    if  $STAGE = 2$  then
14:      Return  $B^* \leftarrow B$ 
15:    else
16:       $T \leftarrow \{i \mid \bar{c}_i > 0\}$ 
17:       $c \leftarrow \check{c}$ 
18:       $STAGE \leftarrow 2$ 
19:    end if
20:  end if
21: end while

```

---

One can see that the differences between Algorithm 8.2 and Algorithm 8.1 are minor. During the first stage of the algorithm any lex-optimal basis of LP (8.1) is found using exactly the same procedure as in Algorithm 8.1, apart from the replacement of the minimum ratio test at Step 6 with the lexicographic minimum ratio test. The basis is detected as being optimal for LP (8.1) at Step 12, at which point we begin to solve LP (8.14) by changing the cost and setting  $x_T$  to zero by preventing any members of  $T$  entering the basis at Step 5.

## 8.7 One-Dimensional Parametric Programming

In this section we will introduce a small extension to the well-known one-dimensional parametric linear programming problem in order to ensure uniqueness. We are interested in solving for all  $\theta \in \mathbb{R}$  the following parametric linear problem (pLP):

$$\begin{aligned} J(\theta) \triangleq \quad & \underset{x}{\text{minimise}} \quad (c + \theta e + \delta \check{c})^T x \\ & \text{subject to} \quad x \in P^\epsilon \end{aligned} \quad (8.18)$$

where  $P \triangleq \{x \mid Ax = b, x \geq 0\}$ ,  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ ,  $c \in \mathbb{R}^n$ ,  $\check{c} \in \mathbb{R}^n$  and  $e \in \mathbb{R}^n$ .

By the ‘solution’ to pLP (8.18) we mean to find for each  $\theta$ , the optimal basis  $B$ . The procedure described here consists of three steps: First, fix  $\theta$  at any value  $\theta_0$  and compute the optimal basis  $B_0$ . Second, compute the region  $\bar{\theta} < \theta < \check{\theta}$  in which  $B_0$  is the optimal basis and third, find the optimal basis  $B_1$  for  $\theta > \bar{\theta}$  and  $B_2$  for  $\theta < \check{\theta}$ . The second and third stages are then iterated for  $B_1$  and  $B_2$  until the lower bound  $\bar{\theta}$  and upper bound  $\check{\theta}$  are equal to  $-\infty$  and  $\infty$  respectively.

If  $\theta$  is fixed at some value  $\theta_0$ , then pLP (8.18) becomes a standard LP, which can be solved via the techniques discussed in Section 8.6. Assume that  $B_0$  is the optimal basis for  $\theta = \theta_0$ . Our goal is now to compute the region  $\bar{\theta} < \theta < \check{\theta}$  in which  $B_0$  is optimal. The uniqueness condition requires slightly more care than usual in pLP solvers.

The following Theorem allows the calculation of the region of optimality, or the so-called *critical region*, of a basis.

**Theorem 8.26.** *If  $B$  is a lex-feasible basis of  $P^\epsilon$  for which there exists a  $\theta$  such that  $B$  is optimal for LP (8.18), then  $B$  is optimal in the range  $\bar{\theta} < \theta < \check{\theta}$  where  $\bar{e}$  is the reduced cost of  $e$  ((8.7)) and*

$$\begin{aligned} \bar{\theta} &\triangleq \begin{cases} -\infty, & \bar{e} \geq 0 \\ \min_i \left\{ -\frac{\bar{c}_i}{\bar{e}_i} \mid \bar{e}_i < 0 \right\}, & \text{otherwise} \end{cases} \\ \check{\theta} &\triangleq \begin{cases} \infty, & \bar{e} \leq 0 \\ \max_i \left\{ -\frac{\bar{c}_i}{\bar{e}_i} \mid \bar{e}_i > 0 \right\}, & \text{otherwise} \end{cases} \end{aligned} \quad (8.19)$$

*Proof.* From Theorem 8.23, we need the basis to be optimal for both LP (8.1) and LP (8.14). We consider LP (8.1) first and recall from Theorem 8.6 that the basis  $B$  is optimal if and only if:

$$\bar{c} + \theta \bar{e} \geq 0. \quad (8.20)$$

---

## 8. GEOMETRY OF THE SIMPLEX METHOD

The range of values for which (8.20) is true is clearly given inclusively by (8.19):  $\check{\theta} \leq \theta \leq \hat{\theta}$ .

Second, we consider for which values LP (8.14) is optimal. From (8.19), we can see that in the range  $\check{\theta} < \theta < \hat{\theta}$ , the set  $T = \{i \mid \bar{c}_i + \theta \bar{e}_i > 0\}$  is fixed and therefore the optimiser of LP (8.14) is fixed. However, at the boundaries this situation changes as all indices that achieve the minimum/maximum in (8.19) are removed from the set  $T$  and therefore the optimal basis for LP (8.14) may change and therefore the critical region is open.  $\square$

We now consider the third step and look to find the basis that is optimal for  $\theta > \hat{\theta}$ . A similar procedure is used to find the optimal basis for  $\theta < \check{\theta}$  and is omitted here.

**Theorem 8.27.** *If  $B$  is a lex-optimal basis of LP (8.18) in the range  $\check{\theta} < \theta < \hat{\theta}$ , then the basis which is lex-optimal for  $\theta > \hat{\theta}$  is the lex-optimal basis of the LP:*

$$\begin{aligned} & \text{minimise} && (e + \delta \check{c})^T x \\ & \text{subject to} && x \in P^e \\ & && x_T = 0, \end{aligned} \tag{8.21}$$

where  $T$  is given by

$$T = \left\{ i \mid \bar{c}_i + \hat{\theta} \bar{e}_i > 0 \right\} \tag{8.22}$$

*Proof.* Theorem 8.23 provides the required conditions: optimality for LP (8.1) and for LP (8.14). From (8.20) we can see that a basis  $B$  is optimal for LP (8.1) at  $\theta > \hat{\theta}$  only if it is also optimal for  $\theta = \hat{\theta}$ . It follows from Theorem 8.19 that the set of optimisers at  $\theta = \hat{\theta}$  is given by the face  $P_T$ , where  $T$  is defined as

$$T = \left\{ i \mid \bar{c}_i + \hat{\theta} \bar{e}_i > 0 \right\}.$$

The goal is now to find a basis associated with a vertex of  $P_T$  such that it is optimal for LP (8.1) at  $\theta = \hat{\theta} + \alpha$ ,  $\alpha > 0$ . From Theorem 8.5, basis  $B$  has this property if and only if:

$$\bar{c} + \hat{\theta} \bar{e} + \alpha \bar{e} \geq 0$$

which can be split into

$$\bar{c}_T + \hat{\theta} \bar{e}_T + \alpha \bar{e}_T \geq 0 \tag{8.23}$$

$$\alpha \bar{e}_{\setminus T} \geq 0 \tag{8.24}$$

## 8.8 PRIMAL-DUAL PAIRS

---

Equation (8.23) is satisfied for sufficiently small  $\alpha$ , as  $\bar{c}_T + \hat{\theta} \bar{e}_T > 0$  and (8.24) is satisfied, as well as the uniqueness conditions, if and only if the basis  $B$  is optimal for LP (8.21).  $\square$

**Remark 8.28.** *Note that if neither of the bases for  $\theta < \hat{\theta}$  and  $\theta > \hat{\theta}$  are dual degenerate, then the set  $T$  will consist of all but one constraint and LP (8.21) will be solved in exactly one pivot.*

The required elements for computing the parametric linear program (8.18) are now in place and the procedure is stated in algorithmic form in Algorithm 8.3.

---

### Algorithm 8.3 One-Dimensional Parametric Linear Program with a Unique Optimiser

---

**Input:** Lex-feasible basis  $B$ , which is optimal for some  $\theta$

**Output:** Optimal basis for every  $\theta$  in the pLP (8.18)

- 1: Compute limits of optimality:  $\bar{\theta}, \hat{\theta}$  (8.19)
  - 2: Report  $B$  optimal for  $\bar{\theta} < \theta < \hat{\theta}$
  - 3: **while**  $\hat{\theta} < \infty$  **do**
  - 4:   Compute constraints  $T$  inactive at  $\theta = \hat{\theta}$  (8.22)
  - 5:   Compute optimal basis  $B'$  for  $\theta > \hat{\theta}$  LP (8.21)
  - 6:   Compute limit of optimality:  $\hat{\theta}^{B'}$  (8.19)
  - 7:   Report  $B'$  optimal for  $\hat{\theta}^B < \theta < \hat{\theta}^{B'}$
  - 8:    $B \leftarrow B'$
  - 9: **end while**
  - 10: Repeat Steps 3 through 9 for  $\theta > -\infty$
- 

## 8.8 Primal-Dual Pairs

As will be seen in the coming chapters, there are very interesting and useful interpretations of the primal-dual pairs for parametric LPs, and therefore we conclude this chapter with a very brief introduction to duality. The primal-dual pair of interest are:

$$\begin{aligned}
 J^D = \text{minimise } & c^T x & J^P = \text{maximise } & b^T y \\
 \text{subject to } & Ax = b, & \text{subject to } & A^T y \leq c \\
 & x \geq 0 & & 
 \end{aligned}
 \tag{8.25}$$

We will call the LP on the right the *primal* and that on the left, the *dual*, although this is an arbitrary choice as the dual of the dual is the primal [Mur83, Theorem 4.1]. The link between the two LPs is given by the fundamental duality theorem, sometimes called the strong duality theorem.

**Theorem 8.29 (The Fundamental Duality Theorem).** *In a primal-dual pair of LPs, if either the primal or the dual problem has an optimal feasible solution, then the other does also and the two optimal objective values are equal.*

The main implication of Theorem 8.29 for our purposes is that the optimality conditions of the primal are precisely the feasibility conditions of the dual, and *vice versa*. If  $B$  is a basis of the primal, then it is optimal if and only if:

$$\bar{c} \geq 0$$

recalling the definition of reduced cost (8.7)

$$\begin{aligned} c^T - c_B^T(B^{-1}A) &\geq 0 \\ (c_B^T B^{-1})A &\leq c^T \end{aligned}$$

and taking the transpose gives

$$A^T(B^{-T}c_B) \leq c \tag{8.26}$$

Equation (8.26) can now be written as:

$$\begin{aligned} A^T y &\leq c, \\ (A^T)_B y &= B^T y = c_B \end{aligned} \tag{8.27}$$

from which it follows that the basis  $B$  is also a basis of the dual and the primal is optimal only if the dual is feasible. A similar argument can be made for the feasibility of the primal implying optimality of the dual.

A second implication of the duality theorem is the following corollary.

**Corollary 8.30.** *Given any primal-dual pair of LPs, the following statements hold:*

- *the primal is primal degenerate if and only if the dual is dual degenerate.*
- *the primal is dual degenerate if and only if the dual is primal degenerate.*



# Parametric Linear Programming

This chapter will introduce a new approach to parametric linear programming. We will begin with a new definition for the ‘solution’ and we prove that this solution is defined over a polyhedral complex. Second, a perturbed version of the problem will be introduced that will ensure non-degeneracy of the optimiser. Finally, four algorithms for computing the solution will be presented and the benefits and limitations of each discussed.

## 9.1 Structure of the Solution

We wish to understand the structure of the solution to the following linear problem in the multi-dimensional parameter  $\theta \in \mathbb{R}^d$ :

$$\begin{aligned} f(\theta) \triangleq & \underset{y}{\text{minimise}} && b^T y \\ & \text{subject to} && (y, \theta) \in P, \end{aligned} \tag{9.1}$$

where  $P \subset \mathbb{R}^{m+d}$  is a polytope and we restrict  $\theta$  to lie in the polyhedron  $\Theta \triangleq \pi_\theta(P) \subseteq \mathbb{R}^d$ .

Our goal is to compute efficiently the optimiser of mpLP (9.1) given any  $\theta \in \Theta$ . To do this we will identify polyhedral subsets of the parameter space called critical regions, in which the optimal basis does not change. Once we have determined which region contains the parameter  $\theta$ , we can then easily compute the optimiser from the basis.

The following sections will define the critical regions formally and prove an important property that will allow the enumeration of all critical regions in a simple manner. Finally, problem (9.1) will be modified slightly to guarantee that the optimiser is unique and continuous.

### 9.1.1 Solution Complex

A key property that is needed for the enumeration algorithms in Section 9.4 is that a facet of any critical region has a full-dimensional intersection with at most one facet of another critical region. Without care, this assumption is easily violated, as shown in [SKJ<sup>+</sup>04]. It is the purpose of this section to define the critical regions in a new manner such that we are able to prove that they form what is called a polyhedral complex, which has this desired property.

**Definition 9.1.** [Grü00] *A finite family  $\mathcal{C}$  of polyhedra in  $\mathbb{R}^n$  is a complex if*

1. *Every face of a member of  $\mathcal{C}$  is itself a member of  $\mathcal{C}$*
2. *The intersection of any two members of  $\mathcal{C}$  is a face of each of them*

The definition of a critical region used here is defined via the epigraph.

**Definition 9.2.** *Let  $g : U \rightarrow \mathbb{R}$ , where  $U \subseteq \mathbb{R}^n$ . The epigraph of  $g$  is:*

$$\text{epi}(g) \triangleq \{(u, w) \mid u \in U, w \in \mathbb{R}, g(u) \leq w\}$$

The epigraph of  $f$  is then as shown in the following lemma:

**Lemma 9.3.** *The epigraph of  $f$ , as defined in (9.1), is*

$$\text{epi}(f) = \pi_{(\theta, J)} Q$$

where  $Q$  is defined as

$$Q \triangleq \{(\theta, J, y) \mid J \geq b^T y, (y, \theta) \in P\}. \tag{9.2}$$

*Proof.* From the definition of the epigraph, we have:

$$\begin{aligned} \text{epi}(f) &= \{(\theta, J) \mid \theta \in \pi_\theta P, J \in \mathbb{R}, f(\theta) \leq J\} \\ &= \{(\theta, J) \mid \exists y, (y, \theta) \in P, J \in \mathbb{R}, f(\theta) \leq J\} \end{aligned}$$

For every  $(y, \theta) \in P$ , we have that  $b^T y \geq f(\theta)$  and that there exists a  $y$  such that  $f(\theta) = b^T y$ . It follows that the condition  $f(\theta) \leq J$  is equivalent to  $b^T y \leq J$ . Therefore, the epigraph of  $f$  is given as in the statement of the Lemma.  $\square$

## 9.1 STRUCTURE OF THE SOLUTION

---

**Corollary 9.4.** *There exists a matrix  $G$  and a vector  $g$  such that  $\text{epi}(f)$  is given by the polyhedron*

$$\text{epi}(f) = \{(\theta, J) \mid \mathbf{1}J \geq G\theta + g, \theta \in \pi_\theta P\}$$

*Proof.* It is shown that  $\text{epi}(f)$  is a polyhedron in Lemma 9.3, and that it is unbounded above in  $J$ . The result follows directly.  $\square$

The complex formed by the proper faces of a polyhedron  $P$ , is denoted by  $\mathcal{B}(P)$  and is called the *boundary complex* of  $P$  [Grü00]. We define the *solution complex* of problem (9.1) to be the projection of the boundary complex of the faces of  $\text{epi}(f)$  where the minimal cost is achieved.

**Definition 9.5.** *The solution complex of  $f$  is defined as:*

$$\mathcal{S}(f) \triangleq \{\pi_\theta F \mid F \in \mathcal{B}(\text{epi}(f)) \text{ and } J = f(\theta) \text{ for all } (\theta, J) \in F\}$$

*The elements of  $\mathcal{S}(f)$  that are of dimension  $\dim(\Theta)$  are referred to as the critical regions.*

The goal of an algorithm that ‘solves’ problem (9.1) will be to compute the critical regions of  $\mathcal{S}(f)$  and the equality sets of the faces of  $P$  that are the pre-image of the critical regions. Given a  $\theta \in \Theta$ , one could then determine which critical region contains  $\theta$  and compute the optimiser from the equality set. We now prove that the name ‘solution complex’ is justified, and prove that  $\mathcal{S}(f)$  is a complex.

**Theorem 9.6.**  *$\mathcal{S}(f)$  is a complex.*

*Proof.* Every member of  $\mathcal{S}(f)$  is the projection of a polytope and is therefore a polytope itself. It follows that  $\mathcal{S}(f)$  is a finite family of polyhedra.

We first show that every face of a member of  $\mathcal{S}(f)$  is in  $\mathcal{S}(f)$ . By definition, every member of  $\mathcal{S}(f)$  is the projection of a face of  $\text{epi}(f)$ . Let  $F \in \mathcal{S}(f)$  and  $\text{epi}(f)_E$  be its preimage, where  $E$  is an equality set of  $\text{epi}(f)$ . Let  $i$  be an element in  $E$ . Describing  $\text{epi}(f)$

as in Corollary 9.4, we have:

$$\begin{aligned}
 \text{epi}(f)_E &= \left\{ (\theta, J) \left| \begin{array}{l} \mathbf{1}J = G_E\theta + g_E, \\ \mathbf{1}J \leq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \\
 &= \left\{ (\theta, J) \left| \begin{array}{l} J = G_i\theta + g_i, \\ \mathbf{1}J = G_{E \setminus \{i\}}\theta + g_{E \setminus \{i\}}, \\ \mathbf{1}J \leq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \\
 &= \left\{ (\theta, J) \left| \begin{array}{l} J = G_i\theta + g_i, \\ \mathbf{1}(G_i\theta + g_i) = G_{E \setminus \{i\}}\theta + g_{E \setminus \{i\}}, \\ \mathbf{1}(G_i\theta + g_i) \leq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \tag{9.3}
 \end{aligned}$$

From (9.3) we see that the projection of  $\text{epi}(f)_E$  is:

$$\pi_\theta \text{epi}(f)_E = \left\{ \theta \left| \begin{array}{l} \mathbf{1}(G_i\theta + g_i) = G_{E \setminus \{i\}}\theta + g_{E \setminus \{i\}}, \\ \mathbf{1}(G_i\theta + g_i) \leq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \tag{9.4}$$

From (9.3) and (9.4) it is clear that  $\text{epi}(f)_{E \cup T}$  is a face of  $\text{epi}(f)_E$  if and only if  $\pi_\theta \text{epi}(f)_{E \cup T}$  is a face of  $\pi_\theta \text{epi}(f)_E$ . Since every face of a member of  $\mathcal{B}(\text{epi}(f))$  is in  $\mathcal{B}(\text{epi}(f))$ , it follows that every face of a member of  $\mathcal{S}(f)$  is in  $\mathcal{S}(f)$ .

We now prove that the second property of complexes is satisfied: The intersection of any two members of  $\mathcal{S}(f)$  is a face of each of them.

Let  $A$  and  $B$  be equality sets of  $\text{epi}(f)$  such that  $\text{epi}(f)_A \cap \text{epi}(f)_B \neq \emptyset$ . Since boundary complexes are complexes [Grü00], there exists an equality set  $T$  of  $\text{epi}(f)$  such that  $\text{epi}(f)_T = \text{epi}(f)_A \cap \text{epi}(f)_B$  and  $\text{epi}(f)_T$  is a face of both  $\text{epi}(f)_A$  and  $\text{epi}(f)_B$ . Since  $T \supset A$  and  $T \supset B$  (Theorem 3.14), it follows that  $\pi_\theta \text{epi}(f)_T$  is a face of both  $\pi_\theta \text{epi}(f)_A$  and  $\pi_\theta \text{epi}(f)_B$  by the previous argument.

## 9.1 STRUCTURE OF THE SOLUTION

---

It remains to be shown that  $\pi_\theta \text{epi}(f)_T = \pi_\theta \text{epi}(f)_A \cap \pi_\theta \text{epi}(f)_B$ . Let  $i \in A$  and  $j \in B$ .

$$\begin{aligned}
 \pi_\theta \text{epi}(f)_T &= \pi_\theta \text{epi}(f)_A \cap \text{epi}(f)_B \\
 &= \left\{ \theta \left| \begin{array}{l} \exists J, \\ \mathbf{1}J = G_A\theta + g_A \\ \mathbf{1}J = G_B\theta + g_B \\ \mathbf{1}J \geq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \\
 &= \left\{ \theta \left| \begin{array}{l} \exists J, \\ J = G_i\theta + g_i \\ J = G_j\theta + g_j \\ \mathbf{1}J = G_A\theta + g_A \\ \mathbf{1}J = G_B\theta + g_B \\ \mathbf{1}J \geq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \\
 &= \left\{ \theta \left| \begin{array}{l} \mathbf{1}G_i\theta + g_i = G_A\theta + g_A \\ \mathbf{1}G_j\theta + g_j = G_B\theta + g_B \\ \mathbf{1}G_j\theta + g_j \geq G\theta + g \\ \mathbf{1}G_i\theta + g_i \geq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \\
 &= \left\{ \theta \left| \begin{array}{l} \mathbf{1}G_i\theta + g_i = G_A\theta + g_A \\ \mathbf{1}G_i\theta + g_i \geq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \cap \left\{ \theta \left| \begin{array}{l} \mathbf{1}G_j\theta + g_j = G_B\theta + g_B \\ \mathbf{1}G_j\theta + g_j \geq G\theta + g, \\ \theta \in \Theta \end{array} \right. \right\} \\
 &= \pi_\theta \text{epi}(f)_A \cap \pi_\theta \text{epi}(f)_B
 \end{aligned}$$

□

A graphical interpretation of the solution complex is shown in Figure 9.1. The polytope  $P$  is shown in the first two dimensions, as it is a function of  $\theta$  and  $y$  only. The unbounded polyhedron  $Q$  is defined as in Lemma 9.3 and the epigraph  $\text{epi}(f)$  is then the projection of  $Q$  onto the  $(\theta, J)$  axis. Finally, the solution complex is the projection of the boundary complex of  $\text{epi}(f)$  onto the  $\theta$  axis. A more colourful example is given as Figure 9.2, which shows a two-dimensional solution complex and three-dimensional epigraph.

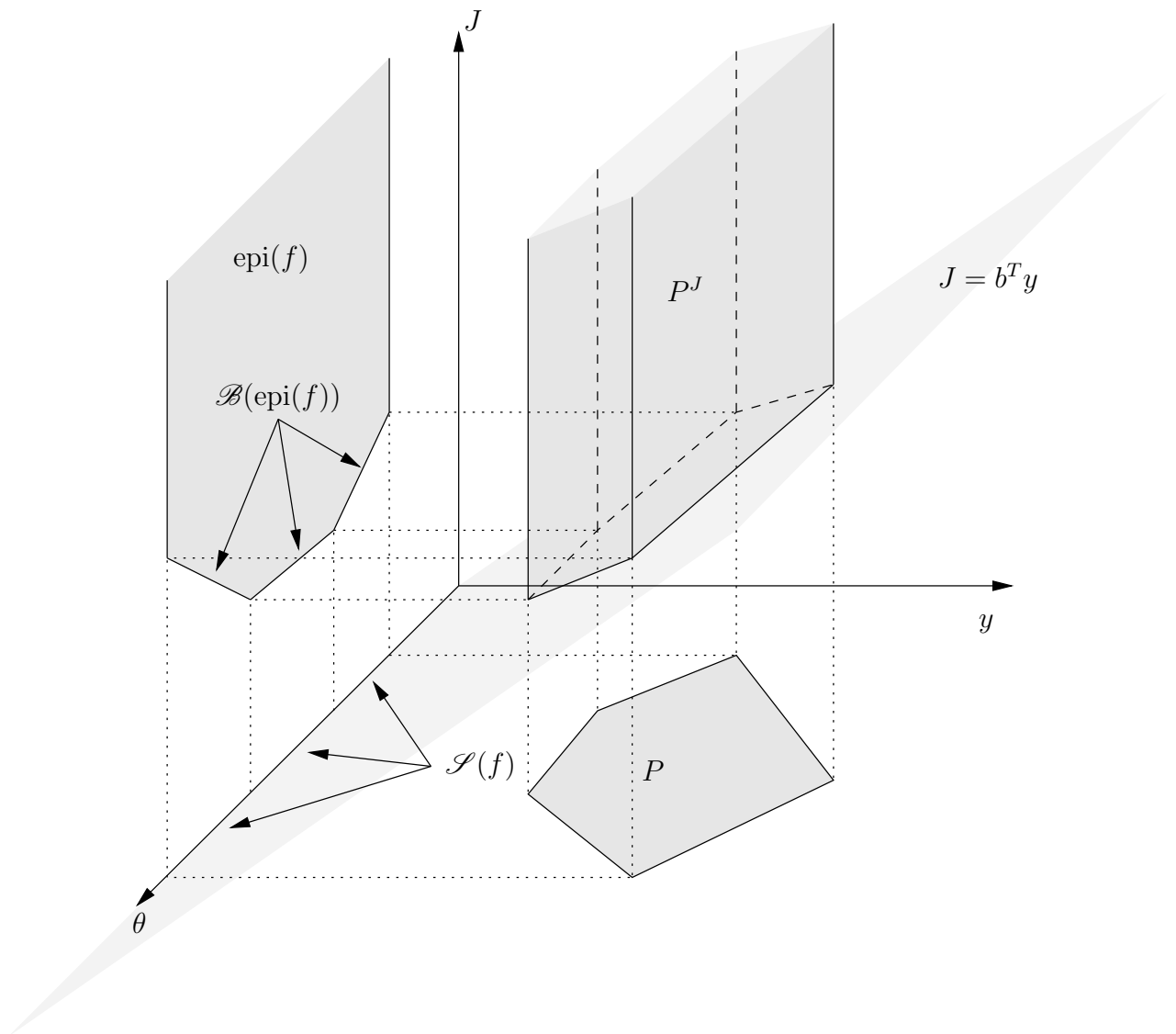


Figure 9.1: Illustration of the Geometry of the Epigraph and the Solution Complex

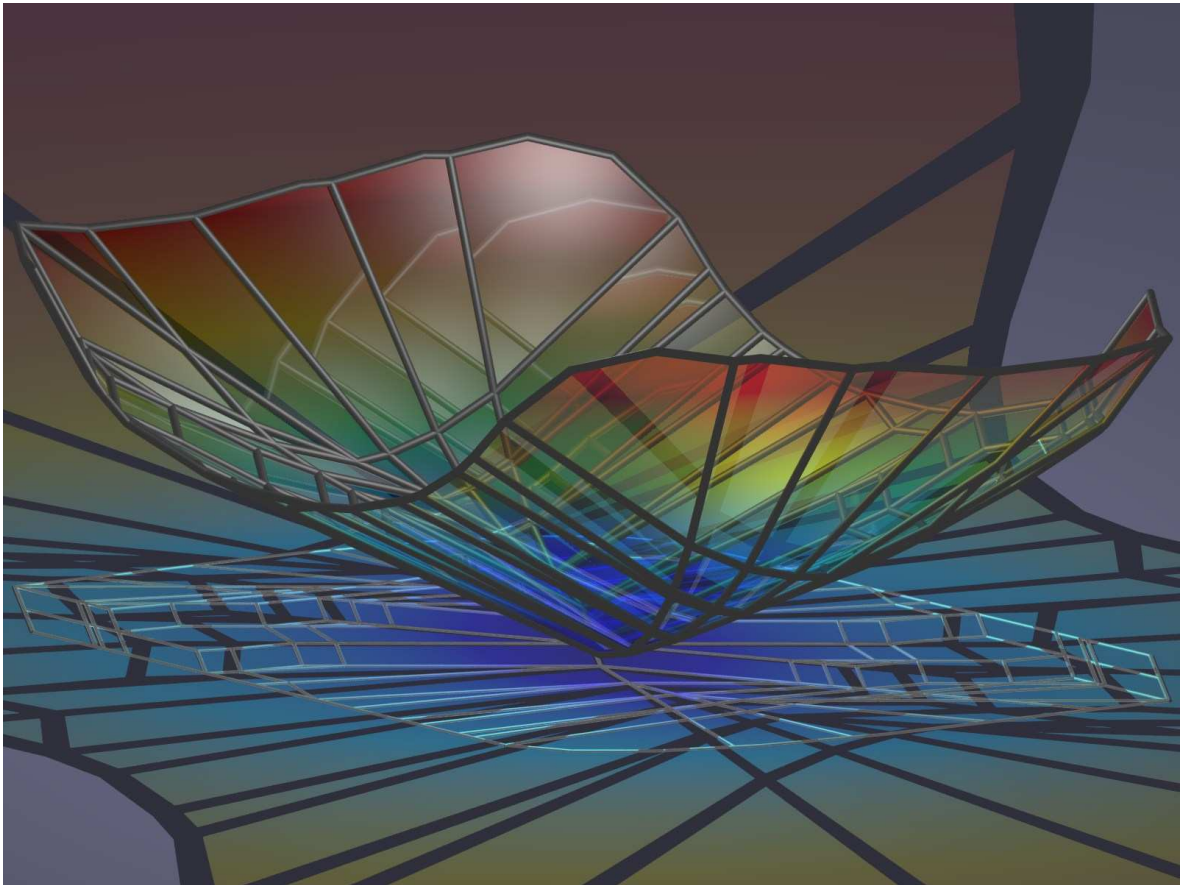


Figure 9.2: Example Epigraph and Solution Complex

## 9.2 The Optimiser

In this section we will discuss the link between the linear programming theory of Chapter 8 and the solution complex. We have thus far treated the constraint polytope  $P$  as an abstract object, without reference to how it is represented. In Part III it will be seen that there are two representations of interest, namely the primal and dual formulation. We will here show that the solution complex for both the primal and for the dual are the same. Furthermore, as all intended uses require a unique and continuous primal optimiser, we will introduce a perturbed problem as in Section 8.6, which guarantees this property.

Consider now the perturbed primal mpLP:

$$\begin{aligned} f(\theta) = \underset{y}{\text{minimise}} \quad & (b + \epsilon)^T y \\ \text{subject to} \quad & (y, \theta) \in P^\delta \end{aligned} \tag{9.5}$$

and its dual:

$$\begin{aligned} f(\theta) = \underset{x}{\text{maximise}} \quad & (c + E\theta + \delta\check{c})^T x \\ \text{subject to} \quad & x \in D^\epsilon \end{aligned} \tag{9.6}$$

The primal and dual constraint polyhedron are defined as:

$$\begin{aligned} P &\triangleq \{(y, \theta) \mid A^T y \leq c + E\theta\} \\ D &\triangleq \{x \mid Ax = b, x \geq 0\} \end{aligned}$$

and the perturbed versions as:

$$\begin{aligned} P^\delta &\triangleq \{(y, \theta) \mid A^T y \leq c + E\theta + \delta\check{c}\} \\ D^\epsilon &\triangleq \{x \mid Ax = b + \epsilon, x \geq 0\} \end{aligned}$$

where  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ ,  $c \in \mathbb{R}^n$  and  $E \in \mathbb{R}^{n \times d}$  and where the vector  $\check{c} \in \mathbb{R}^n$  is chosen such that it is not perpendicular to any of the edges of  $D$  and both  $\epsilon$ , a lexicographic perturbation, and  $\delta \in \mathbb{R}$  are sufficiently small.

**Remark 9.7.** *When discussing a basis  $B$  of the above primal-dual pair, we will use the same meaning as in previous sections. That is,  $B$  is an index set of the columns of  $A$  and is also used to mean the matrix  $A_{\star, B}$ .*

From the developments in Section 8.6, it is known that for each  $\theta$  there exists a unique



## 9.2 THE OPTIMISER

---

optimal basis  $B$  that is optimal if and only if it is feasible for both the primal and for the dual. From (8.27) and (8.3), the primal and dual optimisers are given by the faces  $P_B$  and  $D_N$  respectively:

$$P_B = \{(y, \theta) \mid y = B^{-T}(c_B + E_B\theta)\} \cap P, \quad (9.7)$$

$$D_N = \{x_B = B^{-1}b, x_N = 0\} \quad (9.8)$$

where we recall that by uniqueness, fixing  $\theta$  in (9.7) will cause  $P_B$  to be a singleton. We define the unique primal and dual optimisers as  $y^*(\theta) = \{y \mid (y, \theta) \in P_B\}$  and  $x^*(\theta) = D_N$ .

Solving the parametric program, i.e. finding for each  $\theta$  either the primal optimiser or the dual optimiser, is therefore equivalent to computing the optimal basis  $B$  for *either* the primal (9.5) *or* the dual (9.6).

We now consider the relationship between the optimal basis  $B$  and the solution complex.

**Theorem 9.8.** *If  $F$  is a member of the solution complex  $\mathcal{S}(f)$ , then there exists a unique lex-feasible basis  $B$  of  $P$  such that  $F = \pi_\theta P_B$*

*Proof.* From Theorems 8.23 and 8.10 there exists, for every  $\theta$ , a unique optimal basis  $B$  that achieves the minimal cost  $f(\theta)$ . Furthermore, this optimal basis gives the unique optimiser  $y^* = y = B^{-T}(c_B + E_B\theta)$  in (9.8). It follows that the pre-image of  $F$  is:

$$\begin{aligned} \pi_\theta^{-1}F &= \{(\theta, J, y) \mid J = b^T y^*, y = y^*, (\theta, y^*) \in P\} \\ &= \{(\theta, J, y) \mid J = b^T y, (\theta, y) \in P_B\} \end{aligned}$$

Therefore, the result follows as

$$\begin{aligned} F &= \pi_\theta \pi_\theta^{-1}F \\ &= \pi_\theta \{(\theta, J, y) \mid J = b^T y, (\theta, y) \in P_B\} \\ &= \pi_\theta P_B \end{aligned}$$

□

Given the result from Theorem 9.8, a representation of critical regions can be derived.

**Corollary 9.9.** *Let  $B$  be a unique lex-optimal basis such that  $\pi_\theta P_B$  is a critical region of  $\mathcal{S}(f)$ . Then the critical region is given by:*

$$\pi_\theta P_B = \{\theta \mid \bar{c} + \bar{E}\theta \geq 0\}, \quad (9.9)$$

where the reduced cost  $\bar{E}$  is defined columnwise as  $\bar{E} \triangleq \begin{bmatrix} \bar{E}_{*,1} & \dots & \bar{E}_{*,d} \end{bmatrix}$ .

*Proof.* The critical region is the projection of the primal optimiser as defined in (9.7):

$$\begin{aligned} \pi_\theta P_B &= \{\theta \mid \exists y, (\theta, y) \in P_B\} \\ &= \{\theta \mid \exists y, y = B^{-T}(c_B + E_B\theta), A^T y \leq c + E\theta\} \\ &= \{\theta \mid A^T B^{-T}(c_B + E_B\theta) \leq c + E\theta\} \\ &= \{\theta \mid c + E\theta - A^T B^{-T}(c_B + E_B\theta) \geq 0\} \\ &= \{\theta \mid c - A^T B^{-T}c_B + (E - A^T B^{-T}E_B)\theta \geq 0\} \end{aligned}$$

Recall the definition of the reduced cost (8.7) and the result follows immediately.  $\square$

The algorithms proposed in the following sections will ‘solve’ the mpLP by enumerating all lex-optimal bases  $B$  such that  $\pi_\theta P_B$  is a critical region. However, the optimal bases of the lower-dimensional faces of the solution complex will not be known and may well be different from those of the critical regions. This thesis has two applications for an mpLP algorithm, neither of which requires the basis of the lower-dimensional faces. The first is interested in only the dual optimisers associated with critical regions (projection) and the second is looking for any primal optimiser as long as it is continuous across the boundaries of the critical regions (closed-form MPC). We now show that the above uniqueness conditions also guarantee this continuity.

**Theorem 9.10.** *Let  $C_1$  and  $C_2$  be critical regions of the solution complex  $\mathcal{S}(f)$  for the perturbed mpLP (9.5), (9.6). If the primal optimisers are  $y_1^*(\theta)$  and  $y_2^*(\theta)$  in  $C_1$  and  $C_2$  respectively, then  $y_1^*(\theta) = y_2^*(\theta)$  for all  $\theta \in C_1 \cap C_2$ .*

*Proof.* Let  $B_1$  and  $B_2$  be the unique lex-feasible bases such that  $C_i = \pi_\theta P_{B_i}$  (Theorem 9.8). Note that it is the dual constraints that are lexicographically perturbed and therefore the dual is not primal degenerate (Theorem 8.10). Therefore, by Corollary 8.30, the primal is not dual degenerate and the primal optimiser is unique for all  $\theta$ .

While  $B_1$  and  $B_2$  may not be the unique optimisers in the region  $C_1 \cap C_2$  in the sense of Theorem 8.23, they are still lex-optimal for this region. It follows that for every  $\theta \in C_1 \cap C_2$ , the optimisers  $y_1^*(\theta)$  and  $y_2^*(\theta)$ , defined for  $B_1$  and  $B_2$  respectively as per (9.5), are optimal. As the optimiser is unique, we must have the result  $y_1^*(\theta) = y_2^*(\theta)$ .  $\square$

### 9.3 Neighbourhood Problem

The neighbours of a given critical region  $C$  are defined as those critical regions whose intersection with  $C$  is a facet of each. In this section we will discuss methods of computing adjacent critical regions.

The procedure that we outline here is essentially a generalisation of that for the one-dimensional parametric case in Section 8.7, and is of course related to the oracles of Part I. As in ESP, two oracles will be defined, the first will be called the *facet oracle*, the analogue of the ridge oracle, and the second the *adjacency oracle*.

From Theorem 9.8, it is known that each critical region is the projection of a face  $P_B$ , where  $B$  is a basis. The facets of the critical region are then the facets of  $\pi_\theta P_B$  and each facet is either on the boundary of the feasible region  $\Theta = \pi_\theta P$ , or is the intersection of  $\pi_\theta P_B$  with some other critical region. The neighbours of a critical region can then be determined in two steps: First, determine the facets of  $\pi_\theta P_B$  and second, for each facet, compute the basis that is optimal for a point just outside the facet.

The following theorem provides the tool for computing the optimal basis in a neighbouring critical region given the basis of the known region, and the description of one of its facets.

**Theorem 9.11.** *Let  $B$  be a basis such that  $C = \pi_\theta P_B$  is a critical region of the solution complex  $\mathcal{S}(f)$  to the perturbed parametric program (9.5) and let  $F = \{\theta \mid \alpha^T \theta + \beta = 0\} \cap C$  be a facet of  $C$ . Let  $B'$  be the optimal basis of the following LP:*

$$\begin{aligned} & \text{minimise} && (E\alpha + \delta\check{c})^T x \\ & \text{subject to} && x \in D^\epsilon, \\ & && x_T = 0 \end{aligned} \tag{9.10}$$

where  $T = \left\{ i \mid \begin{bmatrix} \alpha^T & \beta \end{bmatrix} \not\propto \begin{bmatrix} \bar{E}_i & \bar{c}_i \end{bmatrix} \right\} \cup \{i \mid \bar{E}_i = 0, \bar{c}_i > 0\}$ .

*If the minimum exists then the optimal basis  $B'$  defines the adjacent critical region  $C' = \pi_\theta P_{B'}$  such that  $F = C \cap C'$ , otherwise  $F$  is on the boundary of the feasible region  $\Theta$ .*

*Proof.* Let  $\theta_\circ$  be in the facet  $F$ . The goal is to find the optimal basis  $B'$  for mpLP (9.5) at all points  $\theta = \theta_\circ + \gamma\alpha$  for an arbitrarily small  $\gamma$ . It is known that there is only a single such optimal basis as Theorem 9.6 demonstrates that  $\mathcal{S}(f)$  is a complex.

The proof follows along the same lines as that for one-dimensional parametric programming (Theorem 8.27). The basis  $B'$  must satisfy the following three conditions (Theorem 8.23): First, it must be optimal for the unperturbed problem at all points  $\theta \in F$ , second, it must be optimal for the unperturbed problem for arbitrarily small  $\gamma$  and finally, it must be optimal for the perturbed problem for arbitrarily small, but strictly positive  $\gamma$ .

From Theorem 8.19, the set of dual optimisers on  $F$  for the unperturbed problem is given by  $D_T$ , where  $T$  is the set of indices of all variables whose reduced costs are strictly positive for all  $\theta$  in the interior of  $F$ . From Corollary 9.9, the critical region  $C$  is:

$$C = \pi_\theta P_B = \{ \theta \mid \bar{c} + \bar{E}\theta \geq 0 \}$$

As all critical regions are full-dimensional (Definition 9.5), there must exist a point  $\theta$  in  $C$  such that the reduced cost at  $\theta$  is strictly positive. It follows that the only optimality constraints that are active at every point in the interior of  $F$  are those that are equal (proportional) to the affine hull of  $F$ , i.e.  $\begin{bmatrix} \bar{E}_i & \bar{c}_i \end{bmatrix} \propto \begin{bmatrix} \alpha^T & \beta \end{bmatrix}$ , or are identically zero. Therefore, we have that the constraints that are strictly inactive at all points in the (relative) interior of  $F$  are given by:

$$T = \left\{ i \mid \begin{bmatrix} \alpha^T & \beta \end{bmatrix} \not\propto \begin{bmatrix} \bar{E}_i & \bar{c}_i \end{bmatrix} \right\} \cup \{ i \mid \bar{E}_i = 0, \bar{c}_i > 0 \}$$

and therefore the set of all dual optimisers is  $D_T$ .

We have now to compute the set of optimisers from the set  $D_T$  for the parameter  $\theta = \theta_o + \gamma\alpha$  for an arbitrarily small  $\gamma > 0$ . The optimality condition is given by the positivity constraint on the reduced cost:

$$\begin{aligned} \bar{c} + \bar{E}\theta &\geq 0 \\ \bar{c} + \bar{E}\theta_o + \bar{E}\alpha\gamma &\geq 0, \end{aligned}$$

which can then be split into two inequalities:

$$\bar{c}_T + \bar{E}_T\theta_o + \bar{E}_T\alpha\gamma \geq 0, \tag{9.11}$$

$$\bar{E}_{\setminus T}\alpha\gamma \geq 0. \tag{9.12}$$

By the strict positivity of  $\bar{c}_T + \bar{E}_T\theta_o$ , (9.11) is satisfied for sufficiently small  $\gamma > 0$ . Finally, (9.12) is satisfied, along with the uniqueness conditions, if and only if the optimal basis  $B'$  is the optimiser of LP (9.10).

Note that if LP (9.10) is infeasible, then there does not exist a basis  $B'$  such that (9.12) is satisfied. It follows that no optimal basis exists for the mpLP (9.5) at the point  $\theta = \theta_o + \gamma\alpha$  and therefore  $\theta$  is outside the feasible region  $\Theta$  and  $F$  is on the boundary.  $\square$

From Theorem 9.11, one can see that computing the neighbours of a given critical region can be split into two parts: First, compute the facets of the critical region and their strictly

## 9.3 NEIGHBOURHOOD PROBLEM

---

active constraints  $T$  and second, solve the linear program (9.10). These two actions will be dubbed the facet and adjacency oracles respectively.

### 9.3.1 Facet Oracle

Given an optimal basis  $B$ , Corollary 9.9 gives the description of the critical region  $\pi_\theta P_B$ :

$$\pi_\theta P_B = \{\theta \mid \bar{c} + \bar{E}\theta \geq 0\}. \quad (9.13)$$

The goal of the facet oracle is to find those inequalities of (9.13) that define facets and to compute for each of them the set of strictly active constraints  $T$  from Theorem 9.11. This problem comes down to *redundancy elimination*, or the removal of all redundant constraints from a polyhedron.

A straightforward method of checking if the  $i^{\text{th}}$  constraint of (9.13) is redundant is to compute the optimal cost of the following LP [Fuk00]:

$$\begin{aligned} J_i^* = \text{minimise} \quad & \bar{E}_i \theta \\ \text{subject to} \quad & \bar{c}_i + \bar{E}_i \theta \geq 0 \\ & \bar{c}_i + \bar{E}_i \theta \geq -1 \end{aligned} \quad (9.14)$$

The  $i^{\text{th}}$  constraint is then redundant if and only if the optimal cost satisfies  $J_i^* + \bar{c}_i \leq 0$ .

There are several heuristic methods that can improve the efficiency of the redundancy calculation, [Cla94, OSS95, Gri04, SLG<sup>+</sup>04]. These methods however, have been seen to have only a limited effectiveness for control problems and for many LP (9.14) must be computed for each inequality. Comparative simulations are given in Chapter 10.

A simple algorithm for computing the affine hulls and strictly positive constraints of all the facets of a given critical region is given as Algorithm 9.1. One can see that the complexity of this algorithm is entirely a function of Step 6, where one linear program must be solved in  $d$  dimensions and  $n - m$  constraints. As Step 6 is executed in the worst case  $n - m$  times the complexity of the facet oracle is:

$$\mathcal{T}_{\text{facet oracle}} = \mathcal{O}((n - m)LP(d, n - m))$$

### 9.3.2 Adjacency Oracle

The input to the adjacency oracle is a basis that defines a critical region, a facet of the critical region and a list of strictly active constraints on that facet. The goal of the oracle is then

---

**Algorithm 9.1** Facet Oracle

---

**Input:** Lex-optimal basis  $B$  such that  $\pi_\theta P_B$  is a critical region

**Output:** List of strictly active constraints  $T$  for each facet of the critical region  $\pi_\theta P_B$

- 1: Compute  $\bar{c}$  and  $\bar{E}$
- 2:  $ToTest \leftarrow \{i \mid \bar{E}_i \neq 0\}$
- 3: **while**  $ToTest$  is not empty **do**
- 4:   Select any member  $i$  from  $ToTest$
- 5:   Compute constraints  $Q$  equal to  $i$ :

$$Q \leftarrow \{j \mid [\bar{E}_i \quad \bar{c}_i] \propto [\bar{E}_j \quad \bar{c}_j]\}$$

- 6:   Test if  $i^{th}$  constraint is redundant:

$$\begin{aligned} J^* = \text{minimise} \quad & \bar{E}_i \theta \\ \text{subject to} \quad & \bar{c}_{\setminus Q} + \bar{E}_{\setminus Q} \theta \geq 0 \\ & \bar{c}_i + \bar{E}_i \theta \geq -1 \end{aligned} \tag{9.15}$$

- 7:   **if**  $J^* + \bar{c}_i < 0$  **then** If true, then  $i$  is irredundant
- 8:    Compute strictly active constraints:

$$T = \{j \mid [\bar{E}_i \quad \bar{c}_i] \not\propto [\bar{E}_j \quad \bar{c}_j]\} \cup \{j \mid \bar{E}_j = 0, \bar{c}_j > 0\}$$

- 9:    Report facet  $\{\theta \mid \bar{E}_i \theta + \bar{c}_i = 0\} \cap \pi_\theta P_B$  and the strictly active constraints  $T$ .
  - 10: **end if**
  - 11:  $ToTest \leftarrow ToTest \setminus Q$
  - 12: **end while**
-

## 9.4 ENUMERATION ALGORITHMS

---

to compute the unique optimal basis for the adjacent critical region, whose intersection with the given region is the given facet.

As shown in Theorem 9.11, this problem can be solved through the calculation of linear program (9.10). An algorithm table is not given for this oracle as it is straightforward.

In the worst case, the adjacency oracle consists of solving a single linear program in  $m$  dimensions with  $n$  constraints resulting in a complexity of

$$\mathcal{T}_{adj\ ocl} = \mathcal{O}(LP(m, n)).$$

However, if the basis is non-degenerate, then all but two of the constraints are fixed at zero and the LP consists of a single pivot operation. Furthermore, the dimension of the LP that must be solved is equal to the degree of degeneracy, and therefore in practice this oracle will generally take less time to solve than the facet oracle, even though in big- $\mathcal{O}$  notation it is slower.

## 9.4 Enumeration Algorithms

As discussed in the previous section, the goal of the mpLP algorithm is to enumerate all of the bases associated with critical regions of the solution complex  $\mathcal{S}(f)$ . In this section we will present four algorithms for this enumeration. In each method a different graph will be constructed whose enumeration will result in the enumeration of the critical regions of the solution complex.

The first method is a simple depth/breadth first enumeration over a graph whose vertices are the critical regions and whose edges are the facets joining two regions. The second recognises the high computational cost of the adjacency oracle and reduces the number of times it is called through a search approach much like that employed in ESP, where the vertices of the graph are the facets of the critical regions. The third method is an implementation of the reverse search algorithm, made popular by Avis and Fukuda in the 1990s, which enumerates a tree whose nodes are the critical regions. This is theoretically a very efficient algorithm but due to computational overheads does not become advantageous until the problem size is quite large. The final method is a heuristic primal-dual approach that stores both a vertex and a halfspace description of the result, which potentially increases the speed of the facet oracle drastically for smaller problems. The following sections discuss each of these methods in detail.

### 9.4.1 Basic Enumeration

In this section a simple enumeration scheme will be presented that works with the bases that define critical regions directly. We first define the graph that will be enumerated, prove that it is connected and then outline a simple algorithm for the enumeration.

**Definition 9.12.** *Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be as defined in (9.5). The basis solution graph  $\mathcal{G}^{\mathcal{B}}(f)$  is defined as the pair*

$$\mathcal{G}^{\mathcal{B}}(f) \triangleq (V(f), E(f)),$$

where the vertices  $V(f)$  are defined as the critical regions of  $\mathcal{S}(f)$  and a pair  $(v_1, v_2) \in V(f) \times V(f)$  is in  $E(f)$  if and only if  $v_1 \cap v_2$  is a facet of both  $v_1$  and of  $v_2$ .

The key property that is required for an enumeration algorithm is the connectivity of the graph  $\mathcal{G}^{\mathcal{B}}(f)$ .

**Proposition 9.13.** *The graph  $\mathcal{G}^{\mathcal{B}}(f)$  is connected.*

*Proof.* Let  $(V, E)$  be the graph whose vertices  $V$  are the facets of  $\text{epi}(f)$  and whose edges  $E$  connect two facets if and only if the intersection of the facets is a ridge. Note that  $\text{epi}(f)$  is bounded below and therefore the facets of interest are equivalent to facets on a polytope and therefore Theorem 4.1 proves that the graph  $(V, E)$  is connected. As every critical region is the projection of a facet of  $\text{epi}(f)$ , the graph  $(V, E)$  and  $\mathcal{G}^{\mathcal{B}}(f)$  are equivalent and therefore  $\mathcal{G}^{\mathcal{B}}(f)$  is connected.  $\square$

We will now introduce a simple scheme that uses the two oracles from Section 9.3 to fully enumerate all vertices of the graph  $\mathcal{G}^{\mathcal{B}}(f)$ . As the graph  $\mathcal{G}^{\mathcal{B}}(f)$  is connected (Proposition 9.13), we are free to use any basic enumeration scheme. See Algorithm 9.2 for an outline of the proposed scheme. A descriptive example of the algorithm is shown in Figure 9.3.

### Complexity

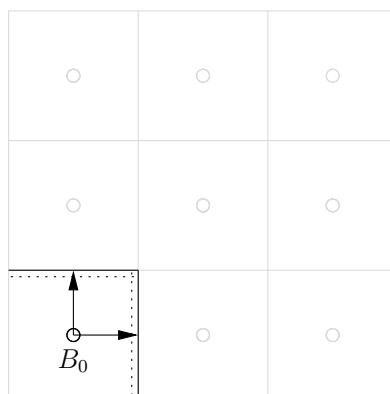
We now examine the complexity of Algorithm 9.2. First, we must consider the cost of operations on the set  $\mathcal{L}_{unexplored}$ . The properties that the storage method must have are: First, the order that elements are inserted (Steps 1 and 10) or removed (Step 4) is not important and second, insertion and removal happen infrequently compared to search (Step 9). The most common data structure that has the required properties is the AVL tree [Pfa02] for which insertion, deletion and search can all be done in  $\mathcal{O}(\log q)$ , where  $q$  is the number of elements in the tree. As Steps 9 and 10 are called once for each facet and there are a maximum of  $n - m$



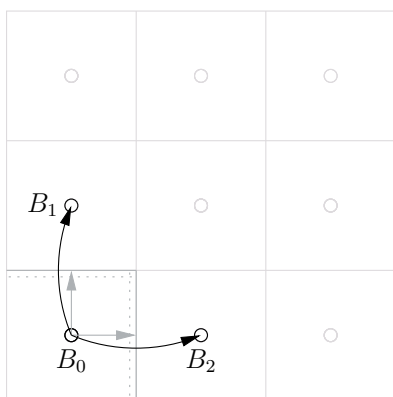
## 9.4 ENUMERATION ALGORITHMS



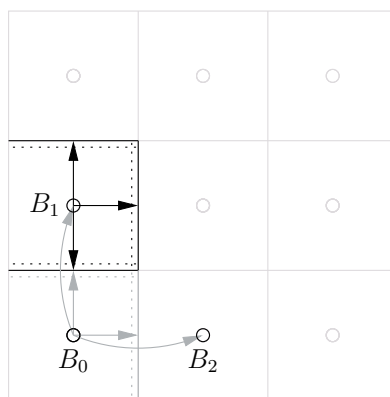
Initial basis  $B_0$



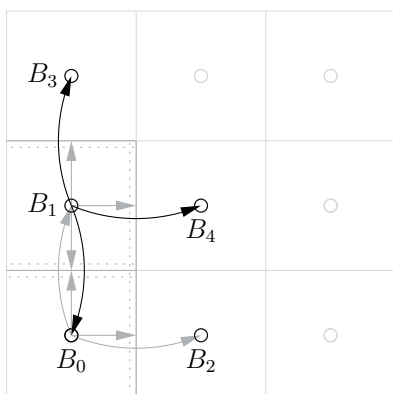
Step 6: Compute facets of  $\pi_\theta P_{B_0}$



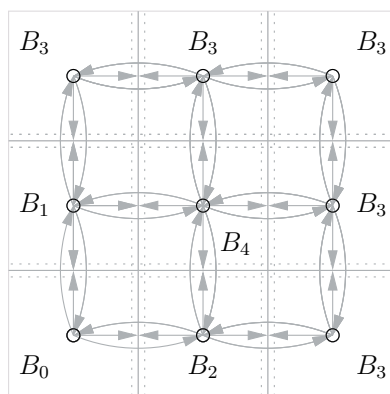
Step 8: Compute adjacent bases



Steps 4 and 6: Choose unexplored basis and compute facets



Step 8: Compute adjacent bases



Repeat

Figure 9.3: Example Enumeration using the Basic Method. The curved lines are the edges of the graph  $\mathcal{G}^{\mathcal{B}}(f)$  that were traversed and the circles are the vertices.

---

**Algorithm 9.2** Multiparametric Linear Programming: Basic Enumeration

---

**Input:** Unique lex-optimal basis  $B$  such that  $\pi_\theta P_B$  is a critical region

**Output:** All bases that define critical regions.

```

1:  $\mathcal{L}_{unexplored} \leftarrow \{B\}$  Initialise set of unexplored bases
2:  $\mathcal{L}_{discovered} \leftarrow \{B\}$  Initialise set of discovered bases
3: while  $\mathcal{L}_{unexplored}$  is not empty do
4:   Remove any basis  $B$  from  $\mathcal{L}_{unexplored}$ 
5:   Report basis  $B$ 
6:   Compute facets  $F_i$  of  $B$  Facet Oracle: Sec. 9.3.1
7:   for each facet  $F_i$  do
8:     Compute adjacent basis:  $B'$  Adjacency Oracle: Sec. 9.3.2
9:     if Adjacent basis exists and is not in  $\mathcal{L}_{discovered}$  then
10:       Insert  $B'$  into  $\mathcal{L}_{discovered}$ 
11:       Insert  $B'$  into  $\mathcal{L}_{unexplored}$ 
12:     end if
13:   end for
14: end while

```

---

facets for each critical region, the complexity is  $N_r(n - m)\mathcal{T}_{search}$ , where  $\mathcal{T}_{search} = \log N_r$ , where  $N_r$  is the number of critical regions.

For problems of reasonably small size, the time taken for these operations is very small compared to that of the two oracles. However, as the complexity is a function of the number of critical regions found their complexity will eventually outweigh that of the oracles. Having said this, in the author's experience, problems must get exceptionally large before this becomes an issue. An algorithm is presented in Section 9.4.4 that is suited to these large problems as it removes the need to make any search over the critical regions found so far, improving both the time and space complexity.

In this algorithm, the facet oracle is called exactly once for each critical region, and the adjacency oracle is called twice for each facet. If  $N_r$  is the number of critical regions, and  $N_f = \mathcal{O}(N_r(n - m))$  is the number of facets, then the total time complexity is

$$\begin{aligned}
\mathcal{T}_{basic} &= N_r \mathcal{T}_{fct\ orl} + N_f \mathcal{T}_{adj\ orl} + N_f \mathcal{T}_{search} \\
&= \mathcal{O}(N_r(n - m)(LP(d, n - m) + LP(n, m) + \log N_r))
\end{aligned} \tag{9.16}$$

The worst-case space complexity is clearly the storage of an  $m$  dimensional vector of numbers (the basis) for each critical region:

$$\mathcal{S}_{basic} = \mathcal{O}(N_r(m))$$

As the size of the output can grow very quickly, it is this storage requirement that limits the

size of problem that this algorithm can handle as the data will grow past the physical memory limitation of the computer.

### 9.4.2 Facet-Based Enumeration

In this section we introduce a variant of the algorithm discussed in the previous section, which will reduce the number of times that the adjacency oracle must be called to once per critical region, rather than twice per facet. This reduction in complexity comes at the cost of a potential increase in the storage requirements and in the resulting search and insert times over the stored data. The approach proposed is similar to that presented for the ESP algorithm in Part I.

The first step is to define the graph that is going to be enumerated. Instead of the critical regions being the vertices of the graph as in the previous section, the facets of the critical regions are taken.

**Definition 9.14.** *Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be as defined in (9.5). The facet solution graph  $\mathcal{G}^{\mathcal{F}}(f)$  is defined as the pair*

$$\mathcal{G}^{\mathcal{F}}(f) \triangleq (V(f), E(f)),$$

where the vertices  $V(f)$  are defined as the  $d - 1$  dimensional faces of  $\mathcal{S}(f)$  that are the intersection of two critical regions and a pair  $(v_1, v_2) \in V(f) \times V(f)$  is in  $E(f)$  if and only if there exists a critical region that contains both  $v_1$  and  $v_2$ .

As before, connectivity of  $\mathcal{G}^{\mathcal{F}}(f)$  is a requirement of the algorithm.

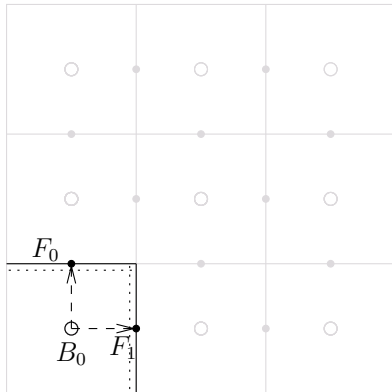
**Proposition 9.15.** *The graph  $\mathcal{G}^{\mathcal{F}}(f)$  is connected.*

*Proof.* Connectivity of the graph  $\mathcal{G}^{\mathcal{F}}(f)$  follows directly from the link between facets of the solution complex and ridges of the epigraph and the proof of the correctness of ESP (Theorem 4.1).  $\square$

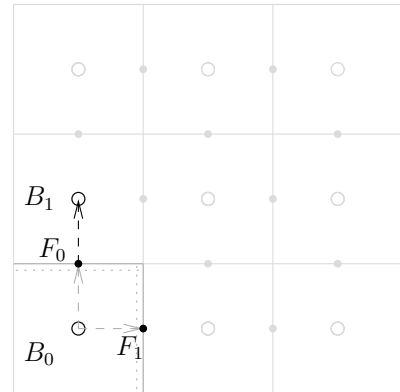
We can now describe an algorithm that uses a search strategy similar to that of ESP. This approach is presented as Algorithm 9.3.

**Remark 9.16.** *Note that the sets of strictly active constraints  $T$  for each facet are equality sets and therefore provide a unique list of integers that can be used to identify a facet, making searching and comparison in Step 13 simple and efficient.*

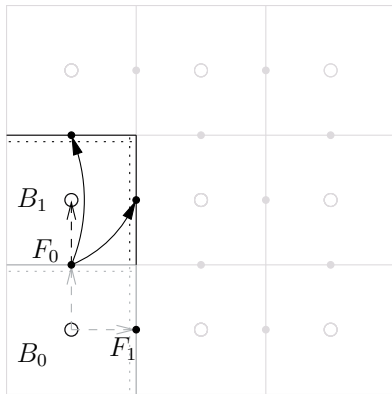
The example shown in Figure 9.3 is displayed again in Figure 9.4 using the facet-based approach. One can see that the reduction in the number of calls to the adjacency oracle is significant.



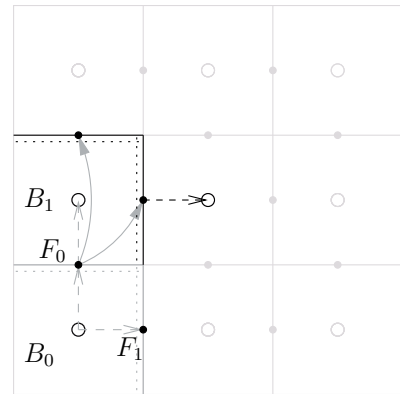
Step 8: Enumerate facets of initial basis  $B_0$ .



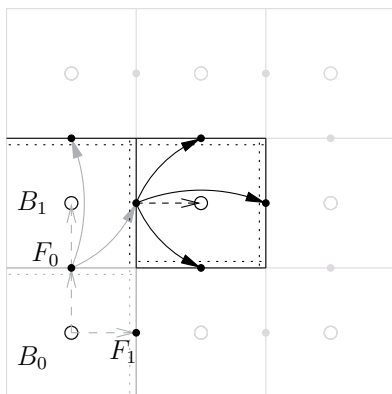
Steps 4 and 5: Choose a facet  $F_0$  and compute the adjacent basis.



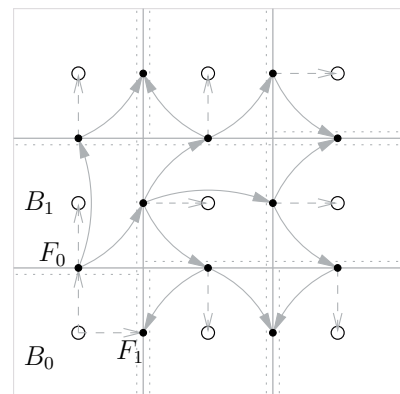
Step 8: Enumerate facets of adjacent basis  $B_1$ .



Steps 4 and 5: Choose a facet and compute the adjacent basis.



Step 8: Enumerate facets of adjacent basis.



Repeat.

Figure 9.4: Example Enumeration using the Facet-Based Method. The curved lines are the edges of the graph  $\mathcal{G}^{\mathcal{F}}(f)$  that were traversed and the black circles are the vertices.

## 9.4 ENUMERATION ALGORITHMS

---



---

### Algorithm 9.3 Multiparametric Linear Programming: Facet-Based Enumeration

---

**Input:** Unique lex-optimal basis  $B$  such that  $\pi_\theta P_B$  is a critical region

**Output:** All bases that define critical regions

```

1:  $\mathcal{L}_{unexplored} \leftarrow \emptyset$ 
2: Goto 7
3: repeat
4:   Select any element  $(F, B')$  from  $\mathcal{L}_{unexplored}$ 
5:   Compute adjacent basis  $B$  such that  $F = \pi_\theta P_B \cap \pi_\theta P_{B'}$  Sec. 9.3.2
6:   if adjacent basis  $B$  exists then
7:     Report  $B$ 
8:     Compute facets  $F_i$  of  $B$  Sec. 9.3.1
9:     for each facet  $F_i$  do
10:      if there exists an element  $(F_i, X)$  in  $\mathcal{L}_{unexplored}$  for some  $X$  then
11:        Remove  $(F_i, X)$  from  $\mathcal{L}_{unexplored}$ 
12:      else
13:        Insert  $(F_i, B)$  into  $\mathcal{L}_{unexplored}$ 
14:      end if
15:    end for
16:  else
17:    Remove  $(F, B')$  from  $\mathcal{L}_{unexplored}$ 
18:  end if
19: until  $\mathcal{L}_{unexplored}$  is empty

```

---

### Complexity

From Algorithm 9.3, one can see that the facet oracle is called the same number of times as for Algorithm 9.2, i.e. once for each critical region. The difference is that the adjacency oracle is now called only once per critical region. This reduction is due to the fact that storing the facets allows the algorithm to determine if a facet has been seen before without having to execute the adjacency oracle.

The cost of this improvement is that the facets of the critical regions are now stored rather than the critical regions themselves, resulting in a search complexity of  $\mathcal{T}_{search} = \log(N_f)$ , where  $N_f$  is the number of facets.

It follows that the total time complexity is given by:

$$\begin{aligned} \mathcal{T}_{facet} &= N_r \mathcal{T}_{fct\ orl} + N_r \mathcal{T}_{adj\ orl} + N_f \mathcal{T}_{search} \\ &= \mathcal{O}(N_r((n-m)(LP(d, n-m) + \log((n-m)N_r)) + LP(n, m))), \end{aligned}$$

where the change from (9.16) is seen by the change from  $N_f$  to  $N_r$  in the adjacency oracle multiplier. Recalling that the storage of each facet requires storing the vector  $T$  from Theorem 9.11, which can be as long as  $n-m$  numbers (complement of the basis) gives the space

complexity as:

$$\mathcal{S}_{facet} = \mathcal{O}(N_r(n - m))$$

### 9.4.3 Primal-Dual Enumeration

In this section we will present a heuristic based on [BFM98b] that can drastically reduce the work done in the facet oracle in some cases. We will restrict our attention to problems in which the cost  $c$  is zero, i.e. problems of the form:

$$\begin{aligned} f(\theta) \triangleq \quad & \text{maximise} && (E\theta)^T x \\ & \text{subject to} && x \in D \end{aligned} \tag{9.17}$$

In Part III it will be seen that all multiparametric linear programs can be cast in this form through a homogenisation procedure and that it is of particular interest in solving projection problems. We will here introduce just enough of this formulation that the proposed search algorithm can be discussed.

From the following Theorem, we see that for this formulation, the goal of computing critical regions can be re-posed as a vertex enumeration problem.

**Theorem 9.17.** *The basis  $B$  of  $D$  defines a critical region of mpLP (9.17) if and only if  $E^T D_B$  is a vertex of  $E^T D$ .*

*Proof.* We first show that every optimal basis of mpLP (9.17) maps to a vertex of  $E^T D$  and then show that every vertex of  $E^T D$  defines a critical region.

The mpLP (9.17) can be re-written in the following form:

$$\begin{aligned} f(\theta) = \quad & \text{maximise} && \theta^T z \\ & \text{subject to} && z \in E^T D, \end{aligned} \tag{9.18}$$

where  $E^T D$  is given by  $\{E^T x \mid x \in D\}$ . For a given  $\theta$ , the set of optimisers of (9.18) will be a face of  $E^T D$  [Zie95, Sec. 3.2], and if the perturbed problem is considered, this face will be a vertex. Since only vertices of  $D$  map to vertices of  $E^T D$ , as it is a linear map, for each vertex  $v$  of  $E^T D$ , there must exist a basis  $B$  of  $D$  such that  $v = E^T D_B$ .

We now show that every basis  $B$  that defines a vertex  $D_B$  that maps to a vertex of  $E^T D$  also defines a critical region. Let  $d_1, \dots, d_n$  be rays along the edges away from the vertex  $D_B$ . It follows that the feasible directions at the vertex  $E^T D_B$  of  $E^T D$  are given by cone  $\{E^T d_1, \dots, E^T d_n\} = \{E^T d_1 \lambda_1 + \dots + E^T d_n \lambda_n \mid \lambda_i \geq 0\}$ . From Theorem 8.5 the

## 9.4 ENUMERATION ALGORITHMS

---

direction  $\theta$  is optimal if and only if  $d_i^T E \theta \leq 0$  for all  $i$  and this cone is full-dimensional if and only if  $E^T D_B$  is a vertex of  $E^T D$ .

Noting that by definition,  $d_i^T E$  is the reduced cost  $\bar{E}_i$  and by Corollary 9.9 that  $\pi_\theta P_B$  is  $\{\theta \mid \bar{E}\theta \geq 0\}$  gives the desired result:  $\pi_\theta P_B$  is a critical region if and only if  $E^T D_B$  is a vertex of  $E^T D$ .  $\square$

We now develop a modification to the mpLP algorithm that exploits the result of Theorem 9.17. The algorithm is called ‘primal-dual’ because both the primal (vertex) and dual (halfspace) representations of  $E^T D$  are computed. At the  $q^{\text{th}}$  step of the algorithm,  $q$  vertices of  $E^T D$  will have been found. At this point, the algorithm has a list of these  $q$  vertices  $V^q \triangleq \{v_1, \dots, v_q\}$ , but unlike previous search methods, it also stores a halfspace representation of their convex hull  $\mathcal{H}^q \triangleq \{z \mid G^q z \leq g^q\} = \text{conv}\{v_0, \dots, v_q\}$ . 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  $\mathcal{H}^{q+1} = \{z \mid G^{q+1} z \leq g^{q+1}\} = \mathcal{H}^q \cup \{v_{q+1}\}$ .

We are now able to use this information to improve the efficiency of the facet oracle. As before, the reduced costs  $\bar{E}$  are computed, but now, instead of computing LPs to remove redundancies in the set  $\{\theta \mid \bar{E}\theta \geq 0\}$ , we notice that the set of feasible directions at the vertex  $v_{q+1}$  of  $E^T D$  is given by cone  $\left\{(\bar{E}_1)^T, \dots, (\bar{E}_n)^T\right\}$  and that the extreme rays of this cone are exactly the directions along the edges of  $E^T D$  away from the vertex  $v_{q+1}$ . We are therefore able to test if each ray  $r_i \triangleq \left\{z \mid z = v_{q+1} + (\bar{E}_i)^T t, t \geq 0\right\}$  points into the interior of the convex hull of the vertices found so far and if it does, then it clearly does not point along an edge. This notion is formalised in the following Theorem.

**Theorem 9.18.** *Let  $\mathcal{H} = \text{conv}\{v_0, \dots, v_q\} = \{z \mid Gz \leq g\}$ , where  $v_i$  are  $q$  vertices of  $E^T D$ . If  $v = E^T D_B$  is a vertex of  $E^T D$ , then facet  $F_i$  of critical region  $\pi_\theta P_B$  is redundant if for each  $j$  such that  $G_j v = g_j$  the condition  $G_j \bar{E}_i^T < 0$  holds.*

*Proof.* The test is simply to check if a point on the ray  $r_i = \{z \mid z = v + \bar{E}_i^T t, t \geq 0\}$  for a strictly positive  $t$  is internal to  $\mathcal{H}$ :

$$\begin{aligned} Gz &\leq g \\ Gv + G\bar{E}_i^T t &\leq g \\ G\bar{E}_i^T t &\leq g - Gv \end{aligned} \tag{9.19}$$

Recall that  $v$  is a vertex of  $\mathcal{H}$  and therefore  $g - Gv \geq 0$ . For those constraints that are strictly greater than zero, (9.19) will clearly be satisfied for some  $t > 0$  and therefore we have only to test those constraints that are equal to zero. Let  $Q = \{i \mid g_i = G_i v\}$ , then there exists a strictly positive  $t$  such that (9.19) is satisfied if and only if  $G_Q \bar{E}_i^T \leq 0$ .  $\square$

Algorithm 9.4 describes a replacement for the facet oracle, which does not compute the redundancy LP but rather returns all potential facets, i.e. those with non-zero reduced costs. The proposed primal-dual approach is given as Algorithm 9.5. One can see that the primary difference from Algorithm 9.3 is in Step 5, where Theorem 9.18 is put to work removing some redundancies without the need for a linear program. This process is described graphically in Figure 9.4.3.

---

**Algorithm 9.4** Primal-Dual: Compute Potential Facets

---

**Input:** Unique lex-optimal basis  $B$  such that  $E^T D_B$  is a vertex of  $E^T D$

**Output:** List of potential facets that is a superset of the true facets

- 1: Compute  $\bar{E}$
  - 2:  $ToTest \leftarrow \{i \mid \bar{E}_i \neq 0\}$
  - 3: **while**  $ToTest$  is not empty **do**
  - 4: Select any member  $i$  from  $ToTest$
  - 5: Compute constraints  $Q$  equal to  $i$ :  $Q \leftarrow \{j \mid \bar{E}_i \propto \bar{E}_j\}$
  - 6: Compute strictly active constraints:  $T \leftarrow Q^c \cup \{j \mid \bar{E}_j = 0\}$
  - 7:  $ToTest \leftarrow ToTest \setminus Q$
  - 8: Report facet  $\{\theta \mid \bar{E}_i \theta = 0\} \cap \pi_\theta P_B$  and the strictly active constraints  $T$ .
  - 9: **end while**
- 

One can see from Algorithm 9.5 that the goal is to avoid computing the redundancy test, LP (9.14). There are two key differences from the previous algorithms that reduce the number of LPs called. First, at Step 11, both the redundant and irredundant inequalities (potential facets) that describe the critical region are placed in  $\mathcal{L}_{unexplored}$ , rather than first removing redundancies. As a result, any inequalities that have been seen before will be removed in Step 14, which potentially prevents two LPs from being calculated: one for the facet under consideration and one for the facet already in the list. The cost of this is that the list  $\mathcal{L}_{unexplored}$  will be potentially much larger as it contains redundant facets. However, as the list can be searched and updated in logarithmic time, this isn't an issue unless the problem is very large.

Second, the primal-dual test is made in Step 5, exploiting the stored halfspace description of  $E^T D$ . As this test can only prove redundancy and not irredundancy, if the test fails then the redundancy test, LP (9.14) must be executed as a last resort. The cost of computing the primal-dual test is clearly the calculation and storage of the halfspace description of  $E^T D$ . While this convex hull can be computed efficiently in an incremental fashion, e.g. via the quickhull algorithm [BDH96], there are two difficulties with this approach. First, no incremental algorithm can be output sensitive [Bre99], implying that even if it is known that the complexity of the halfspace description of  $E^T D$  is simple, the incremental convex hulls may be complex. Second, the relationship between the number of vertices in  $E^T D$  and the number



## 9.4 ENUMERATION ALGORITHMS

---



---

### Algorithm 9.5 Multiparametric Linear Programming: Primal-Dual Enumeration

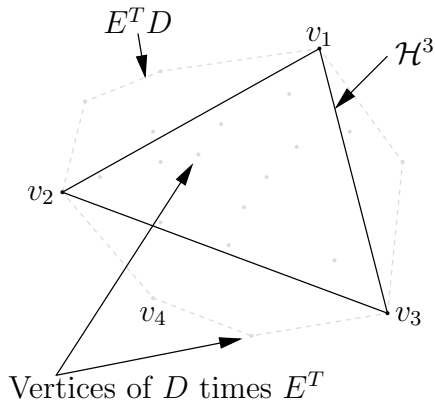
---

**Input:** Unique lex-optimal basis  $B$  such that  $E^T D_B$  is a vertex of  $E^T D$   
Matrix  $G$ , vector  $g$  such that  $\mathcal{H} = \{z \mid Gz \leq g\} \subseteq E^T D$

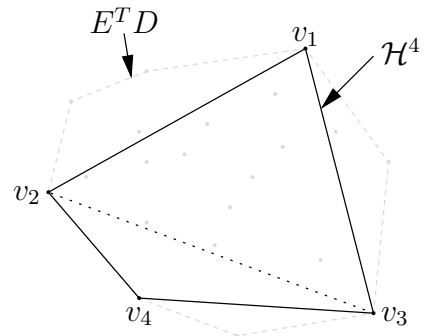
**Output:** All bases that define critical regions  
Matrix  $G$ , vector  $g$  such that  $\mathcal{H} = \{z \mid Gz \leq g\} = E^T D$

- 1:  $\mathcal{L}_{unexplored} \leftarrow \emptyset$
- 2: Goto 9
- 3: **repeat**
- 4:   Select any element  $(F, B')$  from  $\mathcal{L}_{unexplored}$
- 5:   **if**  $F$  does *not* satisfy Theorem 9.18 **then** Primal-Dual test
- 6:     **if** minimum of LP (9.14) is negative **then** Redundancy test
- 7:       Compute adjacent basis  $B$  such that  $F = \pi_\theta P_B \cap \pi_\theta P_{B'}$  Sec. 9.3.2
- 8:       **if** adjacent basis  $B$  exists **then**
- 9:          Report  $B$
- 10:       Increment convex hull:  $\mathcal{H} \leftarrow \mathcal{H} \cup \{E^T D_B\}$   $\mathcal{H} \triangleq \{z \mid Gz \leq g\}$
- 11:       Compute potential facets  $F_i$  Algorithm 9.4
- 12:       **for each** facet  $F_i$  **do**
- 13:          **if** there exists an element  $(F_i, X)$  in  $\mathcal{L}_{unexplored}$  for some  $X$  **then**
- 14:            Remove  $(F_i, X)$  from  $\mathcal{L}_{unexplored}$
- 15:          **else**
- 16:            Insert  $(F_i, B)$  into  $\mathcal{L}_{unexplored}$
- 17:          **end if**
- 18:       **end for**
- 19:       **end if**
- 20:     **end if**
- 21: **end if**
- 22: **if**  $(F, B') \in \mathcal{L}_{unexplored}$  **then**
- 23:    Remove  $(F_i, B')$  from  $\mathcal{L}_{unexplored}$
- 24: **end if**
- 25: **until**  $\mathcal{L}_{unexplored}$  is empty

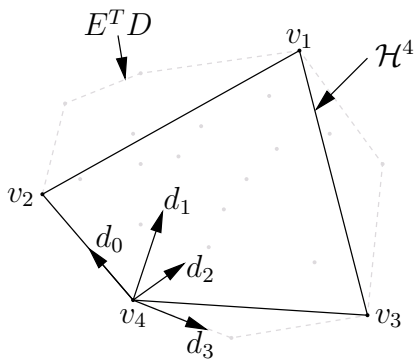
---



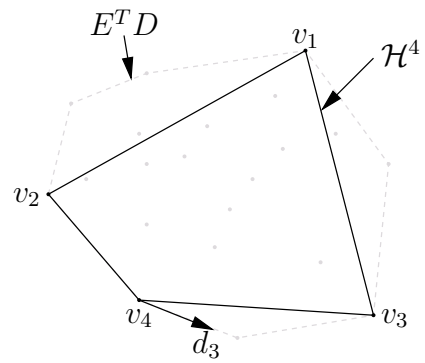
Initial convex hull  $\mathcal{H}^3$  and initial vertex  $v_4$ .



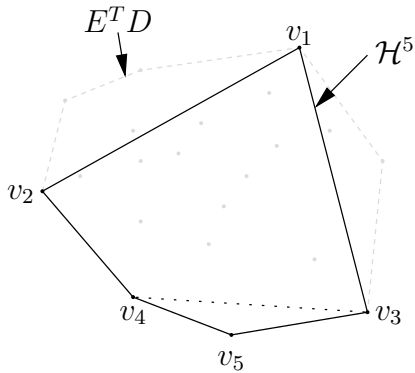
Step 10: Increment convex hull.



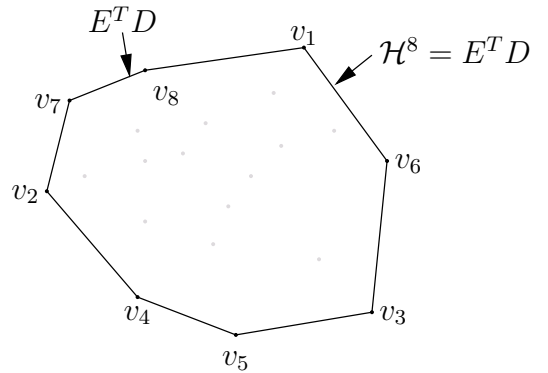
Step 11: Compute edge directions.  
Theorem 9.18:  $d_1$  and  $d_2$  are redundant



Step 4: Select unexplored facet and compute adjacent basis.



Step 10: Increment convex hull.



Repeat

Figure 9.5: Example Enumeration using the Primal-Dual Method.  
Gray dots are  $E^T v$  for the vertices  $v$  of the dual  $D$ .

## 9.4 ENUMERATION ALGORITHMS

---

of inequalities can be exponential. As a result the applicability of this algorithm is limited to those polyhedra  $E^T D$  that can be described with both relatively few inequalities and few vertices. An extension to the algorithm that would ameliorate some of these difficulties would be at Step 10 to only add a new vertex if it is sufficiently far from the existing description  $\mathcal{H}$ , which would prevent the calculation of very small facets in the convex hull.

### Complexity

As the number of halfspaces required to describe the dual is possibly exponential, the worst-case time and space complexity of the primal-dual algorithm is clearly exponential in the number of critical regions. It is therefore not competitive with the other algorithms presented here in terms of the worst-case. However, as will be seen in examples of Chapter 10, it offers a significant improvement for some smaller problems.

#### 9.4.4 Reverse Search for Enumeration

In this section an algorithm based on the reverse search approach of Avis and Fukuda [AF92, AF96, Avi00] will be introduced. The goal of reverse search is to remove the need to test if a newly discovered element has been seen before, i.e. Steps 6 and 13 in Algorithm 9.3 and Step 9 in Algorithm 9.2. While this test can be done in logarithmic time, the complexity is a function of the number of critical regions and so for very large problems can become prohibitive. More importantly, these tests require that discovered bases be stored in main memory, which can quickly be exhausted for large problems.

While both the basic or facet-based enumeration algorithms of the previous sections can be converted to a reverse search approach, we will here focus on the basis enumeration method of Section 9.4.1 for simplicity. In this algorithm we will again be considering the graph  $\mathcal{G}^{\mathcal{B}}(f)$ ; the vertices  $V$  of the graph are the bases that define critical regions and an edge joins two vertices if the intersection of the two critical regions is a facet of each. We will take the restricted form of the problem and assume that  $c = 0$ , recalling again that every mpLP of the form (7.1) can be written in this form.

Reverse search achieves its aim by converting the graph  $\mathcal{G}^{\mathcal{B}}(f)$  into a tree. The root of the tree is taken to be any critical-region defining basis; we shall call this root  $R$ . A unique mapping  $g : V \setminus \{R\} \rightarrow V$  is defined over the remaining vertices. This function is defined in such a way that if  $g$  is applied recursively to any vertex we will eventually arrive at the root  $R$ . In other words,  $g$  defines a unique path from every vertex to the root. Note that the definition implies that there are no circuits in the paths and that every vertex has exactly one edge pointing towards the root and possibly several pointing away. As a result, the mapping

$g$  defines a tree over the graph  $\mathcal{G}^{\mathcal{B}}(f)$ . The reverse search algorithm proceeds to enumerate all vertices by *reversing* the paths taken to the root and searching this tree in a depth-first fashion.

The algorithm begins at the root, chooses the first incoming path and follows it backwards one step to a new vertex. A recursive call is then made on this new vertex, which also defines a tree. Once the first path has been enumerated, the algorithm moves onto the second, and so on until all branches have been enumerated.

The key to the efficiency of this approach is that all decisions can be made based on *local* information. When a new vertex is discovered it does not need to be checked against previously discovered vertices because if it is further down the tree, we can be certain that it has not been seen before.

To apply reverse search to multi-parametric linear programming, two details need to be defined: the function  $g$ , and the notion of ordering on the paths. The function  $g$  is defined in a similar fashion to [AF92, AF96, Avi00] and is defined via the simplex algorithm. A cost  $r \in \mathbb{R}^d$  is chosen such that the transformed vertex  $v = E^T D_R$  is the unique optimiser of:

$$\begin{aligned} & \text{maximise} && r^T z \\ & \text{subject to} && z \in E^T D \end{aligned} \tag{9.20}$$

It is assumed that this cost  $r$  is not parallel to any of the edges of  $E^T D$ , i.e. that there will be no dual-degeneracy in LP (9.20), which can be made almost certain via a random perturbation. At each vertex  $E^T D_B$  there are a number of edges that can be followed to a new vertex. The function  $g$  chooses the edge along which the cost decreases most rapidly. Note that this edge is unique via the above assumption.

We now define the notion of ordering among the set of possible neighbours of a given vertex  $E^T D_B$ . Note that each neighbour is associated with an edge of  $E^T D$  and that there is a one-to-one mapping from edges leaving the vertex to facets of the critical region  $\pi_\theta P_B$  (Section 9.3). Corollary 9.9 gives the critical region  $\pi_\theta P_B$  as  $\{\theta \mid \bar{E}\theta \geq 0\}$  and each row of  $\bar{E}$  either defines a facet of the critical region, or is redundant. If the  $i^{\text{th}}$  row is irredundant, not zero and  $i$  is the smallest index that defines the facet, i.e.  $i = \min \left\{ j \mid \left[ \begin{array}{c} \bar{E}_i \\ \bar{c}_i \end{array} \right] \propto \left[ \begin{array}{c} \bar{E}_j \\ \bar{c}_j \end{array} \right] \right\}$ , then the  $i^{\text{th}}$  neighbour of  $E^T D_B$  is given by Theorem 9.11 for facet  $\{\theta \mid \bar{E}_i \theta = 0\}$  and the function  $Neighbour(B, i)$  is defined to return the adjacent basis. If any of the conditions are not met, then  $Neighbour(B, i)$  returns the empty set.

An example reverse-search tree is shown in Figure 9.6. The figure on the left shows the path that the function  $g$  would take from each vertex of the cube, and the figure on the right is the resulting reverse search tree. Another example is shown in Figure 9.7, where the

## 9.4 ENUMERATION ALGORITHMS

reverse-search tree is shown over points equally distributed across the surface of a sphere.

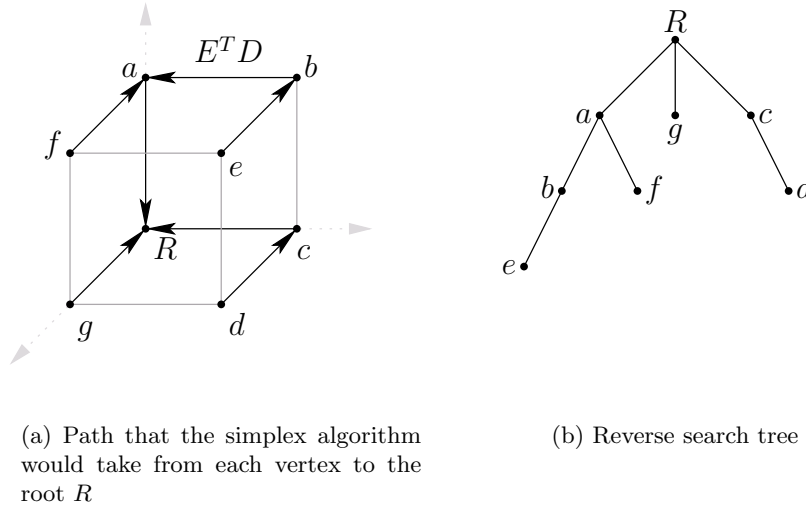


Figure 9.6: Reverse Search Illustration (Figure taken from [AF92])

The reverse-search method is shown as Algorithm 9.6, which is a standard implementation and is not much different from the generic algorithm of [AF96]. While Algorithm 9.6 looks slightly more complicated than those presented previously, the process is a very simple depth-first search, where the cost function  $g$  is used to avoid the requirement of a stack. The inner while loop (Steps 5–14) first move the algorithm down to a leaf of the tree. Step 16 then backtracks up the tree one step and Steps 17–21 determine which branch  $j$  the algorithm followed to get to the leaf. The neighbour counter is then incremented by one (Step 6) and the next branch is followed back down to a leaf. By continuing this until there are no more branches to follow, the entire tree will be visited. The example followed in the previous sections is again shown in Figure 9.8 for the reverse-search algorithm.

### Complexity

The time complexity of the reverse search enumeration is clearly a function of Steps 7, 9, 16 and 19:

**Step 7** In order to compute the  $j^{\text{th}}$  neighbour, the worst case will require a redundancy test (LP (9.14)) and the calculation of the neighbour (LP (9.10)):

$$\mathcal{T}_7 = \mathcal{O}(LP(d, n - m) + LP(n, n - m))$$

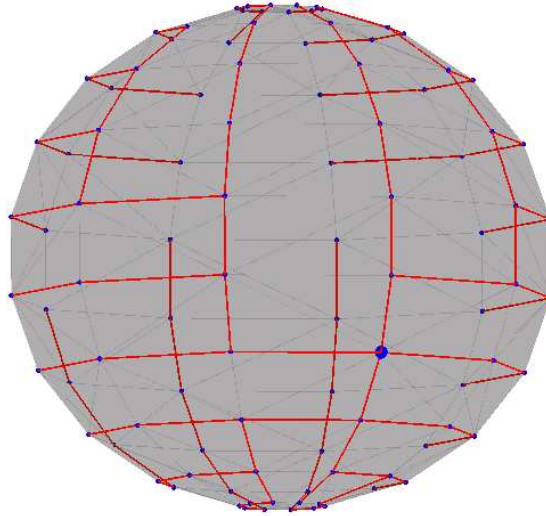


Figure 9.7: Reverse Search Tree for a Polytope  $E^T D$  whose Vertices are on a Sphere

---

**Algorithm 9.6** Multiparametric Linear Programming: Reverse Search for Enumeration

---

**Input:** Unique lex-optimal basis  $R$  such that  $E^T D_R$  is a vertex of  $E^T D$   
 Cost  $r$  such that vertex  $E^T D_R$  is the minimiser of (9.20)

**Output:** All bases that define critical regions

- |                     |                   |
|---------------------|-------------------|
| 1: $B \leftarrow R$ | Begin at the root |
| 2: $j \leftarrow 0$ | Neighbour counter |
| 3: Report $R$       |                   |
| 4: <b>repeat</b>    |                   |

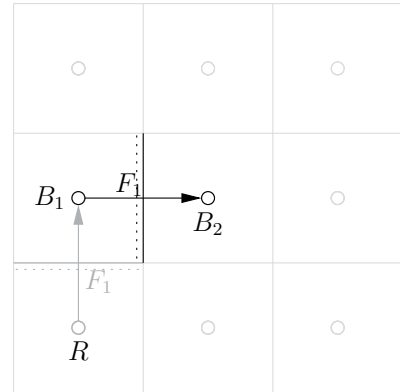
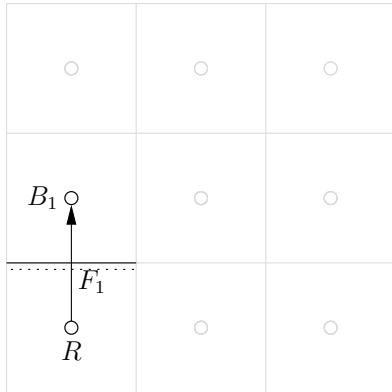
**Reverse traverse**

- |   |  |
|---|--|
| 5: <b>while</b> $j < n$ <b>do</b>   |  |
| 6:     Increment $j \leftarrow j + 1$   |  |
| 7:     Compute $j^{\text{th}}$ neighbour of $B$ : $next \leftarrow Neighbour(B, j)$ |  |
| 8: <b>if</b> $next \neq \emptyset$ <b>then</b>                                      |  |
| 9: <b>if</b> $g(next) = B$ <b>then</b>  |  |
| 10: $B \leftarrow next, j \leftarrow 0$   |  |
| 11:             Report $B$  |  |
| 12: <b>end if</b>   |  |
| 13: <b>end if</b>   |  |
| 14: <b>end while</b>  |  |

**Forward traverse**

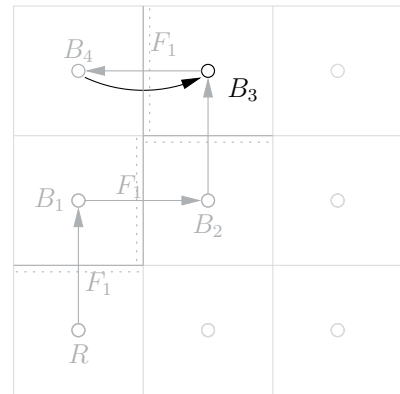
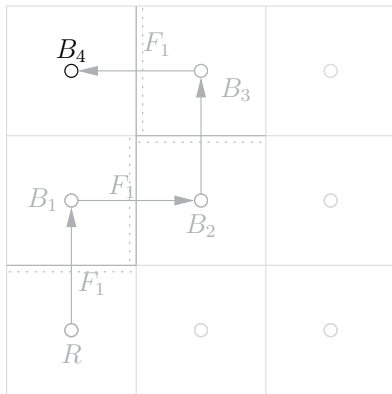
- |  |             |
|--|-------------|
| 15: <b>if</b> $B \neq R$ <b>then</b>                     |             |
| 16: $B' \leftarrow B, B \leftarrow g(B), j \leftarrow 0$ |             |
| 17: <b>repeat</b>  |             |
| 18: $j \leftarrow j + 1$                                 |             |
| 19: <b>until</b> $Neighbour(B, j) = B'$                  | Restore $j$ |
| 20: <b>end if</b>  |             |
| 21: <b>until</b> $v = R$ and $j = n$                     |             |
-

## 9.4 ENUMERATION ALGORITHMS



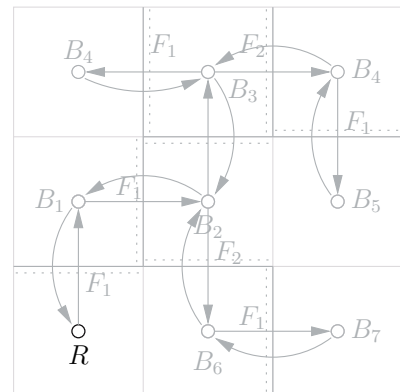
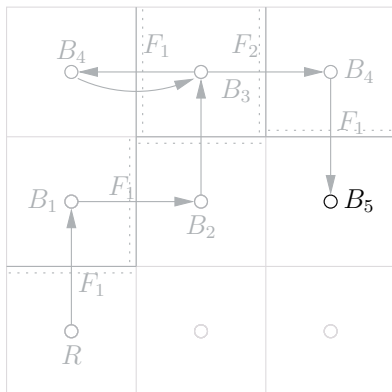
Reverse Traverse: Compute 1<sup>st</sup> neighbour of the root.

Reverse Traverse: Descend down the tree.



Reverse Traverse: Continue down the 1<sup>st</sup> neighbours until a leaf is found.

Forward Traverse: Move up the tree until a new branch is found.



Reverse Traverse: Descend the tree to a leaf.

Repeat until arriving back at the root with all branches explored.

Figure 9.8: Example Enumeration using the Reverse-Search Method.

Straight arrows are reverse steps and curved arrows are forward.

**Step 9** This test can be split into two parts: First, compute the direction of steepest descent, and second, test if this direction goes to the basis  $B$ . The first step requires the removal of redundancies in the critical region, while the second only requires testing the optimality conditions of LP (9.10), which requires fixed time:

$$\mathcal{T}_9 = \mathcal{O}((n - m)LP(d, n - m))$$

**Step 16** The complexity of this step is the same as that of Step 9, except LP (9.10) must actually be calculated, rather than simply testing the optimality conditions:

$$\mathcal{T}_{16} = \mathcal{O}((n - m)LP(d, n - m) + LP(n, n - m))$$

**Step 19** Finally, this step needs to test the optimality conditions of LP (9.10) and then determine if the  $j^{\text{th}}$  facet is redundant via LP (9.14):

$$\mathcal{T}_{19} = \mathcal{O}(LP(d, n - m))$$

We have now to determine how many times per vertex each of the above four steps is executed. Step 7 is called once for each facet of the solution  $N_f$ . As  $E^T D$  is in  $\mathbb{R}^d$ , there are  $d$  edges intersecting at the vertex  $E^T D_B$ , which need to be tested in Step 9. As each vertex of  $E^T D$  is contained in  $d$  edges, it is possible for Step 16 to be called  $d - 1$  times for each vertex. Finally, the loop 17–19 is called the same number of times as Step 16, but iterates up to  $d$  times. If there are  $N_r$  extreme points in the polyhedron  $E^T D$ , the time complexity of the reverse search enumeration method is given by:

$$\begin{aligned} \mathcal{T}_{rs} &= N_f \mathcal{T}_7 + N_r d \mathcal{T}_9 + N_r (d - 1) \mathcal{T}_{16} + N_r d (d - 1) \mathcal{T}_{19} \\ &= \mathcal{O}(N_r (d^2 LP(d, n - m) + d(n - m)(LP(d, n - m) + LP(n, n - m)))) \end{aligned}$$

Note that the time complexity  $\mathcal{T}_{rs}$  is linear in the number of critical regions and is only slightly worse than the complexity seen in Sections 9.4.1 and 9.4.2. The most important feature of the reverse search approach is that the space complexity is fixed and that no search ever needs be done over the critical regions that have been found so far:

$$\mathcal{S}_{rs} = \mathcal{O}(1).$$



## 9.4 ENUMERATION ALGORITHMS

---

There are two important extensions possible to the reverse search approach. First, the algorithm can be parallelised very efficiently, as each branch of the tree can be assigned to a different processor and no data will need to pass between the machines. Second, the computation speed can be drastically improved by storing the path taken down the tree in a stack. In this case, the size of the stack can be limited to a fixed size, maintaining the space complexity as constant.



# Chapter 10

## mpLP Examples

In this chapter we will show comparative numerical simulations of some problems that are of importance to control. We will demonstrate that the methods presented in this thesis are both the fastest currently available and the only approaches which are guaranteed to provide a continuous answer in the presence of degeneracy.

The primary motivation for mpLPs in control is the calculation of so-called closed-form or explicit MPC control 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 multiparametric form, with the state as a parameter, and solved offline.

The goal is to control the following LTI system:

$$x^+ = Ax + Bu,$$

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:

$$\begin{aligned}
 J(x) = & \underset{u_1, \dots, u_{N-1}, x_1, \dots, x_N}{\text{minimise}} && \sum_{i=1}^{N-1} \|Ru_i\|_p + \sum_{i=1}^{N-1} \|Qx_i\|_p + \|Q_F x_N\| \\
 & \text{subject to} && x_0 = x \\
 & && x_{i+1} = Ax_i + Bu_i, \quad i = 0, \dots, N-1 \\
 & && x_i \in \mathcal{X}, \quad i = 1, \dots, N-1 \\
 & && x_N \in \mathcal{X}_F, \\
 & && u_i \in \mathcal{U}, \quad i = 0, \dots, N-1,
 \end{aligned} \tag{10.1}$$

where  $x_i$  and  $u_i$  are future predicted states and inputs respectively, which are constrained to be in the polytopes  $\mathcal{X}$  and  $\mathcal{U}$ , with the state at the end of the horizon  $N$  required to lie in the

terminal set  $\mathcal{X}_F$ . The norm  $p$  is generally taken to be either the 1-, 2-, or  $\infty$ -norm and, if the 2-norm is used,  $R \in \mathbb{R}^{m \times m}$ ,  $Q \in \mathbb{R}^{n \times n}$  and  $Q_F \in \mathbb{R}^{n \times n}$  are positive semi-definite.

If the square of the 2-norm is used in (10.1), then the problem becomes a quadratic program, whereas if the 1- or  $\infty$ -norms are used, then a linear program results. As we are only interested in linear programs in this thesis, we shall make the standing assumption of a linear norm. In order to convert (10.1) into a multiparametric linear program of the form (9.1), a standard trick must be used to remove the norms in the cost. Let  $t \in \mathbb{R}^n$  be any vector, then the 1- and  $\infty$ -norms can be written as:

$$\begin{aligned} \|t\|_\infty &\triangleq \max_{i \in \{1, \dots, n\}} |t_i| \\ &= \min_s \{s \mid |t_i| \leq s, i = 1, \dots, n\} \\ &= \min_s \{s \mid s \geq 0, -s \leq t_i \leq s, i = 1, \dots, n\} \end{aligned} \quad (10.2)$$

$$\begin{aligned} \|t\|_1 &\triangleq |t_1| + \dots + |t_n| \\ &= \min_{s_1, \dots, s_n} \{s_1 + \dots + s_n \mid s_i \geq 0, -s_i \leq t_i \leq s_i, i = 1, \dots, n\} \end{aligned} \quad (10.3)$$

The identities (10.2) and (10.3) allow the MPC problem (10.1) to be re-written as a standard mpLP:

$$\begin{aligned} J(x) = & \quad \text{minimise} && \sum_{i=1}^{N-1} \mathbf{1}^T s_i + \sum_{i=1}^N \mathbf{1}^T t_i \\ & x_1, \dots, x_N, u_0, \dots, u_{N-1} \\ & s_1, \dots, s_{N-1}, t_1, \dots, t_N \\ \text{subject to} & && x_0 = x \\ & && x_{i+1} = Ax_i + Bu_i, \quad i = 0, \dots, N-1 \\ & && x_i \in \mathcal{X}, \quad i = 1, \dots, N-1 \\ & && x_N \in \mathcal{X}_F, \\ & && u_i \in \mathcal{U}, \quad i = 0, \dots, N-1, \\ & && s_i \geq 0, \quad i = 1, \dots, N-1 \\ & && t_i \geq 0, \quad i = 1, \dots, N \\ & && -\mathbf{1}s_i \leq Ru_i \leq \mathbf{1}s_i, \quad i = 0, \dots, N-1 \\ & && -\mathbf{1}t_i \leq Qx_i \leq \mathbf{1}t_i, \quad i = 1, \dots, N-1 \\ & && -\mathbf{1}t_N \leq Q_F x_N \leq \mathbf{1}t_N, \end{aligned} \quad (10.4)$$

where  $s_i$  and  $t_i$  are in  $\mathbb{R}$  if the  $\infty$ -norm is used and  $s_i$  is in  $\mathbb{R}^m$  and  $t_i$  is in  $\mathbb{R}^n$  if the 1-norm is used. LP (10.4) can be written in a simpler form, as in Chapter 6, by defining the vectorised sets  $X^T \triangleq [x_1^T \ \dots \ x_N^T]$ ,  $U^T \triangleq [u_0^T \ \dots \ u_{N-1}^T]$ ,  $S^T \triangleq [s_1^T \ \dots \ s_{N-1}^T]$ , and  $T^T \triangleq [t_1^T \ \dots \ t_N^T]$  and the matrices  $\mathcal{A} \triangleq I_N \otimes [A \ -I_n]$ ,  $\mathcal{A}_x \triangleq \mathcal{A}_{\star, \{1, \dots, n\}}$ ,  $\mathcal{A}_X \triangleq$

## 10.1 DOUBLE INTEGRATOR

---

$$\mathcal{A}_{\star, \{n+1, \dots, nN\}} \text{ and } \mathcal{B} \triangleq I_{N-1} \otimes B, \mathcal{R} \triangleq I_{N-1} \otimes R \text{ and } \mathcal{Q} \triangleq \begin{bmatrix} I_{N-1} \otimes R & 0 \\ 0 & Q_F \end{bmatrix}. \text{ LP (10.4)}$$

becomes:

$$\begin{aligned} J(x) = & \underset{U, S, T}{\text{minimise}} & & \mathbf{1}^T S + \mathbf{1}^T T \\ \text{subject to} & & \mathcal{A}_X^{-1}(\mathcal{A}_x x + \mathcal{B}U) & \in & \mathcal{X}^{N-1} \times \mathcal{X}_F, \\ & & U & \in & \mathcal{U}^{N-1} \\ & & S & \geq & 0, \\ & & T & \geq & 0, \\ & & -(I_N \otimes \mathbf{1})T & \leq & \mathcal{Q}\mathcal{A}_X^{-1}(\mathcal{A}_x x + \mathcal{B}U) \leq (I_N \otimes \mathbf{1})T \\ & & -(I_N \otimes \mathbf{1})S & \leq & \mathcal{R}U \leq (I_N \otimes \mathbf{1})S \end{aligned} \quad (10.5)$$

Now that the MPC problem is in the form (9.1), it can be solved using the methods presented in this part. We now present some examples demonstrating the salient properties of the algorithm.

**Remark 10.1.** *Note that  $\mathcal{A}_X$  is always invertible in (10.5).*

## 10.1 Double Integrator

We first return to the double integrator Example (6.2.1):

$$x(k+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u(k)$$

The following input and state constraints must be met at each point in time:

$$\begin{aligned} -1 & \leq u(k) \leq 1, & \forall k \geq 0 \\ -\begin{pmatrix} 5 \\ 5 \end{pmatrix} & \leq x(k) \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix}, & \forall k \geq 1 \end{aligned}$$

If a prediction horizon of  $N = 2$  is assumed and  $R = 1$ ,  $Q = Q_F = I$ , then the mpLP (10.5) is:

$$\begin{aligned}
 & \underset{U,S,T}{\text{minimise}} && \mathbf{1}^T S + \mathbf{1}^T T \\
 & \text{subject to} && - \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 2 \\ 2 \end{pmatrix} \leq \begin{bmatrix} 2 & 2 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 2 & 4 & 3 & 2 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{pmatrix} x \\ U \end{pmatrix} \leq \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 2 \\ 2 \end{pmatrix} \\
 & && - \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} T \\ S \end{pmatrix} \leq \begin{bmatrix} 2 & 2 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 2 & 4 & 3 & 2 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{pmatrix} x \\ U \end{pmatrix} \leq \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} T \\ S \end{pmatrix} \\
 & && S \geq 0, T \geq 0
 \end{aligned} \tag{10.6}$$

The solution complex to this example is shown as Figure 10.1(a). The input applied to the system,  $u_0$  (i.e. the first primal optimiser), is shown as Figure 10.1(b) and the cost as Figure 10.1(c).

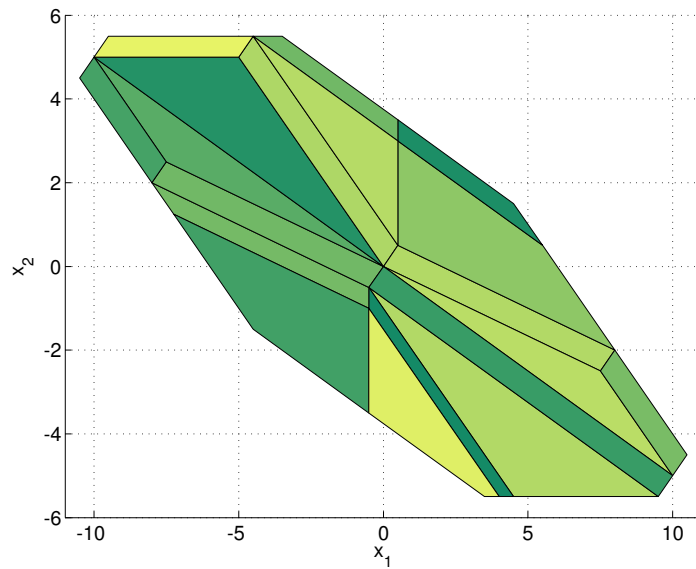
The following two examples will be used to compare the computational complexity of the methods surveyed in Section 7.1 with the enumeration techniques presented in this thesis.

All of the algorithms that will be compared have been implemented in MATLAB, and as such it would seem that the time taken by each approach could be compared directly. However, computational experience has shown that some of the implementations take significantly longer in MATLAB overhead than others. As discussed in Section 9.4, the primary cost of the enumeration algorithms is the number of LPs and the time to search through the stored data. All examples that can be calculated with current implementations have small enough data sets as to make their search almost negligible when compared to the time taken by the LPs. As a result, the time taken by the LPs in each algorithm is a fair comparison of the computational cost. In order to isolate the complexity measurement from any implementation issues, we define an *optimal implementation*, in which only the LPs are computed and all other calculations are instantaneous. This will clearly be a lower bound on any implementation, although for large problems a good code should come close.

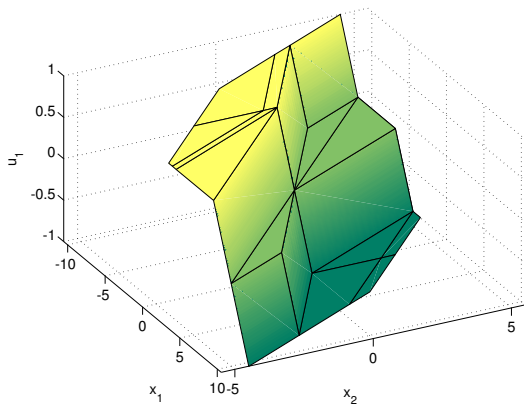
In order to compute the complexity of the optimal implementation, the time for a single

## 10.1 DOUBLE INTEGRATOR

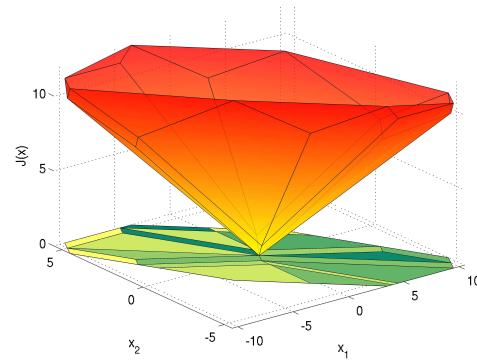
---



(a) Solution Complex



(b) Input  $u_0(x)$



(c) Cost  $J(x)$

Figure 10.1: Solution to Example 10.1

LP pivot is needed, which is known to be approximately a linear function of the number of constraints and a quadratic function of the dimension. The range of interest in the following examples was gridded and 100,000 LPs were carried out on random data at each point, counting the number of pivots taken by each LP. Least-squares was then used to fit the data to the following second-order polynomial and the following equation was determined, relating the number of constraints  $N$  and the number of dimensions  $d$  to the number of microseconds for a single pivot  $t$ :

$$t(N, d) = 0.02238d^2 - 0.80456d + 0.15975N + 0.00457dN + 30.80649 \quad (10.7)$$

The fit is very good, with a correlation coefficient of  $R^2 = 0.9996$  and the curve and fitting error can be seen in Figure 10.2. This time is, of course, dependent on the machine used to do the calculation, but any change in the machine speed would appear as a scaling of  $t(N, d)$ . The machine used to derive (10.7) was a 3GHz Pentium IV and the LP code was the Stanford Systems Optimization Laboratory (SOL) toolbox [MS].

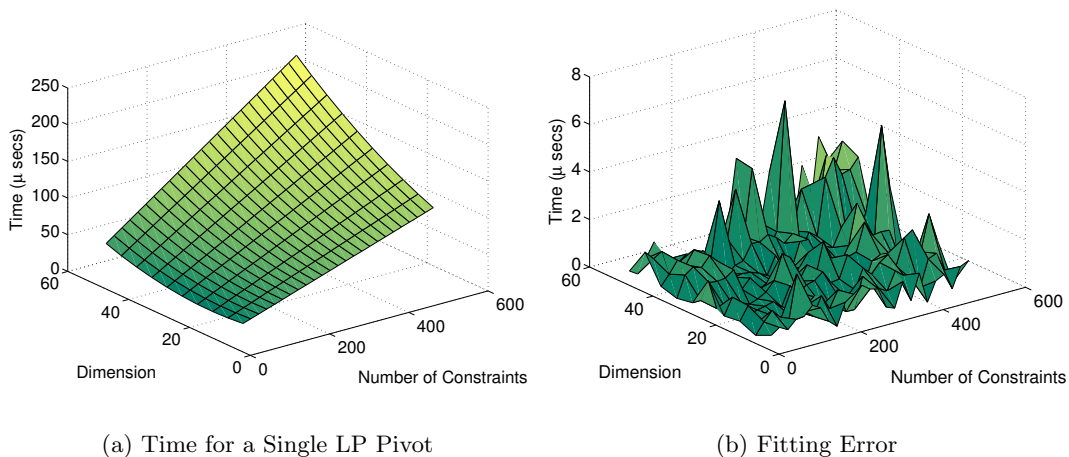


Figure 10.2: Time for a Single LP Pivot as a Function of Size



## 10.2 Closed-Form MPC for Random 3D System

In this example we will investigate the performance difference between currently available mpLP codes. Consider the following randomly generated MPC problem:

$$x(k+1) = \begin{bmatrix} -0.3551 & 0.4523 & -0.1813 \\ 0.4523 & -0.6329 & -0.2076 \\ -0.1813 & -0.2076 & -0.0825 \end{bmatrix} x(k) + \begin{bmatrix} -1.0068 & -0.9992 \\ 1.5975 & 0 \\ 1.0554 & 1.4262 \end{bmatrix} u(k)$$

with a prediction horizon  $N = 5$  and the constraints  $\|u(k)\|_\infty \leq 1$ ,  $\|x(k)\|_\infty \leq 5$  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  $Q$  and  $R$  are taken as the identity.

The resulting mpLP problem is significantly larger than that presented in the previous example. When the problem is written in standard form as:

$$\begin{aligned} & \text{minimise} && (E\theta + c)^T x \\ & \text{subject to} && Ax = b, x \geq 0 \end{aligned} \tag{10.8}$$

the matrix  $A$  is in  $\mathbb{R}^{20 \times 100}$  and  $E$  is in  $\mathbb{R}^{100 \times 3}$  and the solution complex contains 718 regions.

The number and size of the pivots for each of the methods discussed above were collected and analysed. The resulting data is presented in Table 10.1. The ‘Time’ column refers to the measured amount of time that the current implementation of the algorithms took. The ‘Pivots’ columns show the number of low-dimensional (3D) and high-dimensional (20D) pivots that were taken as well as the average number of constraints per pivot. The time taken doing low-dimensional (redundancy elimination) and high-dimensional (adjacency oracle) operations is shown in the next two columns using (10.7). The final column shows the time for an optimal implementation of the given algorithm.

A few comments are in order regarding each of the six approaches:

**Basic/Facet (Algs 9.2 and 9.3)** Note that, as expected, the difference between these two approaches is in the number of high-dimensional (adjacency oracle) pivots required. While the difference in the number of pivots is significant, the time required to do redundancy elimination is far greater and therefore we do not see a large difference in the total optimal time. One would expect this difference to be more pronounced in problems with a larger distance between the high-dimension (20D) and the low-dimension (3D).

**PD (Alg 9.5)** The primal-dual approach eliminates the vast majority of the redundancy elimination requirements, while maintaining the good high-dimensional performance of

the Facet approach. The cost of this is the requirement of computing the convex hull of the dual polytope. In this example, the convex hull calculation took less than 0.1 seconds using the qhull [BDH96] algorithm. The implementation used to produce the results seen in Table 14.1 does not have access to an iterative convex hull algorithm, and therefore the complete convex hull must be computed at every step. This is the reason for the very significant difference between the measured and optimal times.

**RS (Alg 9.6)** The primary appeal of the reverse-search approach is its fixed storage space requirements. For problems of the size considered here, the data fits easily within 100MB for all methods. However, the size of the data grows very quickly with the size of the problem and so it is simple to find problems that exceed the main memory. Once this has occurred, computation time becomes entirely a function of the speed of the hard disk and therefore the reverse-search algorithm is a very useful method.

**Facet Traversal Method (FTM)** The mpLP method included with the Multiparametric Toolbox [KGBM04] has many similarities to the basis enumeration method described in Section 9.4.1. The primary difference that changes the computation time is that the adjacency oracle is computed by explicitly finding a point  $\theta_c$  on the facet, moving over the facet to a point in the neighbouring region  $\theta_o$  by adding a small vector to it and solving a high-dimensional LP for the new point. This is compared to the approach taken in the adjacency oracle presented here where, for a non-degenerate facet, exactly one pivot is required. This can be seen from Table 10.1 in both the number of 20-dimensional pivots to be computed as well as their average number of constraints. For a non-degenerate facet the adjacency oracle presented in this thesis has exactly one constraint more than there are dimensions (21), whereas the MPT adjacency oracle requires all constraints in the original problem to be included (100).

**Region Complement Method (RCM)** The method presented in [BBM00] is required to compute the complement of each critical region found and then intersect this complement with all other regions to test for overlap. These tests cost a very large number of LPs in the low-dimensional space as seen in the table. As a result, the approach has been found not to be competitive with other methods except on specific examples.

**Remark 10.2.** *The implementation of the method in [BBM00] reported in Table 10.1 was written by the author of this thesis. The author of [BBM00] has recently released a toolbox [Bem03] that has also been used to solve this problem. While the time taken by the [Bem03] implementation was only 128 seconds rather than 794, the number of reported pivots was significantly higher, and we therefore report the implementation that gives the best optimal time.*

## 10.2 CLOSED-FORM MPC FOR RANDOM 3D SYSTEM

Method	Time (secs)	Pivots				Optimal Time		
		3D		20D		3D	20D	Total
		Number	Avg N	Number	Avg N	(secs)	(secs)	(secs)
Basic	41.6	134,839	40.0	10,500	23.5	4.8	0.3	5.1
Facet	29.5	134,819	40.0	3,385	24.1	4.8	0.1	4.9
PD	333.7	21,042	71.2	3,435	23.7	0.9	0.1	1.2 <sup>a</sup>
RS	359.9	734,620	77.8	28,218	24.4	30.7	0.9	31.6
FTM	631.5	161,702	39.5	70,167	99.8	5.8	3.4	9.2
RCM	794.5	1,392,740	70.5	36,143	26.2	57.0	1.2	58.2

<sup>a</sup>Includes time to compute convex hull using qhull [BDH96] - 0.2 seconds

Table 10.1: Comparison of mpLP Methods for Example 10.2

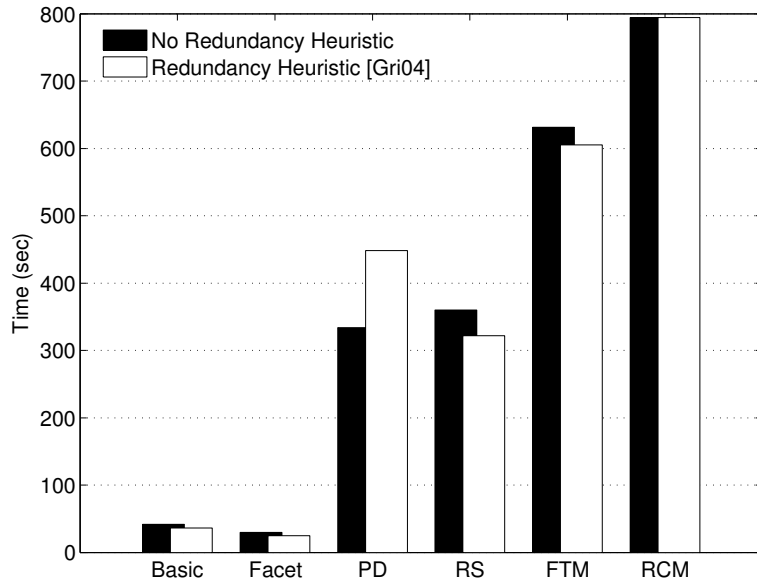
As can be seen from Table 10.1, a very significant portion of the calculation time for all methods is spent on low-dimensional pivots, which are executed for the purpose of redundancy elimination. A heuristic to reduce the cost of redundancy elimination in control problems has recently been proposed in [Gri04]. This method has been implemented and the result shown in Table 10.2. The computation time is reduced enough to make its use worthwhile, and experience has shown that it is very good for 2-dimensional problems. However, the algorithm does not scale well, and seems to have less effect in higher dimensions, as will be seen in the next example.

Finally, the pertinent results from Tables 10.1 and 10.2 are shown graphically in Figure 10.3, which demonstrates the relative performance of the surveyed approaches.

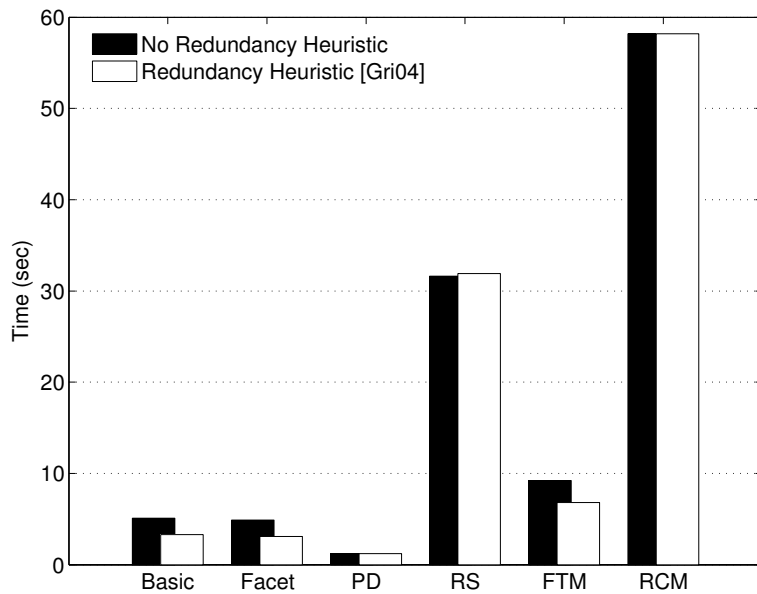
Method	Time (secs)	Pivots				Optimal Time		
		3D		20D		3D	20D	Total
		Number	Avg N	Number	Avg N	(secs)	(secs)	(secs)
Basic	36.5	85,866	35.0	10,272	23.5	3.0	0.3	3.3
Facet	24.7	85,847	35.0	3,353	24.0	3.0	0.1	3.1
PD	448.5	22,142	63.2	4,130	25.7	0.9	0.1	1.2 <sup>a</sup>
RS	321.8	789,701	63.9	28,672	24.4	31.1	0.9	31.9
FTM	605.2	99,352	28.6	70167	99.8	3.4	3.4	6.8
RCM	794.5	1,392,740	70.5	36,143	26.2	57.0	1.2	58.2

<sup>a</sup>Includes time to compute convex hull using qhull [BDH96] - 0.2 seconds

Table 10.2: Comparison of mpLP Methods for Example 10.2 using Redundancy Elimination Heuristic [Gri04]



(a) Time Taken for Current Matlab Implementation



(b) Time Taken for Optimal Implementation

Figure 10.3: Method Comparison for Example 10.2

### 10.3 Closed-Form MPC for 4D System

In this example, we present a larger problem, which is on the limits of what current implementations are capable of. Consider the following system, which is given as an example in the MPT toolbox [KGBM04]:

$$x(k+1) = \begin{bmatrix} 0.7 & -0.1 & 0.0 & 0.0 \\ 0.2 & -0.5 & 0.1 & 0.0 \\ 0.0 & 0.1 & 0.1 & 0.0 \\ 0.5 & 0.0 & 0.5 & 0.5 \end{bmatrix} x(k) + \begin{bmatrix} 0.0 & 0.1 \\ 0.1 & 1.0 \\ 0.1 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} u(k),$$

where the following input and state constraints must be met at each point in time:

$$\|x(k)\|_{\infty} \leq 5, \quad \|u(k)\|_{\infty} \leq 5.$$

The weighting matrices  $Q$  and  $R$  are the identity, the horizon  $N$  is 5 and the norm to be minimised is the  $\infty$ -norm.

Although this system looks hardly more complex than the previous example, the increase in the state dimension of one makes an enormous difference to the complexity of the solution complex: whereas Example 10.2 had 718 critical regions, this one contains 12,128.

The same algorithms as in the previous example were run and the results are shown in Tables 10.3 and 10.4 with and without the redundancy elimination heuristic [Gri04] respectively. There is one notable omission: the algorithm from [BBM00]. This approach does not seem to scale well with dimension and made virtually no progress before crashing MATLAB after ten minutes. As before, a comparison is also made graphically in Figure 10.4.

While the complexity of the problem has increased drastically, the time taken per critical region has only increased by approximately 50% for all methods except PD (Alg 9.5), which has increased by 430%. This increase is due entirely to the relationship between the number of vertices and the number of facets in  $E^T D$ . For this example, there are 282,162 facets to 12,128 vertices, whereas in the previous example there were 5,131 facets to 718 vertices; an increase of 225% in ratio of facets to vertices. There is no reason not to believe that this growth will continue for control problems of this type, which implies that the primal-dual algorithm is of use at low dimensions only unless the problem has particular structure.

## 10. MPLP EXAMPLES

Method	Time (secs)	Pivots				Optimal Time		
		3D		20D		3D	20D	Total
		Number	Avg N	Number	Avg N	(secs)	(secs)	(secs)
Basic	1,219.4	3,261,040	38.3	251,149	23.2	114.9	7.5	122.4
Facet	883.2	3,329,603	38.0	50,717	23.0	117.0	1.5	118.6
PD	13,963.0	761,487	96.5	76,488	25.2	34.3	2.3	107.7 <sup>a</sup>
RS	15,687.1	34,245,395	98.2	628,918	23.8	1550.6	18.9	1569.5
FTM	18,884.0	6,409,503	54.4	670940	119.3	243.7	36.1	280.1

<sup>a</sup>Includes time to compute convex hull using qhull [BDH96] - 71.1 seconds

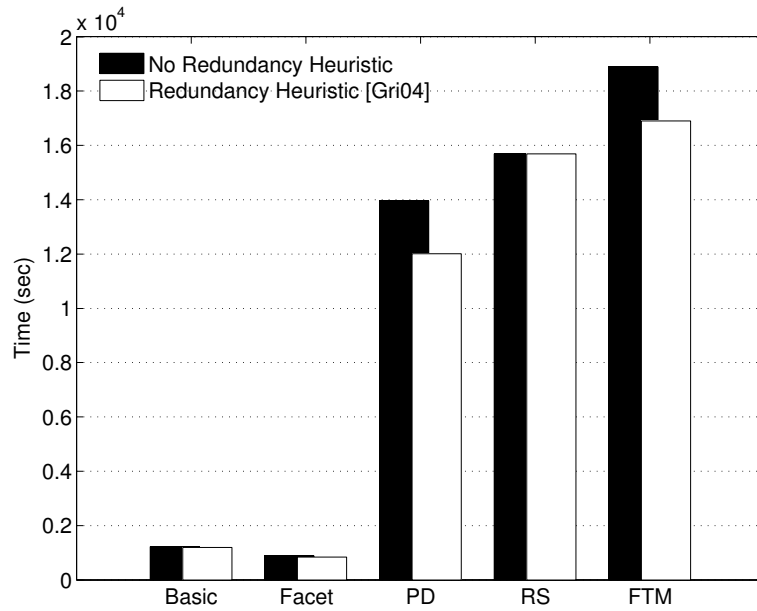
Table 10.3: Comparison of mpLP Methods for Example 10.3

Method	Time (secs)	Pivots				Optimal Time		
		3D		20D		3D	20D	Total
		Number	Avg N	Number	Avg N	(secs)	(secs)	(secs)
Basic	1,193.0	3,511,888	40.6	236,844	22.8	126.1	7.0	133.1
Facet	843.8	2,936,183	36.7	50,717	23.0	101.2	1.5	102.7
PD	12,010.0	761,487	96.5	76,488	25.2	34.3	2.3	107.7 <sup>a</sup>
RS	15,687.1	34,615,931	98.2	635,023	23.8	1567.4	19.0	1586.4
FTM	16,900.1	3,938,090	39.4	670940	119.3	139.0	36.1	175.1

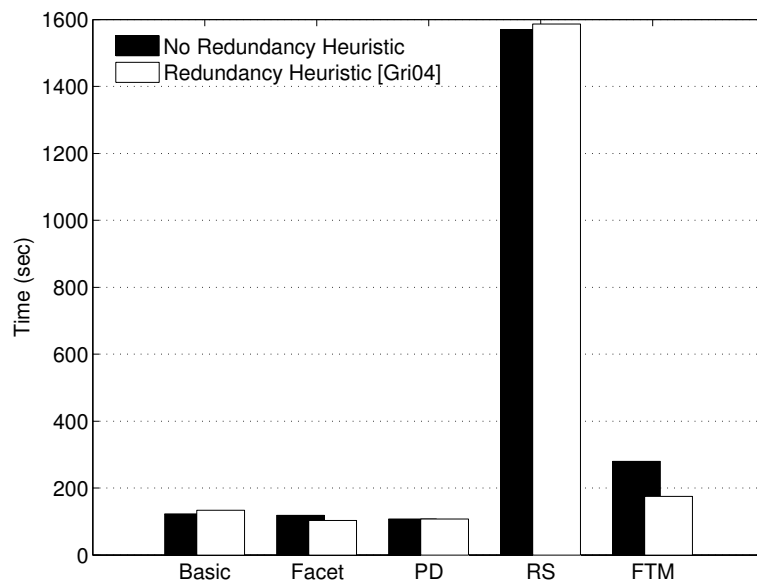
<sup>a</sup>Includes time to compute convex hull using qhull [BDH96] - 71.1 seconds

Table 10.4: Comparison of mpLP Methods for Example 10.3 using Redundancy Elimination Heuristic [Gri04]

### 10.3 CLOSED-FORM MPC FOR 4D SYSTEM



(a) Time Taken for Current Matlab Implementation



(b) Time Taken for Optimal Implementation

Figure 10.4: Method Comparison for Example 10.3

## 10.4 Degenerate Closed-Form MPC Example

For the final example in this section, we re-visit the double integrator of Example 10.1 and use it to investigate degeneracy. Recall that one of the primary contributions of this thesis is the development of an mpLP approach that is guaranteed to both find a solution in the presence of degeneracy and to ensure that it is continuous and that the critical regions form a complex.

We again use the double integrator of Example 10.1, but now the state and input costs are set to zero  $R = 0$ ,  $Q = 0$  and the horizon is  $N = 5$ . The resulting mpLP will, of course, be very degenerate. While this is a slightly pathological example, it demonstrates nicely the effect of degeneracy on the leading algorithms.

**Facet Traversal Method (FTM)** The resulting solution complex and control input are shown in Figure 10.5. It is immediately obvious from the figure that the critical regions no longer form a complex and that the input is discontinuous. More alarming, however, is that the algorithm can miss parts of the state space, as shown in the blown-up region of Figure 10.5(a).

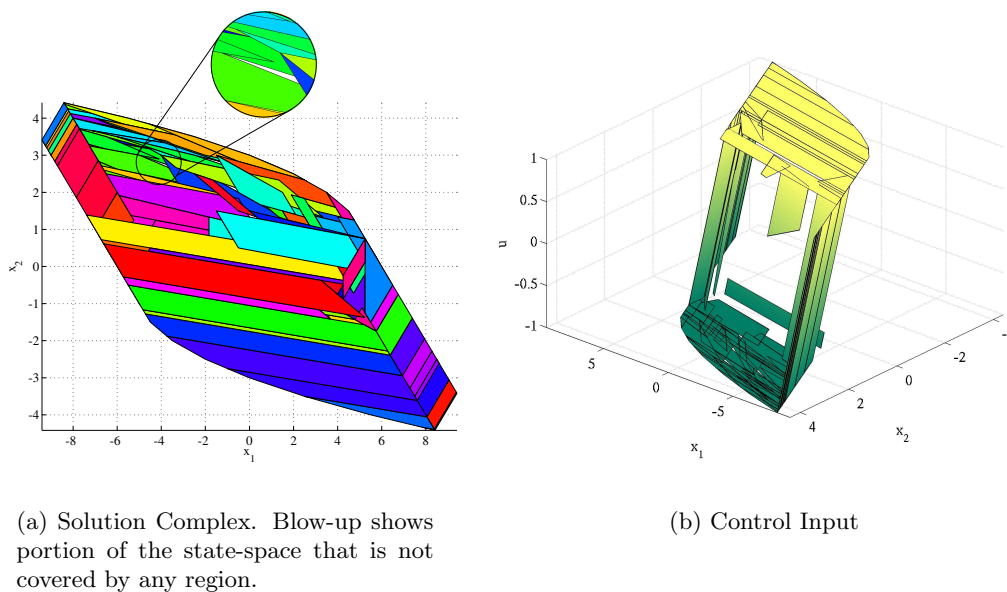


Figure 10.5: MPT Solution for Degenerate Example 10.4

**Region Complement Method (RCM)** The resulting solution complex and control input are shown in Figure 10.6. While the critical regions of this approach are guaranteed to



## 10.4 DEGENERATE CLOSED-FORM MPC EXAMPLE

---

cover the entire feasible region, the control input is certainly not continuous. Note also that the critical regions overlap and that there are many more of them than necessary. The solution complex and input shown in Figure 10.6 are the result of code written by the author of this thesis. There has recently been a preliminary toolbox released by the author of [BBM03]. This toolbox was used to solve this problem and the resulting solution is shown in Figure 10.7, although the toolbox was not able to plot the input. One can see that the solution is not a complex and that there are overlapping regions.

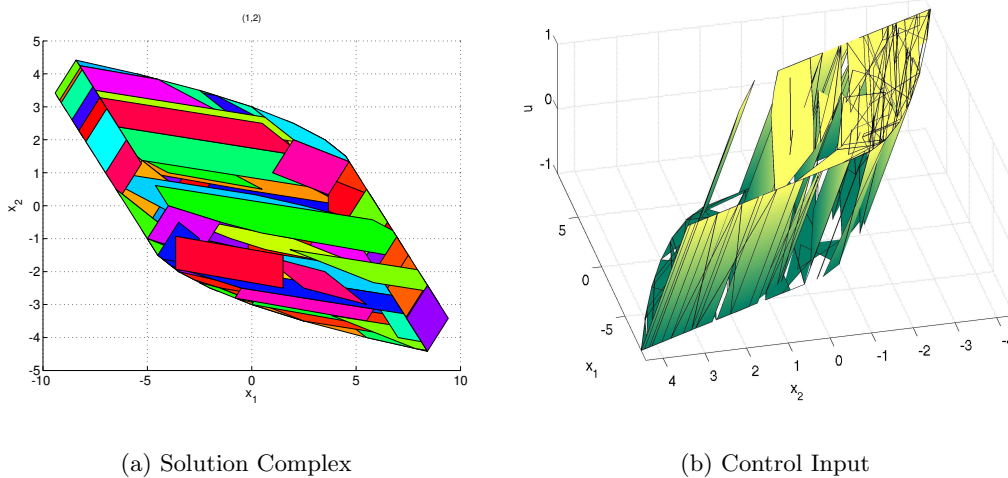


Figure 10.6: RCMSolution for Degenerate Example 10.4

**Basis, Facet, PD, RS (Algs 9.2 through 9.6)** As all of these approaches enumerate the same graph, their outputs are all the same and are shown in Figure 10.8. Note that, as promised, the solution complex is a complex and that the input is continuous.

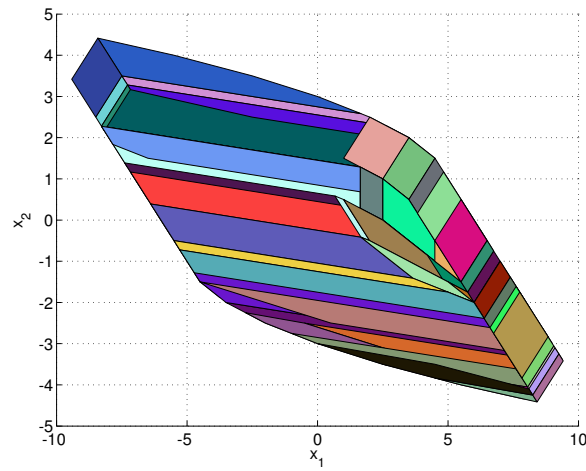


Figure 10.7: RCM Solution for Degenerate Example using Hybrid Toolbox [Bem03]

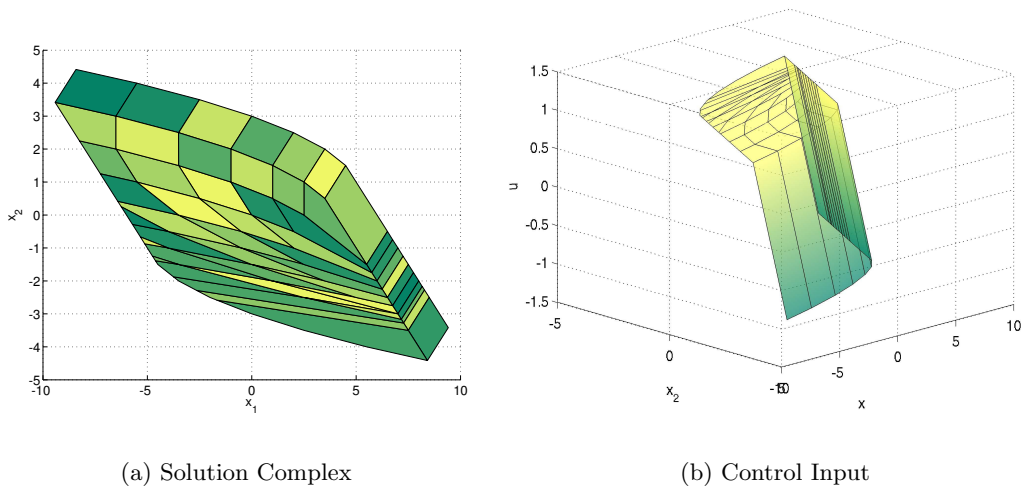


Figure 10.8: Basic Solution (Alg 9.2) for Degenerate Example 10.4

## Part III

# Projection and Parametric Programming



# Chapter 11

## Introduction

In this section we bring together the previous two parts of this thesis and discuss the relationship between projection and multi-parametric linear programming. First, it is shown that given an mpLP problem, a projection algorithm exists whose output provides the solution complex and second, we introduce an mpLP whose dual solution is the projection of a polytope.

Besides the theoretical interest of linking these two algorithms, the importance of this work is obvious: advances in projection algorithms are improvements in mpLP algorithms and *vice versa*. Many problems have a specific structure that means some algorithms are drastically faster than others. As a result of the material given here, a much larger range of both projection and mpLP methods can be searched to find an approach which matches a given problem.

To the best of the author's knowledge, nothing has been published relating these two topics before. However, in a private communication with Prof. D. Klatte<sup>1</sup> it was suggested that Fourier elimination was used to solve rudimentary mpLPs in the 1970s, although no literature could be found on this topic.

### 11.1 Outline

This part consists of three chapters. In Chapter 12, it is shown how to compute parametric linear programs using a projection algorithm. Chapter 13 deals with the opposite direction: an mpLP is formulated whose output provides a desired projection. Finally, Chapter 14 presents comparative simulation results of various tasks that one may put these algorithms

---

<sup>1</sup>Prof. Dr. D. Klatte, Institut für Operations Research, Universität Zürich, Moussonstrasse 15, 8044 Zürich, klatte@ior.unizh.ch

too.

# Chapter 12

## Solving Parametric Linear Programs via Projection

This chapter provides a procedure by which parametric linear programs can be solved using a projection algorithm. While the main result follows readily from the results of Section 9.1, the detail, as always, is in dealing with degeneracy. Specifically, the tool available is any projection algorithm and the aim is to compute all critical regions of the following parametric linear program:

$$\begin{aligned} f(\theta) = \underset{y}{\text{minimise}} \quad & b^T y \\ \text{subject to} \quad & (y, \theta) \in P \end{aligned} \tag{12.1}$$

The definition of the solution complex (Definition 9.5) and Lemma 9.3 show that the solution complex can be computed as follows: First, the polyhedron  $Q$  is defined:

$$Q = \{(\theta, J, y) \mid J \geq b^T y, (y, \theta) \in P\}. \tag{12.2}$$

The supporting hyperplanes of the facets of the epigraph are then computed using a projection operation whose output is the matrix  $G$  and the vector  $g$  (Corollary 9.4):

$$\text{epi}(f) = \pi_{(\theta, J)} Q = \{(\theta, J) \mid \mathbf{1}J \geq G\theta + g, \theta \in \pi_{\theta} P\}. \tag{12.3}$$

**Remark 12.1.** *Note that the projection of  $Q$  will also return the supporting hyperplanes of  $\pi_{\theta} P$ . However, these are easily identified as the coefficient multiplying  $J$  will be zero.*

Finally, the solution complex is given as the projection of the boundary complex of the

epigraph:

$$\mathcal{S}(f) = \{\pi_\theta F \mid F \in \mathcal{B}(\text{epi}(f)) \text{ and } J = f(\theta) \text{ for all } (\theta, J) \in F\}.$$

Recalling that our interest lies only in the critical regions of the solution complex, we have only to focus on the facets of  $\text{epi}(f)$ , and not on the lower dimensional faces. The facets of interest are clearly given by:

$$\text{epi}(f)_i = \{(\theta, J) \mid J = G_i\theta + g_i, \mathbf{1}J \geq G\theta + g, \theta \in \pi_\theta P\},$$

and the associated critical region is then the projection:

$$CR_i = \pi_\theta \text{epi}(f)_i = \{\theta \mid (G - \mathbf{1}G_i)\theta \leq \mathbf{1}g_i - g\} \cap \pi_\theta P. \quad (12.4)$$

If the goal of the algorithm is simply to compute the critical regions, then we are done at this point. However, for most purposes, the primal and/or dual optimisers will be needed and can be found through a post-processing step by computing the optimiser for a point in the strict interior of the region. Such a point can be found by, for example, computing the Chebychev interior, which requires a single linear program [BV04, Sec. 8.4].

Consider the critical region  $CR$  and let  $\theta_\circ$  be a point in its strict interior. The degenerate case will be considered in the next section, but here we assume that the point is non-degenerate. Fixing  $\theta$  to be  $\theta_\circ$  and solving LP (12.1) will give the optimal basis at the point  $\theta_\circ$ . As non-degeneracy has been assumed, this basis will also define the critical region  $CR$  and the primal and dual optimisers by (9.7) and (9.8).

The above post-processing step can clearly be run for each critical region found in (12.4), giving the primal and dual optimisers for every feasible point. The following section considers the situation if the LP (12.1) is degenerate.

## 12.1 Degeneracy

A critical region  $CR$  is dual-degenerate if there is more than one primal optimiser for each  $\theta$  in the interior of  $CR$ . This is equivalent to the dimension of the pre-image being larger than the dimension of the critical region itself  $\dim \pi_\theta^{-1}CR > \dim CR$ . Figure 12.1 depicts this situation for an example multiparametric linear program, which contains only one critical region.

While projection algorithms are certainly capable of computing the epigraph in (12.3) whether the mpLP is degenerate or not, the problem arises if one wants to compute the



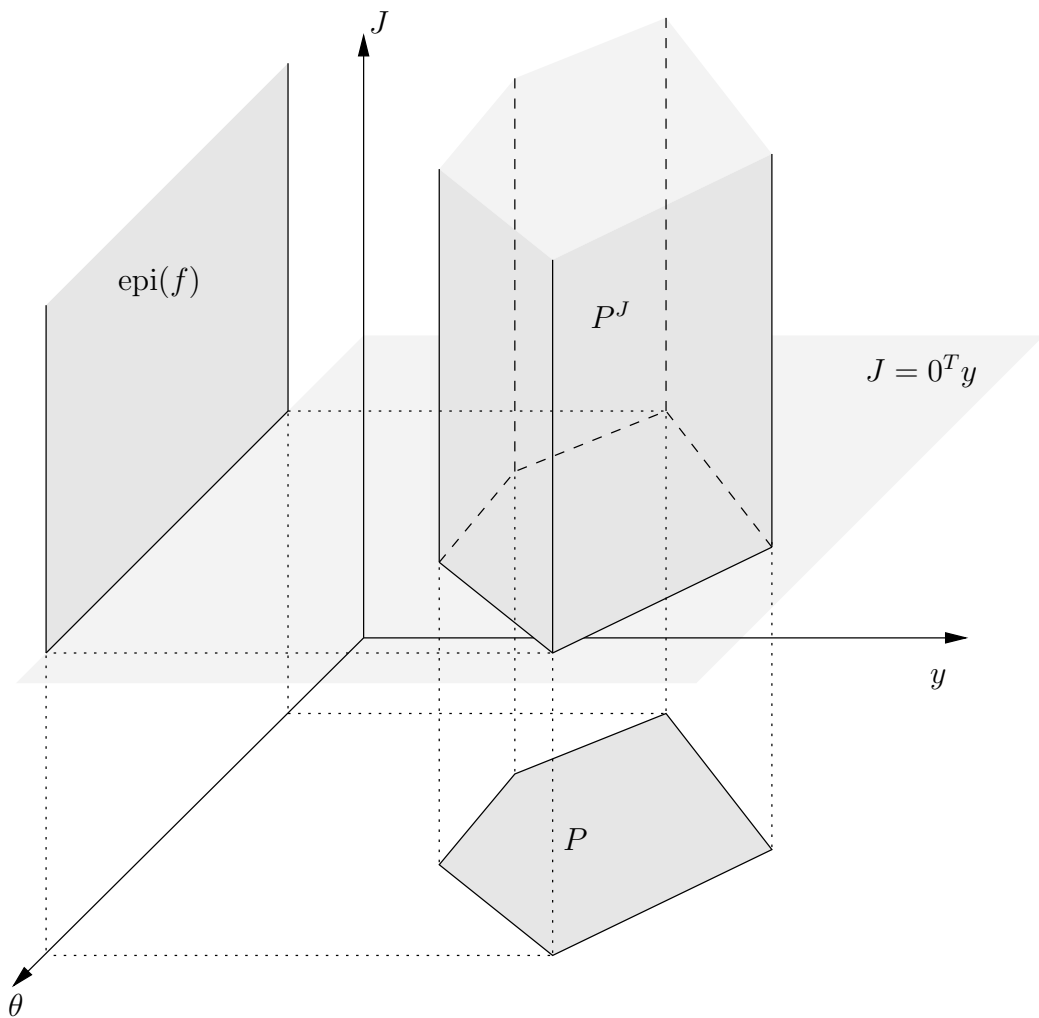


Figure 12.1: Example of a Degenerate Multiparametric Linear Program

primal or dual optimisers. Recall that a primary requirement for many applications is to have either a unique and continuous primal or dual optimiser. The remainder of this section is dedicated to this goal.

**Remark 12.2.** *Recall that a unique primal optimiser is particularly important in closed-form MPC problems as the primal optimiser is the control input.*

First, we discuss how to ensure that the primal optimiser is not dual-degenerate by subdividing the critical region into full-dimensional regions. Let  $CR$  be a dual-degenerate critical region and  $E$  be the equality set of its pre-image;  $CR = \pi_\theta P_E$ . For all parameters  $\theta$  in the critical region  $CR$ , the set of primal optimisers is all points  $y$  such that  $(y, \theta) \in P_E$ . We can now restrict  $(y, \theta)$  to be optimal in terms of mpLP (12.1) and optimise over a secondary cost  $\check{b}$ , which is assumed to be selected such that it is not perpendicular to any edge of  $P$ , and therefore the LP cannot be dual-degenerate. Consider the following multiparametric linear program:

$$\begin{aligned} f(\theta) = \underset{y}{\text{minimise}} \quad & \check{b}^T y \\ \text{subject to} \quad & (y, \theta) \in P_E \end{aligned} \tag{12.5}$$

Note that the feasible region  $\pi_\theta P_E$  is equal to the critical region  $CR$  and that the optimiser of (12.5) is also optimal for (12.1). mpLP (12.5) can now be solved using a projection algorithm as discussed in the previous section. Furthermore, we can be sure that (12.5) will not be dual-degenerate as  $\check{b}$  was chosen to be not perpendicular to any edge

**Remark 12.3.** *Note that if the same  $\check{b}$  is used for every dual-degenerate critical region, then the primal optimiser will be everywhere continuous as discussed in Section 9.2.*

**Remark 12.4.** *The calculation of  $E$  given the critical region  $CR$  is another example of the implicit linearity problem. See Section 5.2 for details on how to solve this problem.*

Finally, some applications demand that there is no primal degeneracy either. This can be ensured quite simply and efficiently during the post-processing step when optimal bases are computed by using a lexicographic LP solver to find the set of active constraints. The approach outlined in this section is given in pseudocode as Algorithm 12.1.

Clearly, the complexity of Algorithm 12.1 is entirely a function of the projection algorithm used. Therefore, if ESP is used, then this approach to computing mpLPs is output sensitive.

**Remark 12.5.** *Note that the lexicographic perturbation prevents primal degeneracy of the primal and the randomly selected vector  $\check{b}$  prevents dual degeneracy, which is the opposite as for the perturbed mpLP problem in Part II.*

---

**Algorithm 12.1** Multiparametric Linear Programming using a Projection Algorithm

---

**Input:** Polytope  $P$  and cost  $b$  defining mpLP (12.1).

**Output:** All bases that define critical regions.

- 1: Compute epigraph:  $\text{epi}(f) = \pi_{(\theta, J)}Q$  Equations 12.2 and 12.3
- 2: **for each** facet  $F$  of  $\text{epi}(f)$  with a non-zero  $J$  coefficient **do**
- 3:     Compute the critical region  $CR = \pi_{\theta}F$  Equation 12.4
- 4:     Choose a parameter  $\theta_{\circ}$  in the strict interior of  $CR$  [BV04, Sec. 8.4]
- 5:     Compute the active constraints  $E$  of  $P^{\epsilon}$   $P^{\epsilon}$  is a lex-perturbed

$$\begin{aligned} & \text{minimise} && b^T y \\ & \text{subject to} && (\theta_{\circ}, y) \in P^{\epsilon} \end{aligned} \tag{12.6}$$

- 6:     **if** LP (12.6) is dual-degenerate **then**
- 7:         Solve and report critical region defining bases of Recursive call

$$\begin{aligned} & \text{minimise} && \check{b}^T y \\ & \text{subject to} && (y, \theta) \in P_E \end{aligned} \tag{12.7}$$

- 8:     **else**
  - 9:         Report  $E$
  - 10:     **end if**
  - 11: **end for**
-



# Chapter 13

## Solving Projection via Parametric Linear Programming

In this chapter we further investigate the relationship between projection and parametric programming by developing a method of computing projections given any multiparametric linear programming tool. Specifically, given a polytope  $P = \{(x, y) \mid Cx + Dy \leq b\}$ , we aim to compute the projection  $\pi_x P$  using an algorithm that can find the critical regions of an mpLP.

We begin with the Projection Lemma, which is often attributed to Černikov [Č63] and can be derived directly from Farkas' Lemma (see, for example [Zie95]).

**Lemma 13.1. (Projection Lemma)** *If  $P = \{(x, y) \mid Cx + Dy \leq b\}$  is a polyhedron, then the projection of  $P$  is:*

$$\pi_x P = \{x \mid v^T Cx \leq v^T b, \forall v \in \text{extr}(W)\} \quad (13.1)$$

where  $\text{extr}(W)$  is the set of extreme rays of the projection cone  $W$ , which is defined as

$$W \triangleq \{v \mid D^T v = 0, v \geq 0\}$$

A standard approach to computing the projection is to first enumerate the rays of the projection cone  $W$ , write down the projection  $\pi_x P$  from (13.1), and then remove redundancies. This approach is not output sensitive because many of the extreme rays of  $W$  generate redundant inequalities of the projection and for many problems, including many of interest to control, there are an exponential number of redundant inequalities generated. The procedure described here uses an mpLP to enumerate only those rays of the projection cone that generate

irredundant inequalities of the projection.

We aim to describe the set of all valid inequalities of the projection, but first a well-known basic result of convexity is needed on the description of valid inequalities.

**Lemma 13.2.** *Let  $P$  be a polyhedron. The halfspace  $h = \{x \mid \alpha^T x \leq \beta\}$ , is valid for  $P$  if and only if there exists a set of valid halfspaces  $\mathcal{H} = \{x \mid \Gamma x \leq \gamma\}$  and a positive vector  $\lambda \geq 0$  such that  $\begin{bmatrix} \alpha^T & \beta \end{bmatrix} = \lambda^T \begin{bmatrix} \Gamma & \gamma \end{bmatrix}$ .*

*Proof.* ( $\Rightarrow$ ) Assume that the halfspace  $h$  is valid. Take  $\mathcal{H} = h$  and  $\lambda = 1$  and the result follows directly.

( $\Leftarrow$ ) Assume the existence of a set of valid halfspaces  $\mathcal{H}$  and a positive vector  $\lambda$  such that  $\begin{bmatrix} \alpha^T & \beta \end{bmatrix} = \lambda^T \begin{bmatrix} \Gamma & \gamma \end{bmatrix}$ .  $\begin{bmatrix} \alpha^T & \beta \end{bmatrix}$  is valid if and only if  $\alpha^T x \leq \beta$  for all  $x$  in  $P$ , which we derive as follows. For each pair  $(\Gamma_i, \gamma_i)$  we have  $\Gamma_i x \leq \gamma_i$  for all  $x \in P$ , which is still true for all positive  $\lambda_i > 0$ :  $\lambda_i \Gamma_i x \leq \lambda_i \gamma_i$ . Finally, the inequality holds under addition:  $\lambda_i \Gamma_i x + \lambda_j \Gamma_j x \leq \lambda_i \gamma_i + \lambda_j \gamma_j$ , which proves the result:  $\forall i, \Gamma_i x \leq \gamma_i$  implies  $\lambda^T \Gamma x \leq \lambda^T \gamma$  implies  $\alpha^T x \leq \beta$ .  $\square$

From Lemmas 13.1 and 13.2 we can see that the set of all valid inequalities of the projection  $\pi_x P$  is given by the set  $S$ :

$$S \triangleq \left\{ (\alpha, \beta) \mid \exists \lambda \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{bmatrix} C^T \\ b^T \end{bmatrix} \lambda, D^T \lambda = 0, \lambda \geq 0 \right\} \quad (13.2)$$

The set  $S$  is what is referred to as a *pointed cone*, that is, a cone with exactly one vertex, the origin. By definition, the extreme rays of  $S$  are exactly those halfspaces of  $\pi_x P$  that cannot be written as a positive combination of other valid halfspaces. It follows that the extreme rays of  $S$  define the irredundant halfspaces of  $\pi_x P$ .

The goal of computing all irredundant inequalities of  $\pi_x P$  is now reduced to finding all extreme rays of  $S$ . As will be seen shortly, the problem of finding all extreme points of a polytope can be posed as a multiparametric linear program and therefore, if we wish to use an mpLP, the cone  $S$  must be bounded. We assume a vector  $a$  such that for every point  $(\alpha, \beta) \neq (\infty, \infty)$  in  $S$  we have  $0 < \begin{bmatrix} \alpha^T & \beta \end{bmatrix} a < \infty$ . The existence of such a vector  $a$  is proven in the following lemma.

**Lemma 13.3.** *Let  $S$  be as defined in (13.2), then there exists a vector  $a$  such that  $0 < \begin{bmatrix} \alpha^T & \beta \end{bmatrix} a < \infty$  if and only if  $a$  is in the pointed cone  $\pi_a \left\{ (a, z) \mid Dz \leq \begin{bmatrix} C & b \end{bmatrix} a \right\}$ .*

*Proof.* The result follows almost immediately from an application of Farkas' Lemma. The

desired property can be written as:

$$\exists a, \text{ such that } \begin{bmatrix} \alpha^T & \beta \end{bmatrix} a > 0, \forall \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{bmatrix} C^T \\ b^T \end{bmatrix} \lambda, D^T \lambda = 0, \lambda \geq 0$$

re-writing gives:

$$\exists a, \text{ such that } \lambda \begin{bmatrix} C & b \end{bmatrix} a > 0, \forall \lambda \lambda D = 0, \lambda \geq 0$$

This statement is now exactly in the form of one of the many variants of Farkas' Lemma (see, for example [Zie95]), which proves that this inequality is true if and only if:

$$Dz < \begin{bmatrix} C & b \end{bmatrix} a$$

□

From Lemma 13.3, it is clear that any element in the strict interior of the homogenisation of  $P$  defines an appropriate vector  $a$ . An appropriate one can be found by, for example, computing the Chebychev center of the homogenisation while restricting the Chebychev radius to one:

$$\begin{aligned} a = \operatorname{argmax}_{z,a,t} \quad & t \\ \text{subject to} \quad & Dz - \begin{bmatrix} C & b \end{bmatrix} a + \left\| \begin{bmatrix} D & -C & -b \end{bmatrix} \right\|_2^2 t \leq 0 \\ & 0 \leq t \leq 1, \end{aligned} \tag{13.3}$$

where the 2-norm is taken row-wise and an appropriate  $a$  exists if and only if  $t > 0$ .

The set  $S$  can now be bounded by the inclusion of the constraint  $\begin{bmatrix} \alpha^T & \beta \end{bmatrix} a = 1$ :

$$\bar{S} \triangleq \left\{ (\alpha, \beta) \mid \exists \lambda \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{bmatrix} C^T \\ b^T \end{bmatrix} \lambda, D^T \lambda = 0, \lambda \geq 0, a^T \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = 1 \right\}$$

As each ray of  $S$  intersects the hyperplane  $\begin{bmatrix} \alpha^T & \beta \end{bmatrix} a = 1$  exactly once, it is clear that there is a one-to-one correspondence between vertices of  $\bar{S}$  and rays of  $S$ .

The following Theorem will allow the extreme points of  $\bar{S}$  to be enumerated via an mpLP.

**Theorem 13.4.**  $x_0 \in \mathbb{R}^n$  is an extreme point of a polyhedron  $P$  if and only if for some vector  $c \in \mathbb{R}^n$  we have  $\max \{ c^T x \mid x \in P \} = c^T x_0 > c^T x$  for all  $x \in P, x \neq x_0$ .

*Proof.* See, for example [Pad99, 7.2(d)].

□

We can see from Theorem 13.4 that if we can compute the maximum for every vector  $c$ , then all vertices will be enumerated. We can therefore pose the following multiparametric linear program in the parameter  $\theta$  to do just this:

$$\begin{aligned} J(\theta) = \underset{\alpha, \beta}{\text{maximise}} \quad & \theta^T \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \\ \text{subject to} \quad & \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \in \bar{S}. \end{aligned} \tag{13.4}$$

The mpLP (13.4) can be re-written in the simpler form:

$$\begin{aligned} J(\theta) = \underset{\lambda}{\text{maximise}} \quad & \left( \begin{bmatrix} C & b \end{bmatrix} \theta \right)^T \lambda \\ \text{subject to} \quad & D^T \lambda = 0, \\ & a^T \begin{bmatrix} C^T \\ b^T \end{bmatrix} \lambda = 1 \\ & \lambda \geq 0 \end{aligned} \tag{13.5}$$

Note also that the condition  $c^T x_0 > c^T x$  from Theorem 13.4 is satisfied if and only if the linear program  $\max \{c^T x \mid x \in P\}$  is not dual-degenerate. Note that the solution of an mpLP is dual-degenerate in every region of the solution complex, other than in the interior of the critical regions. It follows that there is a one-to-one mapping from vertices of  $\bar{S}$  to the critical regions of the above mpLP. Therefore, we can compute the projection by enumerating all of the critical regions of the above mpLP.

This approach to computing projections using a multiparametric linear program is clearly output sensitive if and only if the mpLP algorithm is. Pseudocode for the proposed approach is given as Algorithm 13.1.

**Remark 13.5.** *As promised in Section 9.4, we have now also shown that all mpLPs of the form (7.1) can be written in the form (9.17), i.e. with  $c = 0$ . This follows from Chapter 12 in which it was shown that these mpLPs can be solved via a projection, and in this section that all projections can be solved by an mpLP of the appropriate form.*

**Remark 13.6.** *Note that while each extreme ray of  $S$  defines a non-redundant inequality of the projection, the mpLP (13.5) may return a particular ray more than once. If a ray is primal-degenerate then the lexicographic perturbation will cause it to be found several times as there will be several lex-perturbed rays corresponding to the single ‘true’ ray. This form of redundancy is trivial to detect and remove as the projection simply needs to be tested for duplicate inequalities and therefore no linear programs are required.*



---

**Remark 13.7.** *No assumption has been made on whether the polyhedron  $P$  is bounded in any of the above discussion. As the projection lemma holds for general polyhedra, the approach described in this section will also.*

---

**Algorithm 13.1** Projection using a Multiparametric Linear Programming Algorithm

---

**Input:** Polytope  $P = \{(x, y) \mid Cx + Dy \leq b\}$ .

**Output:** Irredundant valid inequalities of  $\pi_x P$

- 1: Solve the mpLP (13.5)
  - 2: **for each** critical region-defining basis  $B$  of (13.5) **do**
  - 3:   Let  $\lambda^*$  be the optimiser of (13.5) for basis  $B$  (9.8)
  - 4:   Report valid, irredundant constraint:  $\lambda^* Cx \leq \lambda^* b$  (13.2), Lemma 13.2
  - 5: **end for**
- 

It is clear that the complexity of this approach is entirely dependant on the complexity of the mpLP used to solve (13.5). Therefore, if any of the approaches discussed in Chapter 9 are used, then this will be an output sensitive projection algorithm for non-degenerate problems.



# Chapter 14

## Projection and mpLP Examples

In this chapter we will examine, through simulation, the use of projection algorithms for solving multi-parametric linear problems and *vice versa*.

### 14.1 Solving Parametric Linear Programs via Projection

In this section some of the examples of Part II will be computed using the projection algorithms discussed in Part I.

#### 14.1.1 Double Integrator

We first revisit the double integrator of Example 10.1, in which the goal was to compute the closed-form MPC solution. The resulting mpLP is:

$$\begin{aligned} f(x) = \underset{U,S,T}{\text{minimise}} \quad & \mathbf{1}^T S + \mathbf{1}^T T \\ \text{subject to} \quad & -\gamma \leq \begin{bmatrix} \Lambda & \Sigma \end{bmatrix} \begin{pmatrix} x \\ U \end{pmatrix} \leq \gamma \\ & -\Psi \begin{pmatrix} T \\ S \end{pmatrix} \leq \begin{bmatrix} \Lambda & \Sigma \end{bmatrix} \begin{pmatrix} x \\ U \end{pmatrix} \leq \Psi \begin{pmatrix} T \\ S \end{pmatrix} \\ & S \geq 0, T \geq 0 \end{aligned} \tag{14.1}$$

where we have defined the following matrices for simplicity:

$$\begin{aligned}
 \gamma &\triangleq [10 \ 10 \ 10 \ 10 \ 2 \ 2]^T & \Lambda &\triangleq \begin{bmatrix} 2 & 0 & 2 & 0 & 0 & 0 \\ 2 & 2 & 4 & 2 & 0 & 0 \end{bmatrix}^T \\
 \Sigma &\triangleq \begin{bmatrix} 2 & 1 & 3 & 1 & 2 & 0 \\ 0 & 0 & 2 & 1 & 0 & 2 \end{bmatrix}^T & \Psi &\triangleq \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T
 \end{aligned} \tag{14.2}$$

Equation 12.3 can now be used to formulate the mpLP as a projection:

$$\begin{aligned}
 \text{epi}(f) &= \pi_{(x,J)}Q \\
 &= \pi_{(x,J)} \left\{ (x, J, U, T, S) \left| \begin{array}{l} \begin{bmatrix} -\Gamma & 0 \\ \Gamma & 0 \\ -\Gamma & 0 \\ \Gamma & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} x \\ J \end{pmatrix} + \begin{bmatrix} \Sigma & 0 \\ -\Sigma & 0 \\ \Sigma & -\Psi \\ -\Sigma & -\Psi \\ 0 & -I \\ 0 & \mathbf{1}^T \end{bmatrix} \begin{pmatrix} U \\ T \\ S \end{pmatrix} \leq \begin{bmatrix} \gamma \\ \gamma \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \right. \right\}
 \end{aligned}$$

The polytope  $Q$  is in  $\mathbb{R}^9$ , contains 25 constraints and the projection yields the epigraph in  $\mathbb{R}^3$ , which has 20 inequalities of which 8 define the boundaries of the feasible region and the remaining 12 define critical regions. Six of the regions of this example are degenerate, and require a second projection for each region in order to compute a unique optimiser. Of course, if the ESP degeneracy handling method of Section 5.1 is used then this will be unnecessary.

The resulting projection is shown in Figure 14.1.

### 14.1.2 Random 3D System

In this example we consider the random three dimensional system with two inputs from Example 10.2. Posing this as a projection problem requires a projection from  $\mathbb{R}^{24}$  to  $\mathbb{R}^4$  of a polytope with 87 constraints. Table 14.1 compares the time taken by the various projection and mpLP methods. Note that even this small example is too taxing for most existing projection methods.

## 14.1 MPLP VIA PROJECTION

---

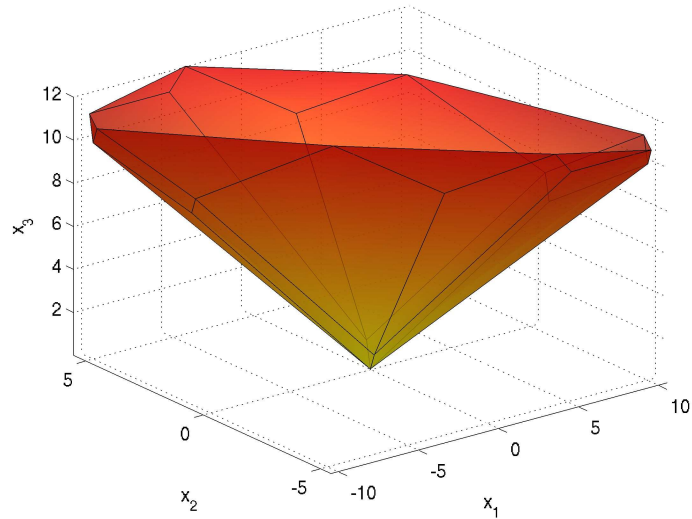


Figure 14.1: Projection of  $Q$  for Example 14.1.1

mpLP Method	Time (secs)	Projection Method	Time (secs)
Basic	41.6	ESP	38.95
Facet	29.5	VpH (CDD)	> 24 hours
PD	333.7	Ray (CDD)	> 24 hours
RS	359.9	Fourier	> 24 hours
FTM	631.5		
RCM	794.5		

Table 14.1: Comparison of mpLP Methods for Example 10.2

## 14.2 Solving Projection via Parametric Linear Programming

In this section we will return to some of the examples of Part I and re-solve the projections using the mpLP algorithms of Part II.

### 14.2.1 Double Integrator

Consider again the calculation of the feasible region for the double integrator of Example 6.2.1:

$$\mathbb{X}_F = \pi_x \left\{ (x, U) \mid \begin{array}{l} \left( \begin{array}{c} 10 \\ 10 \\ 10 \\ 10 \\ 2 \\ 2 \end{array} \right) - \left( \begin{array}{c} 10 \\ 10 \\ 10 \\ 10 \\ 2 \\ 2 \end{array} \right) \leq \left[ \begin{array}{cccc} 2 & 2 & 2 & 0 \\ 0 & 2 & 1 & 0 \\ 2 & 4 & 3 & 2 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{array} \right] \begin{pmatrix} x \\ U \end{pmatrix} \leq \left( \begin{array}{c} 10 \\ 10 \\ 10 \\ 10 \\ 2 \\ 2 \end{array} \right) \right\} \quad (14.3)$$

Using the definitions of (14.2), the projection (14.3) can now be written in standard form as:

$$\mathbb{X}_F = \pi_x \left\{ (x, U) \mid \left[ \begin{array}{c} \Lambda \\ -\Lambda \end{array} \right] x + \left[ \begin{array}{c} \Sigma \\ -\Sigma \end{array} \right] U \leq \left[ \begin{array}{c} \gamma \\ \gamma \end{array} \right] \right\}$$

Using the results of Chapter 13, this projection can be re-written as the following mpLP:

$$\begin{aligned} & \text{maximise} && \left( \left[ \begin{array}{cc} \Gamma & \gamma \\ -\Gamma & \gamma \end{array} \right] \theta \right)^T \lambda \\ & \text{subject to} && \left[ \begin{array}{cc} \Sigma^T & -\Sigma^T \end{array} \right] \lambda = 0 \\ & && a^T \left[ \begin{array}{cc} \Lambda^T & -\Lambda^T \\ \gamma^T & \gamma^T \end{array} \right] \lambda = 1 \\ & && \lambda \geq 0 \end{aligned} \quad (14.4)$$

where  $a$  is computed via LP (13.3) and is found to be:

$$a = \left[ 0.7 \ 0.7 \ 1.9 \ 2.5 \ 1.9 \ 3.5 \ 0.7 \ 0.7 \ 0.9 \ 2.2 \ 1.4 \ 3.2 \right]^T$$

As discussed in Chapter 13, the vertices  $(\alpha_i, \beta_i)$  of the polytope  $E^T D$  define the halfspace inequalities  $\alpha_i x \leq \beta_i$  of the projection  $\pi_x P$ . By the inclusion of the bounding constraint, all of the vertices lie on the hyperplane  $\left[ \alpha^T \ \beta \right] a = 1$  and therefore the dimension of  $E^T D$  is

## 14.2 PROJECTION VIA MPLP

---

two, although it is contained in  $\mathbb{R}^3$ . To facilitate plotting, we normalise all of the vertices by dividing through by  $\beta_i$ . This can be done as the feasible set  $\mathbb{X}_F$  is bounded and contains the origin, and therefore all of the vertices have a  $\beta_i$  that is strictly positive. Shown in Figure 14.2(a) is the polytope  $E^T D$ . Note that this polytope is generally referred to as the *polar dual* of the projection in the literature.

**Remark 14.1.** *Note that the polytope  $E^T D$  is the polar dual of the homogenisation of the projection  $\pi_x P$  intersected with the cut-plane  $T = \{z \mid a^T z = 1\}$ , where  $T$  was defined with the requirement that all rays of the polar dual of  $E^T D$  pass through it. See [Zie95] for details on the polar dual and the homogenisation.*

Recalling (9.18), we know that if  $c = 0$ , the mpLP can be re-written as  $\min \{\theta^T z \mid z \in E^T D\}$ . It follows that for each vertex  $v$  of  $E^T D$ , there will be some set of directions  $\theta$  such that  $v$  is the optimiser. From the definition, this set is the *normal cone* of the vertex  $v$  and it is clear that it is also the critical region associated with  $v$ . It is clear that if all of the vertices of  $E^T D$  are considered, then the *normal fan* (set of all normal cones) will result. The solution complex, which is also the normal fan of  $E^T D$ , is shown in Figure 14.2(b).

Each normal fan is drawn originating from its associated vertex in Figure 14.2(c). Primal degeneracy of the dual appears as more than one cone attached to a vertex. This is because the point has been lexicographically perturbed and is therefore, from the point of view of the algorithm, several points. It is clear, however, that degeneracy of this type makes the algorithm no longer output sensitive, as more than one critical region needs to be enumerated in order to determine a single inequality of the projection.

Finally, the projection itself is shown as Figure 14.2(d).

### 14.2.2 Random 3D System

We now move onto the second and final example in this section, where we re-visit the feasibility calculation of Example 6.2.2. The computation of the feasible set in this example requires the projection of a polytope in  $\mathbb{R}^{13}$  containing 80 constraints. The resulting set  $\mathbb{X}_F$  is in  $\mathbb{R}^3$  and consists of only 16 halfspaces.

Each of the mpLP algorithms discussed in Part II has been used to compute the projection and the results are collected in Table 14.2, where they are compared against the computation times for the known projection algorithms.

**Remark 14.2.** *Note that the methods FTMan and RCM cannot compute parametric programs in which the primal feasible set is unbounded. In order to make this problem calculable, constraints were added to restrict the primal optimiser to lie in the unit cube.*

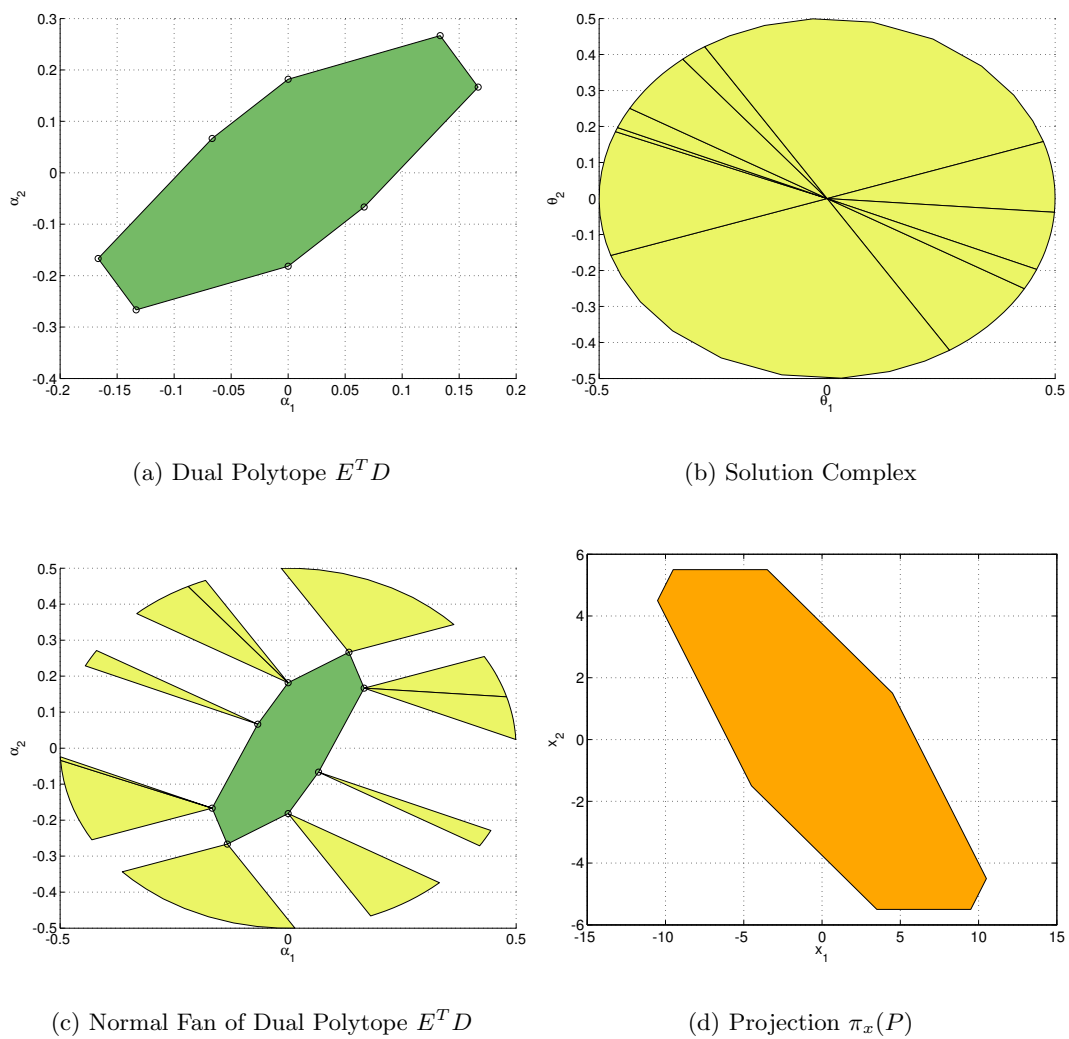


Figure 14.2: Solution for Example 14.2.1



## 14.2 PROJECTION VIA MPLP

---

**Remark 14.3.** *It should be noted that this projection is quite degenerate and as such the method FTM cannot be guaranteed to find the correct solution, as shown in Figure 14.3. In practice, it has been observed that for small problems, the correct projection is generally returned, however, the number of critical regions explored is often much higher than expected.*

The solution complex for this example is shown in Figure 14.4 and the dual polytope  $E^T D$ , with its associated normal fan, in Figure 14.5. Finally, the projection is shown in Figure 14.6(a) and its polar dual  $E^T D$  in Figure 14.6(b).

Method	Time (secs)
ESP	0.22
Fourier Elimination	30.2 – 20739.0
VpH	
CDD	4186.7
lrs	455.47
qhull	391.0
Projection Cone	
CDD	9327.7
lrs	> 12 hours
qhull	> 2GB RAM
mpLP	
Basic	3.69
Facet	3.37
PD	4.87
RS	22.45
FTM	12.19
RCM	23.71

Table 14.2: Comparison of Projection and mpLP Methods for the Calculation of the Feasible Region of Example 14.2.2

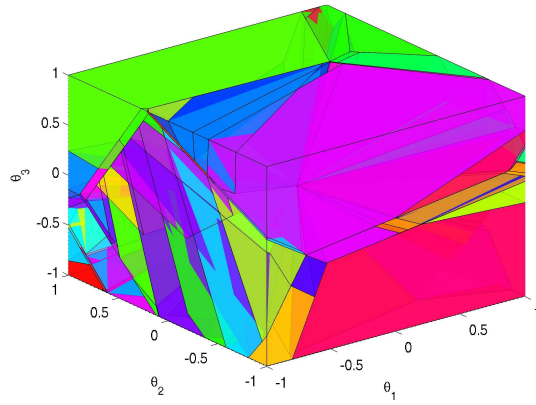


Figure 14.3: Critical Regions found by MPT for Example 14.2.2

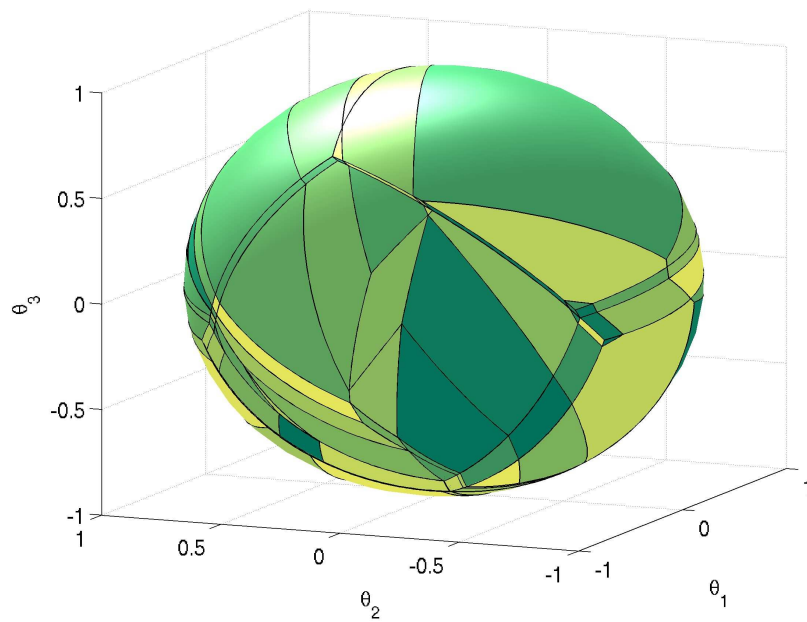


Figure 14.4: Solution Complex for Example 14.2.2

## 14.2 PROJECTION VIA MPLP

---

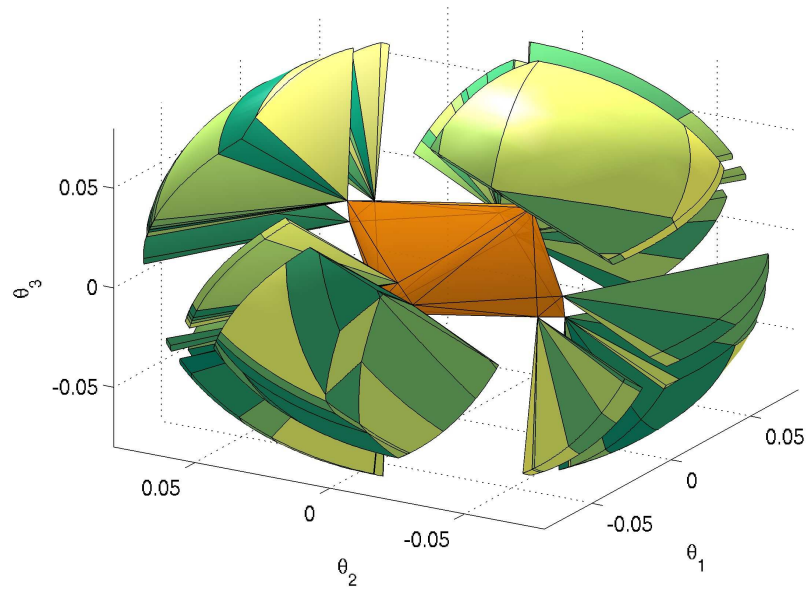
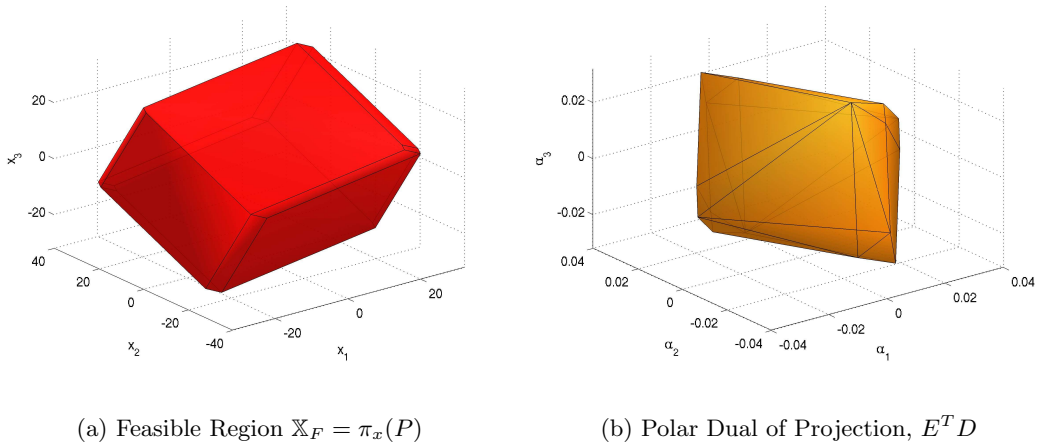


Figure 14.5: Normal Fan and Dual Polytope  $E^T D$  for Example 14.2.2



(a) Feasible Region  $\mathbb{X}_F = \pi_x(P)$

(b) Polar Dual of Projection,  $E^T D$

Figure 14.6: Solution to Example 14.2.2



## Part IV

# Point Location Problem



# Chapter 15

## Introduction

It is standard practice to implement an MPC controller by solving on-line an optimal control problem that, when the system is linear and the constraints are polyhedral, amounts to computing a single linear or quadratic program at each sampling instant depending on the type of control objective. In recent years, it has become well-known that the optimal input is a piecewise affine function (PWA) defined over a polyhedral partition of the feasible states [Bor03]. Methods of computing this affine function can be found in Parts II and III of this thesis as well as in several sources in the literature (e.g., [TJB01, BMDP02, Bor03]). The on-line calculation of the control input then becomes one of determining the region that contains the current state and is known as the *point location problem*.

The complexity of calculating this function is clearly dependent on the number of affine regions in the solution. This number of regions is known to grow very quickly and possibly exponentially, with horizon length and state/input dimension [BMDP02]. The complexity of the solution therefore implies that for large problems an efficient method for solving the point location problem is needed.

The key contributions to this end have been made by Töndel, *et al* [TJB03] and Borelli, *et al* [BBBM01]. In [TJB03], the authors propose to construct a binary search tree over the polyhedral state-space partition. Therein, auxiliary hyper-planes are used to subdivide the partition at each tree level. Note that these auxiliary hyper-planes may subdivide existing regions. The necessary on-line identification time is logarithmic in the number of subdivided regions, which may be significantly larger than the original number of regions. Although the scheme works very well for smaller partitions, it is not applicable to large controller structures due to the prohibitive pre-processing time. If  $R$  is the number of regions and  $\bar{F}$  the average number of facets defining a region, then the approach requires the solution to  $R^2 \cdot \bar{F}$  LPs<sup>1</sup>.

---

<sup>1</sup>It is possible to improve the pre-processing time at the cost of less efficient (non-logarithmic) on-line

However, the scheme in [TJB03] is applicable to *any* type of *closed-form* MPC controller, whereas the algorithm proposed in this thesis considers only the case in which controllers are obtained via a *linear* cost. The approach proposed here is not directly applicable to non-convex controller partitions and can only be applied to controllers obtained for a *quadratic* cost if the solution exhibits a specific structure.

In [BBBM01] the authors exploit the convexity properties of the piecewise affine (PWA) value function of linear MPC problems to solve the point location problem efficiently. Instead of checking whether the point is contained in a polyhedral region, each affine piece of the value function is evaluated for the current state. Since the value function is PWA and convex, the region containing the point is associated to the affine function that yields the largest value. Although this scheme is efficient, it is still linear in the number of regions.

In this part, we combine the concept of region identification via the value-function [BBBM01] with the construction of search trees [TJB03], by using the link between *parametric* linear programming, Voronoi Diagrams and Delaunay triangulations, recently established in [RGJ04]. We demonstrate that the PWA cost function can be interpreted as a weighted power diagram, which is a type of Voronoi diagram, and exploit recent results in [AMN<sup>+</sup>98] to solve the point location problem for Voronoi diagrams in logarithmic time at the cost of very simple pre-processing operations on the controller partition.

We focus on MPC problems with 1- or  $\infty$ -norm objectives and show that evaluating the optimal PWA function for a given state can be posed as a nearest neighbour search over a finite set of points. In [AMN<sup>+</sup>98] an algorithm is introduced that solves the nearest neighbour problem in  $d$  dimensions with  $R$  regions in time  $\mathcal{O}(c_{d,\epsilon} n \log R)$  and space  $\mathcal{O}(dR)$  after a pre-processing step taking  $\mathcal{O}(dR \log R)$ , where  $c_{d,\epsilon}$  is a factor depending on the state dimension and an error tolerance  $\epsilon$ . Hence, the optimal control input can be found on-line in time logarithmic in the number of regions  $R$ .

## 15.1 Outline

The remainder of this part is organised as follows. In section 16.1, the problem addressed in this part is formally defined. Section 16.2 demonstrates that the point location problem can be posed as a nearest neighbour search over  $R$  points. Section 16.4 provides a brief overview of the logarithmic nearest neighbour algorithm from [AMN<sup>+</sup>98]. Finally, Chapter 17 provides numerical examples and compares the approach to the current state of the art.

---

computation times.



### 15.2 Acknowledgements

The work presented in this part is a collaborative effort with Pascal Grieder and Sasa Raković. The text is based almost entirely on the following paper:

C.N. Jones, P. Grieder and S.V. Raković. A Logarithmic-Time Solution to the Point Location Problem for Closed-Form Linear MPC. To appear in *Proceedings of the 16<sup>th</sup> IFAC World Congress*, Prague, Czech Republic, 2005.

The examples in this part have been prepared with the MPT toolbox [KGBM04] and Figure 17.3 was calculated using the ANN library [MA98].



# Chapter 16

## Logarithmic Point Location

### 16.1 Introduction

Consider the following multiparametric linear program:

$$\begin{aligned}
 f(\theta) \triangleq \min_y \quad & b^T y \\
 \text{subject to} \quad & (y, \theta) \in P.
 \end{aligned}
 \tag{16.1}$$

Given a particular parameter  $\theta \in \mathbb{R}^d$ , the goal is to determine the primal and/or dual optimiser of mpLP (16.1) for  $\theta$ . This can be done by determining which critical region  $CR$  of the solution complex  $\mathcal{S}(f)$  contains the given parameter  $\theta$ . Once the critical region has been found, the primal and/or dual optimisers can be computed directly from the optimal basis as seen in (9.7) and (9.8).

The point location problem can therefore be stated as:

**Definition 16.1. (Point Location Problem)** *Given a vector  $\theta$  and the solution complex  $\mathcal{S}(f) = \{\mathcal{R}_1, \dots, \mathcal{R}_R\}$  of the mpLP (16.1), determine any integer<sup>1</sup>  $i(\theta) \in \{1, \dots, R\}$  such that polytope  $\mathcal{R}_{i(\theta)}$  contains  $\theta$ .*

From Corollary 9.4, the epigraph of  $f$  can be written as:

$$\text{epi}(f) = \{(\theta, J) \mid \mathbf{1}J \geq G\theta + g, \theta \in \pi_\theta P\}$$

and from the definition of the solution complex (Definition 9.5), there is a one-to-one mapping from the facets of  $\{(\theta, J) \mid \mathbf{1}J \geq G\theta + g\}$  to the critical regions of  $\mathcal{S}(f)$ . It follows that  $i(\theta)$

---

<sup>1</sup>The state may be on the boundary of several regions.

can be computed as [BBBM01]:

$$i(\theta) = \operatorname{argmax}_{r \in \{1, \dots, R\}} \{G_r \theta + g_r\}. \quad (16.2)$$

As was proposed in [BBBM01],  $i(\theta)$  can be computed from (16.2) by simply evaluating the cost  $G_r \theta + g_r$  for each  $r \in \{1, \dots, R\}$  and then taking the largest. This procedure requires  $2dR$  flops and has a storage requirement of  $(d+1)R$ , where  $d$  is the dimension of the parameter.

In the following sections we will show that with a negligible pre-processing step, (16.2) can be computed in *logarithmic* time, which is a significant improvement over the *linear* time result of [BBBM01].

## 16.2 Point Location and Nearest Neighbours

In this section we show that for mpLPs, the point location problem can be written as an *additively weighted nearest neighbour search*, or a search over  $R$  points in  $\mathbb{R}^d$  to determine which is closest to the parameter  $\theta$ .

Consider the finite set of points called *sites*  $\mathcal{S} \triangleq \{s_1, \dots, s_R\}$  and the weights  $\mathcal{W} \triangleq \{w_1, \dots, w_R\}$ , where  $(s_i, w_i) \in \mathbb{R}^d \times \mathbb{R}$ ,  $\forall i \in \mathbb{N}_R$  (recall that  $\mathbb{N}_R = \{1, \dots, R\}$ ). Given a point  $\theta$  in  $\mathbb{R}^d$ , the weighted nearest neighbour problem is the determination of the point  $s_r \in \mathcal{S}$  that is closest to  $\theta$ , for all  $(s_j, w_j) \in \mathcal{S} \times \mathcal{W}$ ,  $j \in \{1, \dots, R\}$ . Associated with each site is a set of points  $\mathcal{L}_r \subset \mathbb{R}^d$  such that for each  $\theta \in \mathcal{L}_r$ ,  $\theta$  is closer to  $s_r$  than to any other site:

$$\mathcal{L}_r \triangleq \{\theta \mid \|s_r - \theta\|_2^2 + w_r \leq \|s_j - \theta\|_2^2 + w_j, \forall j \in \mathbb{N}_R\}. \quad (16.3)$$

Note that the sets  $\mathcal{L}_r$  form a complex  $\mathcal{C}_V \triangleq \{\mathcal{L}_1, \dots, \mathcal{L}_R\}$  [Aur91]. If the weights  $w_r$  are all zero, then the sets  $\mathcal{L}_r$  form a *Voronoi diagram*, otherwise they are called a *power diagram* [Aur91]. An example Voronoi diagram is shown in Figure 16.1 for a random set of sites.

We now state the main result that enables a logarithmic search time:

**Theorem 16.2.** *If  $\mathcal{C}$  is a solution complex, then  $\mathcal{C}$  is the intersection of a power diagram with the feasible region of mpLP (16.1).*

*Proof.* It suffices to show that for any solution complex of mpLP (16.1),  $\mathcal{C} \triangleq \{\mathcal{R}_1, \dots, \mathcal{R}_R\}$ , it is possible to define a set of sites and weights such that their power diagram is equivalent to  $\mathcal{C}$ .

## 16.2 POINT LOCATION AND NEAREST NEIGHBOURS

---

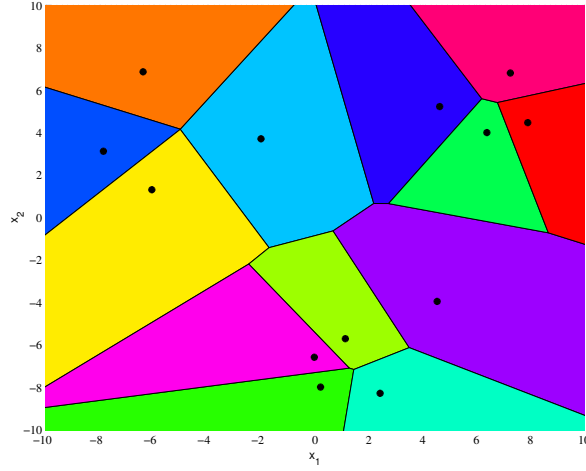


Figure 16.1: Example of a Random Voronoi Diagram

It follows from the definition of the solution complex (Definition 9.5) and from Corollary 9.4 that  $\theta \in \pi_\theta P$  is contained in critical region  $\mathcal{R}_r$  if and only if

$$G_r \theta + g_r \geq G_j \theta + g_j, \quad \forall j \in \mathbb{N}_R,$$

or equivalently, if and only if:

$$-G_r x - g_r \leq -G_j x - g_j, \quad \forall j \in \mathbb{N}_R.$$

Define the  $R$  sites and weights as:

$$\begin{aligned} s_r &\triangleq \frac{G_r^T}{2} \\ w_r &\triangleq -g_r - \left\| \frac{G_r}{2} \right\|_2^2 = -g_r - \|s_r\|_2^2 \end{aligned} \tag{16.4}$$

For all  $r \in \{1, \dots, R\}$  and a given  $\theta$  it follows that:

$$\|s_r - \theta\|_2^2 + w_r = -G_r \theta - g_r + \|\theta\|_2^2$$

Recalling the definition of  $\mathcal{L}_r$  in (16.3) we obtain the following  $\forall j \in \mathbb{N}_R$ :

$$\begin{aligned}
\mathcal{L}_r &\triangleq \left\{ \theta \mid \|s_r - \theta\|_2^2 + w_r \leq \|s_j - \theta\|_2^2 + w_j, \right\} \\
&= \left\{ \theta \mid -G_r\theta - g_r + \|\theta\|_2^2 \leq -G_j\theta - g_j + \|\theta\|_2^2, \right\} \\
&= \{ \theta \mid -G_r\theta - g_r \leq -G_j\theta - g_j, \} \\
&= \{ \theta \mid G_r\theta + g_r \geq G_j\theta + g_j, \}
\end{aligned} \tag{16.5}$$

We see from (16.5) and Corollary 9.4 that

$$\mathcal{L}_r \cap \pi \{P\} = \mathcal{R}_r.$$

Thus the equivalence of the power diagram of the set of sites and weights (16.4) and the solution complex  $\mathcal{C}$  of a corresponding mpLP is established.  $\square$

A very important consequence of Theorem 16.2 is that the point location problem (16.2) can be solved by determining which site  $s_r$  is closest to the parameter  $\theta$ :

$$\begin{aligned}
i(\theta) &= \left\{ r \mid \|s_r - \theta\|_2^2 + w_r \leq \|s_j - \theta\|_2^2 + w_j, \forall j \in \mathbb{N}_R \right\} \\
&= \min_{r \in \{1, \dots, R\}} \left\| \begin{pmatrix} s_r \\ \sqrt{w_r} \end{pmatrix} - \begin{pmatrix} \theta \\ 0 \end{pmatrix} \right\|_2^2
\end{aligned}$$

Since this problem has been well studied in the computational geometry literature we propose to adapt an efficient algorithm introduced in [AMN<sup>+</sup>98] that solves the nearest neighbour problem in *logarithmic time* and thereby solves the point location problem in *logarithmic time*. Section 16.4 will give a brief introduction to the algorithm introduced in [AMN<sup>+</sup>98].

**Remark 16.3.** In [Aur87] it was shown that a complex is a power diagram if and only if there exists a piecewise affine, continuous and convex function in  $\mathbb{R}^{d+1}$  such that the projection of each affine piece of the function from  $\mathbb{R}^{d+1}$  to  $\mathbb{R}^d$  is a cell in the complex. This piecewise affine function is called a *lifting* of the complex. From the proof of Theorem 16.2, it is clear that the solution complex of every mpLP has a *lifting*.

**Remark 16.4.** If the parametric program has a quadratic cost, rather than linear, then the resulting solution complex may or may not have a *lifting*. Although it is not difficult to find problems for which a *lifting* does not exist, general conditions for the existence of a *lifting* for quadratic costs are not known. See [Aur91, Ryb99] for details on testing when a complex has an appropriate *lifting*.

## 16.3 Degeneracy

As seen in Part II, if a critical region is degenerate, then a perturbation can be used to remove this degeneracy and select a unique basis. The result of this is that there will be several critical regions with the same minimal cost. In other words, a facet of the epigraph will project onto the union of several critical regions. Clearly, the method described above can only be used to determine which facet of the epigraph a parameter is in and not the critical region if there is degeneracy.

However, once the facet of the epigraph has been determined, the approach can be applied a second time on the complex formed within the degenerate region by the perturbed cost in order to find the appropriate basis.

## 16.4 Approximate Nearest Neighbour: Logarithmic Solution

In this section, the key aspects of the approximate nearest neighbour search algorithm presented in [AMN<sup>+</sup>98] will be summarised. Given a point  $q \in \mathbb{R}^d$ , a positive real  $\epsilon$  and a set of  $R$  points in  $\mathbb{R}^d$ , the point  $p$  is a  $(1 + \epsilon)$ -approximate nearest neighbour of  $q$ , if its distance from  $q$  is within a factor of  $(1 + \epsilon)$  of the distance from the true nearest neighbour.

**Remark 16.5.** *The  $\epsilon$  error is required in order to prove the logarithmic search time [AMN<sup>+</sup>98]. If the method proposed in Part II is used, then the primal optimiser is continuous and this error in determining the region translates into a maximum error in the primal optimiser that is proportional to  $\epsilon$ . Therefore, the error in the optimiser can be made arbitrarily small with an appropriate selection of  $\epsilon$ .*

As shown in [AMN<sup>+</sup>98], it is possible to pre-process the  $R$  data points in  $\mathcal{O}(dR \log R)$  time and  $\mathcal{O}(dR)$  space, such that the approximate nearest neighbour can be identified in  $\mathcal{O}(c_{d,\epsilon} \log R)$  time, where  $c_{d,\epsilon}$  is a factor depending only on state-space dimension  $d$  and accuracy  $\epsilon$ .

The authors in [AMN<sup>+</sup>98] propose to construct a so-called *balanced box-decomposition tree* or BBD-tree. The BBD-tree is a hierarchical decomposition of the state-space into hyperrectangles (cells) whose sides are orthogonal to the coordinate axes. The BBD tree has two key properties which are vital in obtaining the logarithmic runtime bounds. Namely, as one descends the BBD-tree, the number of points associated to each cell decreases exponentially *and* the aspect ratio (ratio of longest to shortest side of each cell) is bounded by a constant.

The BBD-tree is constructed through the repeated application of two operations, *splits* and *shrinks*. A *split* subdivides a cell into two equally sized *children* by adding an axis-orthogonal hyperplane. This operation guarantees the exponential decrease in the number

of points associated to each cell but it cannot give bounds on the aspect ratio. The *shrink*, partitions a cell into two subcells by using a hyper-rectangle that is located in the interior of the parent cell. The shrink operation corresponds to ‘zooming in’ to regions where points are highly clustered. A simple strategy to construct the BBD-tree is to apply splits and shrinks alternately. This procedure is repeated until the number of points associated to each cell is at most one.

In order to describe the on-line search, we will introduce the following definition: the *distance* between a point  $q$  and a cell is the closest distance between  $q$  and any part of the cell. Given a query point  $q$ , the algorithm first identifies the associated leaf cell by a simple descent through the tree in  $\mathcal{O}(\log R)$  time. It is possible to enumerate the  $s$  cells closest to  $q$  in increasing order in  $\mathcal{O}(sd \log R)$  time [AMN<sup>+</sup>98]. The necessary number of cells  $s$  is bounded by a constant which can be determined without constructing the BBD-tree [AMN<sup>+</sup>98]. Each cell is then visited (closest cell first) and the closest point seen so far is stored as  $p$ . As soon as the distance from a cell to  $q$  exceeds  $\text{dist}(p, q)/(1 + \epsilon)$ , it follows that the search can be terminated and  $p$  can be reported as the approximate nearest neighbour [AMN<sup>+</sup>98].

In this chapter, we have shown that the point location problem can be posed as a weighted nearest neighbour search. By using existing results from computer science, this equivalence allows the point location problem to be solved in logarithmic time.



# Chapter 17

## Point Location Examples

In this chapter we consider various systems and compare the on-line calculation times of the method proposed in this thesis to the scheme in [BBBM01]. Although the scheme in [TJB03] may lead to more significant runtime improvements than [BBBM01], the necessary pre-processing time is prohibitive for large partitions and we therefore refrain from performing a comparison to that scheme.

### 17.1 Double Integrator

Consider the double integrator

$$x(k+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u(k)$$

The task is to regulate the system to the origin while fulfilling the input constraint  $\|u(k)\|_\infty \leq 1$  and state constraint  $\|x(k)\|_\infty \leq 5$ . For this system, we consider the optimisation problem (10.5) with a 2-norm objective for a prediction horizon  $N = 10$ . The objective weight matrices are set to  $Q = Q_F = I$  and  $R = I$ . For this example, there exists a lifting according to Remark 16.4 such that it is possible to construct the associated search tree. The construction process for Example 17.1 is depicted in Figure 17.1 and a rather fanciful version is given in Figure 17.2.

**Remark 17.1.** *For problems with a linear norm the lifting can be computed directly as the epigraph. However, this is not the case for quadratic problems as their epigraphs are quadratic. An appropriate lifting was computed for this example as a linear optimisation with the slopes of the affine regions as decision variables and continuity across boundaries as constraints.*

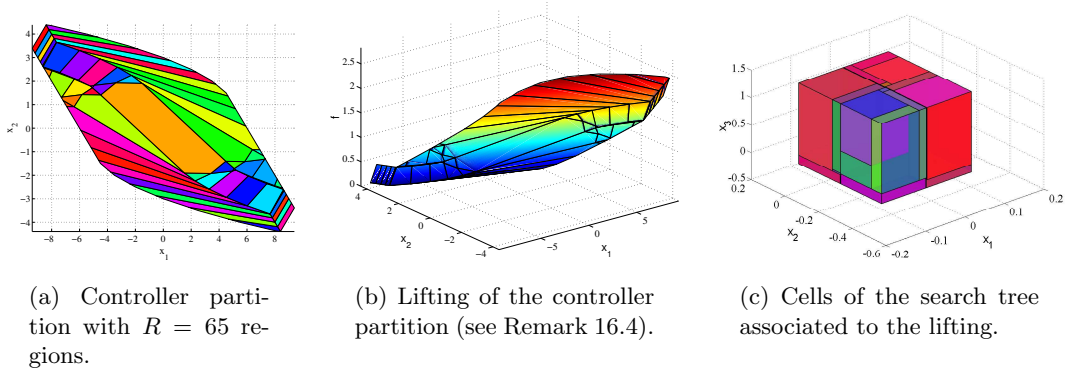


Figure 17.1: Search Tree Construction for Example 17.1

The search tree was constructed using the MPT toolbox [KGBM04] whose search tree construction differs slightly from the BBD-tree presented in Section 16.4

## 17.2 Large Random System

Consider the following 4-dimensional LTI system:

$$x(k+1) = \begin{bmatrix} 0.7 & -0.1 & 0 & 0 \\ 0.2 & -0.5 & 0.1 & 0 \\ 0 & 0.1 & 0.1 & 0 \\ 0.5 & 0 & 0.5 & 0.5 \end{bmatrix} x(k) + \begin{bmatrix} 0 & 0.1 \\ 0.1 & 1 \\ 0.1 & 0 \\ 0 & 0 \end{bmatrix} u(k)$$

subject to constraints  $\|u(k)\|_\infty \leq 5$  and  $\|x(k)\|_\infty \leq 5$ . This problem was solved for the  $\infty$ -norm, prediction horizon  $N = 5$  and for weighting matrices  $Q = I$  and  $R = I$ . The resulting controller partition consists of  $R = 12,290$  regions. The construction of the search tree required 0.03 seconds. In comparison, the approach in [TJB03] would require the solution to approximately 151,000,000 LPs, which is clearly prohibitive in terms of runtime. For  $\epsilon = 0.01$ , the average and worst-case number of floating point operations to compute the input using ANN [MA98] are 29,450 and 36,910 respectively. In comparison, the approach in [BBBM01] always takes exactly 160,000 operations.

## 17.3 Randomly Generated Regions

In this section we compare the computational complexity of the approach presented in this part with that discussed in [BBBM01] for very large systems. The currently available multiparametric solvers produce reliable results for partitions of up to approximately 30,000 regions. However, methods are currently being developed that will provide solutions for much

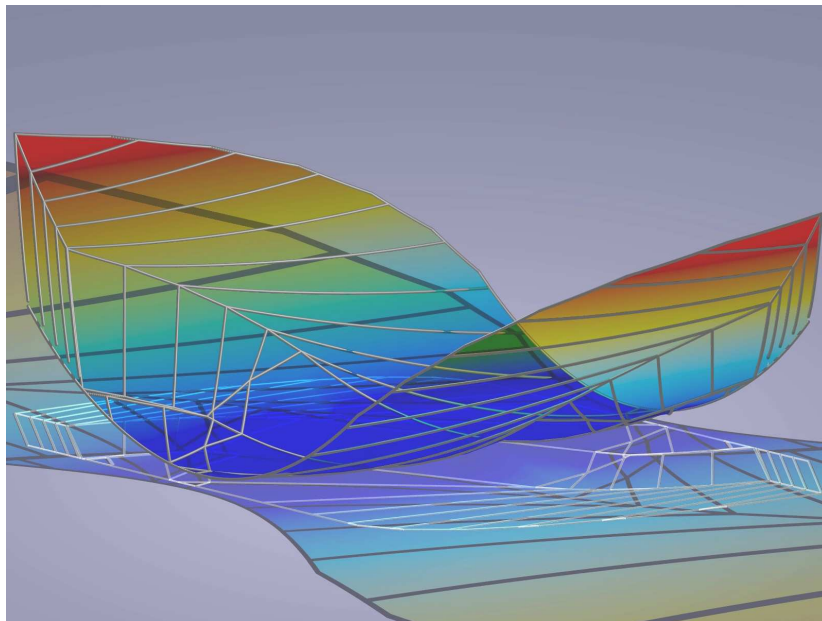


Figure 17.2: Controller Partition for the Double Integrator

larger problems. Therefore, in order to give a speed comparison we have randomly generated vectors  $G_r$  and  $g_r$  in the form of (16.2). The code developed in [AMN<sup>+</sup>98], which is available at [MA98], was then used to execute 1,000 random queries and the worst-case is plotted in Figure 17.3. For all of the queries the error parameter  $\epsilon$  was set to zero and therefore the solution returned is the exact solution. It should be noted that the preprocessing time for one million regions and 20 dimensions is merely 22.2 seconds.

Figure 17.3 shows the number of floating point operations (flops) as a function of the number of regions for the two approaches and the dimension of the state-space. Note that both axes are logarithmic.

A 3.0GHz Pentium 4 computer can execute approximately  $800 \times 10^6$  flops/second (estimated from timing large matrix multiplications in MATLAB). It follows that for a 10 dimensional system whose solution has one million regions, the control action can be computed at a rate of 20kHz using the proposed method, whereas that given in [BBBM01] could run at only 35Hz.

It is clear from Figure 17.3 that the calculation speed of the proposed method is very good for systems with a large number of regions. Furthermore note that controller partitions where ANN does worse than [BBBM01] are virtually impossible to generate, i.e. a partition in dimensions  $n = 10$  with less than  $R = 100$  regions is very difficult to contrive. Hence, it can be expected that for all systems of interest, the proposed scheme will result in a significant increase in speed. Since explicit feedback MPC is generally being applied to systems with

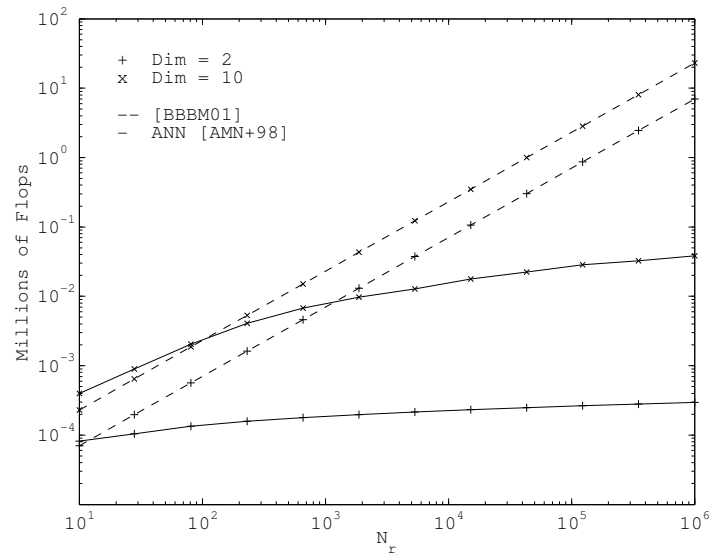


Figure 17.3: Comparison of ANN [AMN<sup>+</sup>98] (Solid lines) to [BBBM01] (Dashed lines)

very fast dynamics, any speedup in the set-membership test is useful in practice, i.e. the scheme proposed here is expected to significantly increase sampling rates.

## Part V

# Conclusions and Future Research



# Conclusions

In conclusion, the main contributions of this thesis and directions for future research are outlined.

## 18.1 Main Contributions

The main focus of this thesis has been the use of linear polyhedral computations in control. The specific contributions in this area are detailed as follows:

### **Part I: Equality Set Projection (ESP)**

This part presented the ESP algorithm, which is a new approach for the projection of polytopes in halfspace form. The proposed algorithm requires a linear number of linear programs per output facet in the absence of degeneracy and, as shown by simulation, degrades gracefully in its presence. If the size (dimension and number of inequalities) of the polytope is kept constant, it has been shown that the complexity becomes linear in the number of facets of the projection. ESP is an important contribution to the field of computational geometry, as it is currently the only output sensitive algorithm for projection. The loss of output sensitivity for degenerate problems is a common problem across most geometric algorithms, and so work continues towards resolving this problem.

The algorithm has been implemented in MATLAB and comparative simulations were given in Section 6.1, which greatly favoured ESP. The simulations presented were targeted at a class of polytopes for which ESP is particularly suited, namely those of interest to control. We do not claim, however, that ESP is the best algorithm in all cases. For example, there is a large class of polytopes for which the projection cone has a very small number of extreme rays or the number of vertices of the polytope is small, such as in a simplex. However, the

simulations demonstrate that there is a large class of polytopes for which ESP is uniquely suited and is able to calculate projections of polytopes in significantly higher dimensions than existing methods. Of particular note are high dimensional polytopes that are to be projected to a low dimension, polytopes represented by a small number of inequalities and hypercubes.

### **Part II: Multi-Parametric Linear Programming**

Three significant contributions were made in this part. First, the structure of the multi-parametric linear program was explored and it was proven that if the mpLP is non-degenerate, then the solution forms a complex, which is a vital property for the proof of completeness of the algorithms presented in this part. Second, a modification of the problem was introduced, which maintains this property in the presence of degeneracy. Furthermore, the modification also ensures that the primal optimiser is continuous, which is a very desirable property for the computation of control laws. Finally, four algorithms were presented to enumerate the critical regions of the solution complex, each with different strengths and weaknesses. The algorithms were compared in simulation for problems of interest to control, where they performed favourably.

### **Part III: Projection and multi-Parametric Linear Programs**

In this part an important link was made between multi-parametric linear programs and projection. Namely, an mpLP algorithm can be used to compute a projection and a projection algorithm can be used to compute mpLPs.

### **Part IV: Point Location Problem**

The promise of closed-form MPC computation is that the online calculation can then be done much faster than standard MPC, in which an optimisation problem must be solved at each time step. This allows the application of MPC to small, fast systems. However, if the number of regions is large, then the online calculation is not necessarily faster than solving the online optimisation.

This part introduced a new method of solving the point location problem for linear-cost MPC problems. If the controller partition exhibits a specific structure, the proposed scheme can also be applied to quadratic-cost MPC problems. It has been shown that the method is linear in the dimension of the state-space and logarithmic in the number of regions. Numerical examples have demonstrated that this approach is superior to the current state of the art and that for realistic examples, several orders of magnitude improvement in sampling rates are possible.



### 18.2 Future Research

Possible directions for future research are outlined as follows.

#### Approximation

As was seen in the simulations presented in this thesis, it is often the case that the number of critical regions or the number of facets in a projection is very large, and grows quickly with the problem size. From [AZ96], it is known that the worst-case complexity of a projection is in fact exponential. As was shown in Part III, mpLPs are equivalent to projection and therefore the worst-case number of critical regions is also exponential. This result threatens to limit the application of closed-form MPC (for linear costs) to very small problems, certainly less than ten dimensions, and probably less than five.

Recent work [Gri04] has shown how this complexity can be reduced by modifying the statement of the problem. A second method to reduce this is to approximate the projection directly. The ideal outcome would be an algorithm that would return a polytope of fixed complexity with a minimum fitting error to the projection.

#### Parametric Quadratic Programming

While this thesis has focused on linear problems, of which there are many applications both in and out of control, the norm most used in control is the 2–norm, which results in a multi-parametric quadratic program (mpQP). There is still much work to be done to fully understand the mpQP problem. First, there is as yet no proof that the set of critical regions forms a complex [SKJ<sup>+</sup>04]. This lack means that the current state-of-the-art algorithm [Bao02] must assume this result and therefore there is no proof that it will find the complete solution even if the cost is positive definite. Second, it would be useful to handle degeneracy in a similar manner as in Part II by applying an appropriate perturbation. A similar idea has been suggested in [STJ05b], where a perturbation is made to the cost when there is degeneracy. However, it should be possible to apply lexicographic perturbation methods by posing the mpQP as a parametric linear complementarity problem (LCP) [MY88]. In [TJB01], it was shown that for a non-degenerate problem, the active constraints in an adjacent region can be derived directly from those that are active on the boundary facet. Formulating the problem as an LCP may allow for a result similar to that in Section 9.3, which would extend this result to the degenerate case.

### Exploitation of Structure

The problems of most interest in control generally have a very similar structure. Namely, upper and lower bounds are placed on all states and inputs and these variables are linked through linear equations representing the dynamics. This results in a polytope that is a hypercube intersected with a cut-plane. It was found in early work that the projection of a rotated hypercube (zonotope) could be computed extremely quickly using ESP, as the inequalities defining ridges could be identified without any LPs and adjacent equality sets could be computed in a single pivot. While this result did not extend to axis-aligned hypercubes intersected with a subspace, the potential is there to exploit the known structure to an advantage.

### Redundancy Elimination

Despite all complexity results that show that the primary cost depends on the adjacency oracle for both ESP and mpLP calculations, every example presented in this thesis has demonstrated that for problems that are small enough to be calculable, redundancy elimination takes the majority of the time. The primal-dual algorithm of Section 9.4.3 was an effort to combat this problem, but its application is limited to small problems. Redundancy elimination is a fundamental problem in computational geometry and any improvement in this area would have a significant impact on many fields.

# Bibliography

- [AF92] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8:295–313, 1992.
- [AF96] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Math*, 65:21–46, 1996.
- [AGG03] H. Alt, M. Glisse, and X. Goaoc. On the worst-case complexity of the silhouette of a polytope. In *Proceedings of the 15th Canadian Conference on Computational Geometry (CCCG'03)*, pages 51–55, 2003.
- [AMN<sup>+</sup>98] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.
- [Arm93] P. Armand. Bounds on the number of vertices of perturbed polyhedra. *Annals of Operations Research*, (47):249–269, 1993.
- [Aur87] F. Aurenhammer. A criterion for the affine equivalence of cell complexes in  $\mathbb{R}^d$  and convex polyhedra in  $\mathbb{R}^{d+1}$ . *Discrete and Computational Geometry*, 2:49–64, 1987.
- [Aur91] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3), September 1991.
- [Avi00] D. Avis. lrs: A revised implementation of the reverse search vertex enumeration algorithm. In G. Kalai and G. Ziegler, editors, *Polytopes - Combinatorics and Computation*, DMV Seminar Band 29, pages 177–198. Birkhauser-Verlag, 2000.

- [AZ96] N. Amenta and G.M. Ziegler. Shadows and slices of polytopes. In *Proceedings of the twelfth annual symposium on computational geometry*, pages 10–19. ACM Press, 1996.
- [Bal61] M.L. Balinski. On the graph structure of convex polyhedra in n-space. *Pacific Journal of Math*, 95(11):431–434, 1961.
- [Bal98] E. Balas. Projection with a minimum system of inequalities. *Computational Optimization and Applications*, 10:189–193, 1998.
- [Bao02] M. Baotić. An efficient algorithm for multi-parametric quadratic programming. Technical report, ETH Zürich, Institut für Automatik, Physikstrasse 3, CH-8092, Switzerland, 2002.
- [BBBM01] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Efficient on-line computation of constrained optimal control. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 1187–1192, Orlando, Florida, December 2001.
- [BBM00] A. Bemporad, F. Borrelli, and M. Morari. Explicit solution of constrained  $l_\infty$ -norm model predictive control. In *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000.
- [BBM03] F. Borrelli, A. Bemporad, and M. Morari. Geometric algorithm for multiparametric linear programming. *Journal of Optimization Theory and Applications*, 118(3):515–540, September 2003.
- [BDH96] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.
- [Bem03] A. Bemporad. *Hybrid Toolbox - User's Guide*, December 2003. <http://www.dii.unisi.it/hybrid/toolbox/>.
- [BFM98a] D. Bremner, K. Fukuda, and A. Marzetta. Primal-dual methods for vertex and facet enumeration. *Discrete and Computational Geometry*, 20:333–357, 1998.
- [BFM98b] D. Bremner, K. Fukuda, and A. Marzetta. Primal-dual methods for vertex and facet enumeration. *Discrete and Computational Geometry*, 20:333–357, 1998.
- [Bla99] F. Blanchini. Set invariance in control - a survey. *Automatica*, 35(11):1747–1768, November 1999.

## BIBLIOGRAPHY

---

- [BMDP02] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002.
- [BO98] E. Balas and M. Oosten. On the dimension of projected polyhedra. *Discrete Applied Mathematics*, 87:1–9, 1998.
- [Bor02] F. Borrelli. *Discrete Time Constrained Optimal Control*. PhD thesis, Swiss Federal Institute of Technology (ETH), October 2002.
- [Bor03] F. Borrelli. *Constrained Optimal Control Of Linear And Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer, 2003.
- [BP83] E. Balas and W.R. Pulleybank. The perfectly matchable subgraph polytope of a bipartite graph. *Networks*, (13):495–516, 1983.
- [Bre99] D. Bremner. Incremental convex hull algorithms are not output sensitive. *Discrete and Computational Geometry*, 21(1):57–68, January 1999.
- [BT97] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.
- [BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [Cam85] S.A. Cameron. A study of the clash detection problem in robotics. In *IEEE Proc. Int. Conf. Robotics Automation*, volume 1, page 488, Saint Louis, March 1985.
- [Cha52] A. Charnes. Optimality and degeneracy in linear programming. *Econometrica*, (20):160–170, 1952.
- [CL97] T. Christof and A. Loebel. porta. Version 1.3.1, March 1997.
- [Cla] K.L. Clarkson. hull 1.0. AT&T Bell Labs.
- [Cla94] K.L. Clarkson. More output-sensitive geometric algorithms. In *35th Annual IEEE Symposium on the Foundations of Computer Science*, pages 695–702, 1994.
- [CLL00] V. Chandru, C. Lassez, and J-L. Lassez. Qualitative theorem proving in linear constraints. Working Paper 00-04, Systems Engg., UPenn, March 2000.
- [Dan48] G.B. Dantzig. Programming in a linear structure. In *Comptroller*, Washington D.C., 1948. USAF.

- [Dan51] G.B. Dantzig. *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*, chapter XXI. Wiley, New York, 1951.
- [DOW55] G.B. Dantzig, A. Orden, and P. Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, (5):183–195, 1955.
- [EM90] H. Edelsbrunner and E.P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.
- [FLL00] K. Fukuda, T.M. Liebling, and C. Lütolf. Extended convex hull. In *12<sup>th</sup> Canadian Conference on Computational Geometry*, pages 57–64, July 2000.
- [FLN97] K. Fukuda, H.J. Lüthi, and M. Namkiki. The existence of a short sequence of admissible pivots to an optimal basis in lp and lcp. *International Transactions of Operational Research*, 4(4):273–384, 1997.
- [FP96] K. Fukuda and A. Prodon. Double description method revisited. In M. Deza, R. Euler, and I. Manoussakis, editors, *Combinatorics and Computer Science*, volume 1120 of *Lecture Notes in Computer Science*, pages 91–111. Springer-Verlag, 1996. Postscript file available from <ftp://ftp.ifor.math.ethz.ch/pub/fukuda/reports/ddrev960315.ps.gz>.
- [Fuk99] K. Fukuda. Vertex enumeration for polyhedra algorithms and open problems. Seminar given at the University of Illinois in the course Advanced Topics in Analysis of Algorithms, 1999.
- [Fuk00] K. Fukuda. Frequently asked questions in polyhedral computation. <http://www.ifor.math.ethz.ch/fukuda/polyfaq/polyfaq.html>, October 2000.
- [Gal95] T. Gal. *Postoptimal Analyses, Parametric Programming and Related Topics*. Walter de Gruyter, 2<sup>nd</sup> edition, 1995.
- [GBTM04] P. Grieder, F. Borrelli, F. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40:701–708, 2004.
- [Gri04] P. Grieder. *Efficient Computation of Feedback Controllers for Constrained Systems*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, 2004.

## BIBLIOGRAPHY

---

- [Grü00] Branko Grünbaum. *Convex Polytopes*. Springer-Verlag, second edition, 2000.
- [JMSY93] J. Jaffar, M.J. Maher, P.J. Stuckey, and R.H.C. Yap. Projecting  $\text{clp}(\mathcal{R})$  constraints. *New Generation Computing*, 11(3,4):449–469, 1993.
- [Ker00] E.C. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, University of Cambridge, 2000.
- [KGBM04] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi Parametric Toolbox (MPT). In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 448–462, Philadelphia, Pennsylvania, USA, March 2004. Springer Verlag. <http://control.ee.ethz.ch/~mpt>.
- [MA98] D.M. Mount and S. Arya. Ann: Library for approximate nearest neighbour searching, June 1998. <http://www.cs.umd.edu/~mount/ANN/>.
- [MS] W. Murray and M. Saunders. Systems Optimization Laboratory (SOL). <http://www.sbsi-sol-optimize.com>.
- [Mur83] K.G. Murty. *Linear Programming*. John Wiley & Sons, 1983.
- [MY88] K.G. Murty and F.T. Yu. *Linear Complementarity, Linear and Nonlinear Programming*. Helderman-Verlag, 1988. [http://ioe.engin.umich.edu/people/fac/books/murty/linear\\_complementarity\\_webbook/](http://ioe.engin.umich.edu/people/fac/books/murty/linear_complementarity_webbook/).
- [OSS95] T. Ottmann, S. Schuierer, and S. Soundaralakshmi. Enumerating extreme points in higher dimensions. In E.W. Mayer and C. Puech, editors, *STACS 95: 12th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 562–570. Springer-Verlag, 1995.
- [Pad99] M. Padberg. *Linear Optimization and Extensions*. Algorithms and Combinatorics. Springer Verlag, 1999.
- [Pfa02] B. Pfaff. *An Introduction to Binary Search Trees and Balanced Trees*. Free Software Foundation, Inc, <http://benpfaff.org/>, 2002.
- [PSS<sup>+</sup>95] J. Ponce, S. Sullivan, A. Sudsang, J. Boissonnat, and J. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *International Journal of Robotics Research*, February 1995.
- [RGJ04] S. Raković, P. Grieder, and C.N. Jones. Computation of Voronoi Diagrams and Delaunay Triangulation via Parametric Linear Programming. Technical Report

- AUT04-03, Automatic Control Lab, ETHZ, Switzerland, 2004. <http://control.ethz.ch/>.
- [RKKM05] S. Raković, E.C. Kerrigan, K. Kouramas, and D. Mayne. Invariant approximations of the minimal robust positively invariant set. *IEEE Transactions on Automatic Control*, 2005.
- [Roc70] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.
- [Ryb99] K. Rybnikov. Stresses and liftings of cell complexes. *Discrete and Computational Geometry*, 21(4):481 – 517, June 1999.
- [SH95] V. Saraswat and P.V. Hentenryck, editors. *Principles and Practice of Constraint Programming*, chapter 14, pages 245–268. The MIT Press, 1995.
- [SKJ<sup>+</sup>04] J. Spjøtvold, E.C. Kerrigan, C.N. Jones, T.A. Johansen, and P. Tøndel. Conjectures on an algorithm for convex parametric quadratic programs. Technical Report CUED/F-INFENG/TR.496, Department of Engineering, University of Cambridge, Cambridge, UK, October 2004.
- [SLG<sup>+</sup>04] R. Suard, J. Löfberg, P. Grieder, M. Kvasnica, and M. Morari. Efficient computation of controller partitions in multi-parametric programming. In *IEEE Conference on Decision and Control*, Bahamas, 2004.
- [STJ05a] J. Spjøtvold, P. Tøndel, and T.A. Johansen. A method for obtaining continuous solutions to multiparametric linear programs. In *accepted IFAC*, 2005.
- [STJ05b] J. Spjøtvold, P. Tøndel, and T.A. Johansen. Unique polyhedral representations to continuous selections for convex multiparametric quadratic programs. In *accepted ACC*, 2005.
- [Sys] SysBrain Ltd., Southampton, UK. *User's Manual. Reference of the Geometric Bounding Toolbox*, March. Version 7.3.
- [TJB01] P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, Florida, USA, 2001.
- [TJB03] P. Tøndel, T.A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39:743–749, 2003.



## BIBLIOGRAPHY

---

- [TU03] C.E. Testuri and S. Uryasev. *Handbook of Numerical Methods in Finance*. Birkhauser, 2003.
- [TZ91] T. Terlaky and S. Zhang. A survey on pivot rules for linear programming. Technical Report 91-99, Delft University of Technology, 1991. ISSN 0922-5641.
- [Č63] S.N. Černikov. Contraction of finite systems of linear inequalities (in russian). *Doklady Akademiia Nauk SSSR*, 152(5):1075–1078, 1963. (English translation in *Societ Mathematics Doklady*, Vol. 4, No. 5 (1963), pp.1520-1524).
- [VLS00] R. Vidal, S. Schaffert, J. Lygeros, and S. Sastry. *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, chapter Controlled Invariance of Discrete Time Systems, pages 437–450. Springer Verlag, 2000.
- [Web94] R. Webster. *Convexity*. Oxford University Press, 1994.
- [Zie95] G.M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, New York, 1995.



# Author Index

- Alt, H. 2  
Amenta, N. 11, 195  
Armand, P. 88  
Arya, S. 5, 6, 178, 179, 184–186, 188–190  
Aurenhammer, F. 182, 184  
Avis, D. 10, 11, 60, 67, 125–127  
  
Balas, E. 10, 19, 23  
Balinski, M.L. 29, 81  
Baotić, M. 177, 178, 182, 187–190  
Barber, C.B. 11, 60, 67, 125, 139, 141, 144  
Bemporad, A. 74, 75, 140, 143, 146, 147,  
177, 178, 182, 187–190, 195  
Bertsimas, D. 79, 81, 82  
Blanchini, F. 9  
Boissonnat, J. 1, 10, 11  
Borrelli, F. 54, 74, 75, 140, 143, 146, 177,  
178, 182, 187–190  
Boyd, S. 154, 157  
Bremner, D. 11, 60, 67, 120, 125  
  
Cameron, S.A. 2  
Chandru, V. 2  
Charnes, A. 86  
Christof, T. 11, 60, 67  
Clarkson, K.L. 11, 60, 111  
  
Dantzig, G.B. 83, 86  
Dobkin, D.P. 11, 60, 67, 125, 139, 141, 144  
  
Dua, V. 177  
  
Edelsbrunner, H. 11  
  
Fukuda, K. 10, 11, 16, 33, 59, 60, 67, 93,  
111, 120, 125–127  
  
Gal, T. 73, 76  
Glisse, M. 2  
Goaoc., X. 2  
Grieder, P. 6, 75, 111, 140, 141, 143, 144,  
178, 179, 188, 195  
Grünbaum, Branko 100–102  
  
Huhdanpaa, H. 11, 60, 67, 125, 139, 141, 144  
  
Jaffar, J. 10  
Johansen, T.A. 73, 77, 100, 177, 178, 187,  
188, 195  
Jones, C.N. 73, 100, 178, 195  
  
Kerrigan, E.C. 2, 9, 66, 73, 100, 195  
Kouramas, K. 66  
Kvasnica, M. 6, 75, 111, 140, 143, 179, 188  
  
Lassez, C. 2  
Lassez, J-L. 2  
Liebling, T.M. 33  
Loebel, A. 11, 60, 67  
Löfberg, J. 111

- Lüthi, H.J. 93  
Lütolf, C. 33  
Lygeros, J. 9
- Maher, M.J. 10  
Marzetta, A. 11, 60, 67, 120  
Mayne, D. 66  
Merlet, J. 1, 10, 11  
Morari, M. 6, 74, 75, 111, 140, 143, 146,  
177–179, 182, 187–190  
Mount, D.M. 5, 6, 178, 179, 184–186,  
188–190  
Mücke, E.P. 11  
Murray, W. 60, 138  
Murty, K.G. 39, 79, 83, 86, 88, 90, 97, 195
- Namkiki, M. 93  
Netanyahu, N.S. 5, 178, 184–186, 189, 190
- Oosten, M. 19, 23  
Orden, A. 86  
Ottmann, T. 111
- Padberg, M. 161  
Pfaff, B. 114  
Pistikopoulos, E.N. 177  
Ponce, J. 1, 10, 11  
Prodon, A. 10, 11, 59, 60, 67  
Pulleybank, W.R. 10
- Raković, S. 66, 178  
Rockafellar, R.T. 14, 15, 28  
Rybnikov, K. 184
- Sastry, S. 9  
Saunders, M. 60, 138  
Schaffert, S. 9  
Schuierer, S. 111
- Silverman, R. 5, 178, 184–186, 189, 190  
Soundaralakshmi, S. 111  
Spjøtvold, J. 73, 77, 100, 195  
Stuckey, P.J. 10  
Suard, R. 111  
Sudsang, A. 1, 10, 11  
Sullivan, S. 1, 10, 11
- Terlaky, T. 83  
Testuri, C.E. 2  
Tøndel, P. 73, 77, 100, 177, 178, 187, 188,  
195  
Torrìsi, F. 75  
Tsitsiklis, J.N. 79, 81, 82  
Uryasev, S. 2
- Vandenbergh, L. 154, 157  
Černikov, S.N. 10, 159  
Vidal, R. 9
- Webster, R. 23  
Wolfe, P. 86  
Wu, A.Y. 5, 178, 184–186, 189, 190
- Yap, R.H.C. 10  
Yu, F.T. 195
- Zhang, S. 83  
Ziegler, G.M. 11, 17, 18, 22, 23, 29, 91, 121,  
159, 161, 169, 195

# Index

- additively weighted nearest neighbour, 182
- affine hull, 14
  - calculation of, 54
- affine set, 14
  - parallel, 14
  - translate, 14
- affinely independent, 15
- approximate nearest neighbour, 185
- basis, 80
- boundary complex, 101
- complex, 100
- convex, 15
- convex combination, 15
- convex hull, 15
- critical region, 95
- cycling, 84
- degree of degeneracy, 87
- diamond property, 18
- duality, 97
- edge, 17
- epigraph, 100
- equality set, 19
- ESP
  - adjacency oracle, 29, 31–39
  - degeneracy, 51
  - degenerate, 51
  - Equality Set Projection, 27–50
  - ray-shooting oracle, 29
  - ridge oracle, 29, 39–46
  - shooting oracle, 46–48
- face, 16
  - adjacent, 18
  - face lattice, 17
- facet, 17
- facet traversal method (FTM), 75
- feasible direction, 81
- fundamental duality theorem, 97
- hyperplane, 14
- irredundant, 16
- lex-feasible, 87
- lex-pivot, 90
- lexico-positive, 87
- lexicographic perturbation, 86
- mpLP
  - critical region, 101
  - enumeration
    - basic, 114
    - facet-based, 117
    - primal-dual, 120
    - reverse search, 125

- neighbourhood problem, 109
  - adjacency oracle, 113
  - facet oracle, 111
  - solution complex, 100, 101
- multi-parametric linear program, 73
  
- point location problem, 181
- pointed cone, 160
- polyhedron, 15
- power diagram, 182
- projection, 22
- projection lemma, 159
  
- redundancy elimination, 111
- redundant, 16
- region compliment method (RCM), 75
- ridge, 17
  
- simplex method, 83
  - degeneracy, 83
    - dual, 90
    - primal, 84
  - leaving variable, 81
  - ratio test, 81
  - unique optimal basis, 92
- subspace, 13
  - dimension, 13
  
- vector space, 13
- vertex, 17
  - adjacent, 80
- voronoi diagram, 182