A Robust Linear Program Solver for Projectahedra

by

Marius Laza

M.Sc., Polytechnic Institute of Bucharest, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

December 2001

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of COMPUTER SCIENCE

The University of British Columbia
Vancouver, Canada

Date DEC 19, 2001

Abstract

Linear programming has a wide range of applications, optimization-related problems being one of them. Important concerns in linear programming are efficiency, robustness, and accuracy. Linear programming is used in a reachability analysis tool called Coho [GM99] for dynamical systems. Previous experience has shown that linear programs in this tool lead to highly ill-conditioned linear systems which prevented successful reachability analysis. This thesis presents a robust linear program solver with provable error bounds that exploits the special structure of the linear programs that result in the reachability tool. This contribution is of interest for the particular application for which it was developed. Furthermore, it shows how duality and combinatorial aspects of linear programming can be exploited to achieve greater efficiency, robustness, and accuracy.

Contents

List of Figures

Acknowledgements

This thesis would not have been possible without substantial help from several individuals.

It is difficult to overstate my gratitude to my supervisor, Mark Greenstreet, for his extensive support and encouragement.

I wish to thank David Kirkpatrick, James Varah, and Eldad Haber for their contribution of time and ideas to my thesis.

I am grateful to Alan Hu for the encouragement and advice that he provided to me when I needed them the most.

I would like to thank my wife, Mira, for constantly supporting me and for believing in my eventual success even when I despaired.

MARIUS LAZA

The University of British Columbia
December 2001

To my parents, who wanted it more than anybody else.

Chapter 1

Introduction

1.1 Motivation

The problem of verification is that of showing that a design satisfies its specification. The design may be of an electronic circuit, a computer program, a network or security protocol, a chemical plant, an airplane, etc. For our purposes, the specification describes the desired behaviors of the design: that the circuit implements a particular finite state machine, that the security protocol does not disclose passwords, that the chemical plant does not explode, etc. The goal of formal verification is to produce a formal, mathematical proof that the design has the desired properties. For this approach, both the design and the specification must be modeled in a mathematical framework where such a proof is mathematically meaningful.

This thesis is concerned with verification where the design is modeled by a system of non-linear, ordinary differential equations (non-linear ODEs) and a description of the possible initial states of the system. The specification describes a "safe" region in which the trajectories for all solutions to the model must be contained. The verification task is to determine whether from a possible initial state the system can ever reach a forbidden state or not. This type of verification is termed reachability analysis.

For non-trivial systems, reachability analysis tools must use approximation

techniques: closed-form solutions do not exist. Coho, the verification system described in this thesis, is one such tool. It computes over-approximations of the reachable space to provide a sound verification of safety properties: Coho may fail to verify a correct system, but it will not erroneously verify an incorrect system. As part of computing the evolution in time of the reachable state space of a system, Coho solves a large number of linear programs. The soundness of Coho relies on computing accurate error bounds for the solutions of these linear programs. To avoid false negatives, it is desirable that these solutions be as accurate as practical.

Linear programming is a well-studied problem that has a classical solution, namely the Simplex algorithm. On any real machine, errors are an inherent part of floating-point computations. In this thesis, we address the impact of numerical errors from floating-point computations on the accuracy and robustness of the Simplex algorithm, as applied to our verification system.

The effect of errors in the input and in the intermediate computations on the result of a problem is measured by its conditioning. The result of an ill-conditioned problem may be affected by large errors even though the errors in the input or in the intermediate computations are small.

Many of the linear programs that arise in Coho are ill-conditioned problems. For such problems Simplex tends to yield solutions that contain large errors, compromising the applicability of Coho. Moreover, no error bounds are available on the solutions, preventing the tool from producing a guarantee of correctness for the system being analyzed.

A noteworthy property of Coho linear systems is that their structure is special. More precisely, the feasible region of a Coho linear program is the intersection of set of orthogonal back-projections into the full-dimensional space of 2D polygons. This means that any inequality in the definition of the feasible region contains only two variables. This thesis explores ways of exploiting this special structure in order to obtain greater accuracy and robustness while keeping efficiency reasonable.

1.2 Contribution

The special structure of the linear programs arising from orthogonal projections allows the implementation of an efficient and numerically robust version of the Simplex algorithm.

The main contributions of this research are the following:

1. An implementation of Simplex where the combinatorial representation of bases remains explicit.
    - The key to practicality is an $O(n)$ linear system solver, where n is the number of variables in the system.
2. Numerical robustness achieved by computing accurate error bounds.
    - The combinatorial approach above allows us to avoid numeric error propagation between steps, thus keeping the error bounds reasonably tight.
    - When the optimal basis is ill-conditioned, it is shown that a pivot can be made to another basis that has nearly the same cost and is well-conditioned with respect to the cost function.
3. An error bound for the optimal cost that is independent of the numerical value appearing in the constraints.
4. Implementation

1.3 Outline

This thesis presents an efficient and numerically robust method of solving the linear programs that arise in the Coho verification tool.

- Chapter 2 presents the context in which linear programs with a particular structure arise. First, the reachability analysis class of problems and its applicability to verification are introduced. Then a system that implements reachability analysis is described with emphasis on its use of linear programs. Finally, the special structure of these linear programs and their impact on the usability of the system are underlined.

- Chapter 3 reviews the Simplex algorithm for solving linear programs. Both the geometric and the combinatorial aspects of the problem are presented. The concepts of basis, pivoting, duality, and cycling receive particular attention. The standard implementation of the Simplex algorithm is presented briefly, pointing out its problems in the case at hand.

- Chapter 4 presents the main features of the proposed modified version of Simplex. These include the lazy computation of tableau columns and the linear-time algorithm that accomplishes the computation. The presentation of the method of computing error bounds on the results of the floating-point operations proposed to be used by the linear program solver concludes the chapter.

- Chapter 5 reviews the numerical accuracy and stability of the new algorithm for solving the particular type of linear systems that arise in Coho. An error bound on the solution to such linear system as computed by the new algorithm is established.

  The rest of the chapter focuses on ways of obtaining a good approximation of the optimal cost of a linear program. The geometrical meaning of an optimal ill-conditioned basis is analyzed and a method of approximating its cost is proposed and then analyzed. Then the application of this method to the Coho linear programs is studied.

- Chapter 6 presents the implementation details that were found to be significant

during algorithm implementation.

A chapter of conclusions and suggestions for further research completes the core of the thesis, followed by an appendix containing the definitions of the mathematical notations and definitions used in the thesis.

Chapter 2

Background

2.1 Projectahedra

A projectahedron is a high-dimensional polyhedron represented by its projections onto two-dimensional subspaces, where these projections are not required to be convex. The high-dimensional object is the largest set of points that satisfies each projection. A full-dimensional polyhedron can be obtained from its projections by back-projecting each into a prism in $\mathbf{R}^d$ and computing the intersection of those prisms. Each (1-dimensional) edge of a projection polygon corresponds to a $(d-1$-dimensional) face of the projectahedron.

The intersection computation for arbitrary polyhedra in high dimensions is computationally hard. Projectahedra are a restricted class of high-dimensional polyhedra and the complexity of computing the intersection of projectahedra does not appear to have been studied. In the work described in this thesis, the intersections of projectahedra are never explicitly nor exactly computed. Instead, operations on projectahedra are performed projectionwise. Sometimes this leads to an overapproximation of the result projectahedron. We choose these operations in such a way as to preserve the soundness of Coho. Whereas the projectionwise computation of common projectahedra operations like union typically leads to overapproximation, the same method yields the exact result in the case of intersection.

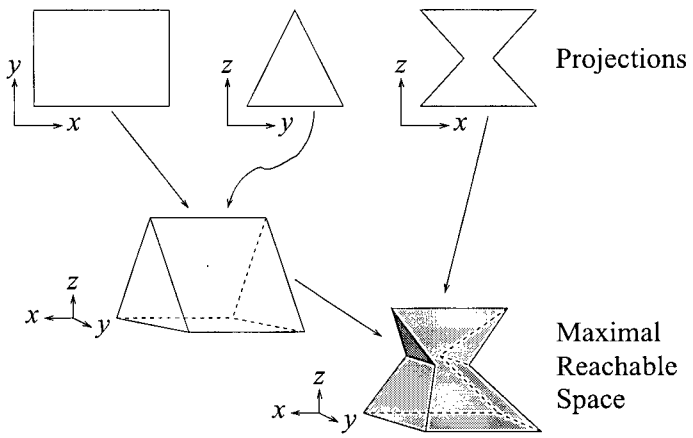

Figure 2.1: A three dimensional "projectahedron"

2.2 Verification as Reachability

Consider a system whose dimension (i.e. number of variables) is d. The continuous state space of the system is $\mathbf{R}^d$. Suppose the behavior of the system is described by the differential inclusion:

$$\dot{x} \in F(x)$$

where $x \in \mathbf{R}^d$. The inclusion models uncertainty in the model, environment etc.

Given two regions, $A_0 \subseteq B \subseteq \mathbf{R}^d$, the reachability problem is to determine whether all trajectories that start in A_0 at $t = 0$ remain in B, either during some time interval, $[0, t_{end}]$, or for all time.

For example, we can find $A_t \subseteq \mathbf{R}^d$ such that $x(t) \in A_t$. The reachability problem is satisfied if $A_t \subseteq B$, $\forall t \in [0, t_{end}]$.

A related problem is the following: given a time t_1 and a region A_1, show that at $t = t_1$, all trajectories are inside A_1. This can be reduced to the first problem by including time in the state with $\dot{t} = 1$ [AL94].

Many verification problems can be formulated as reachability analysis problems. Consider for example a system consisting of two aircraft [TPS98]. Given the possible initial positions of the aircraft and their equations of motion, the question is whether the distance between the two aircraft remains above a lower bound for all times of interest. Modeling each aircraft as a point in $\mathbf{R}^3$, the state of the system is a point in $\mathbf{R}^6$. The safety requirement partitions $\mathbf{R}^6$ into safe and unsafe regions. Solving the verification problem boils down to determining whether A_t intersects the unsafe part of $\mathbf{R}^6$ for any time t of interest.

An important problem in modern circuit design is determining whether a circuit correctly implements its high-level specification. Reachability analysis can be used for verifying that circuits, as modeled by non-linear ODE's, correctly implement discrete specifications.

2.3 Coho

Coho is a verification system that performs reachability analysis [GM99].

Closed form solutions to reachability problems exist only for a few special cases. Consequently, approximation techniques are used to analyze real systems. These techniques ensure that the approximations always lead to an over-approximation of the reachable space. Every point that actually is reachable is included in the approximation computed by Coho. The approximation may also include points that cannot be reached by the real system. Thus, the verification performed is sound - an incorrect design will never pass verification, but a correct one might fail it because of the approximations.

A general representation of high-dimensional objects is intractable. For this reason, Coho uses projectahedra to approximate high-dimensional objects, such as the initial region and the reachable regions at various times.

The Coho reachability computation is an iterative, computation algorithm. A single time-step of this algorithm proceeds as follows [GM98]:

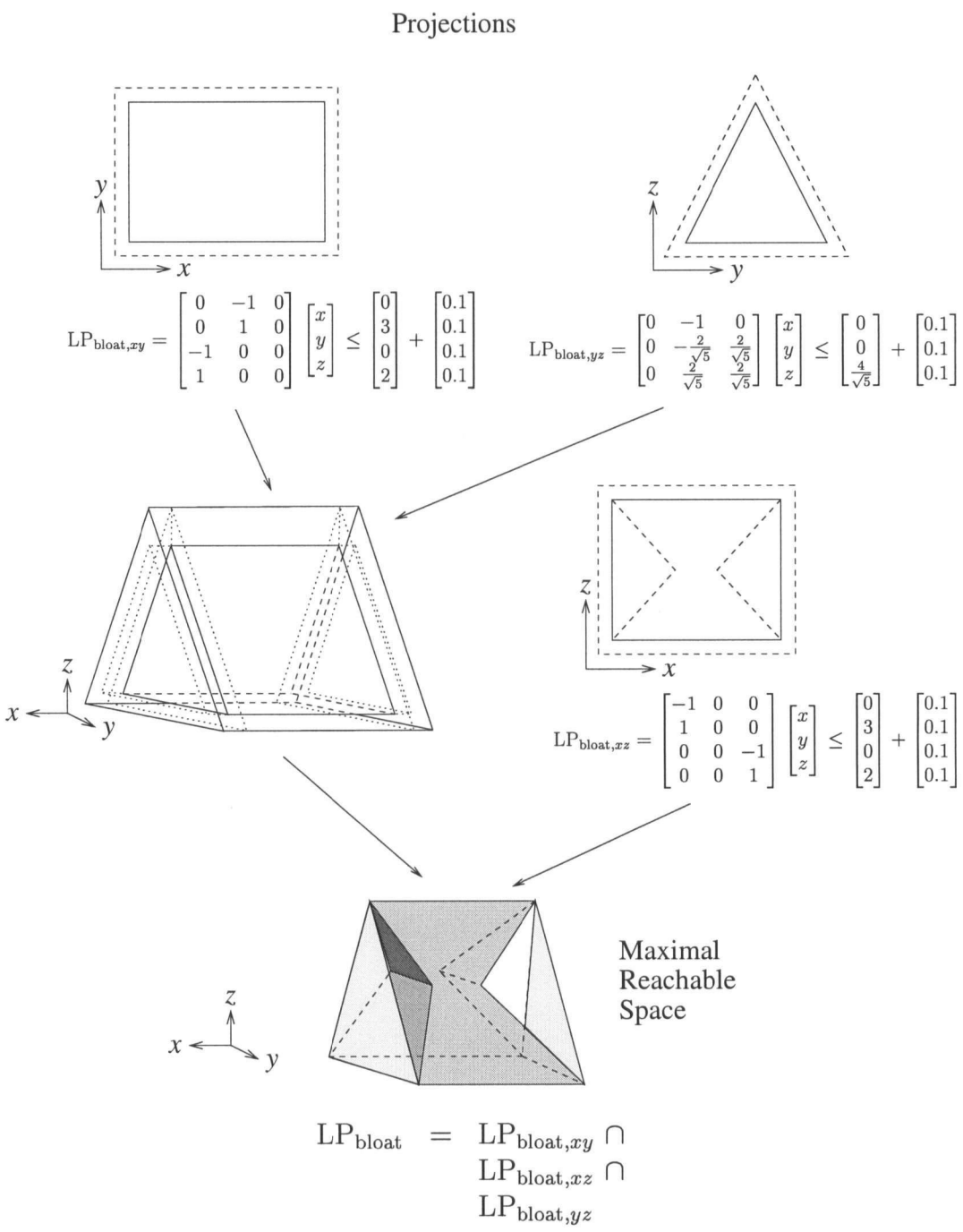

Figure 2.2: The creation of a bloated linear program for the convex hull of the projectahedron

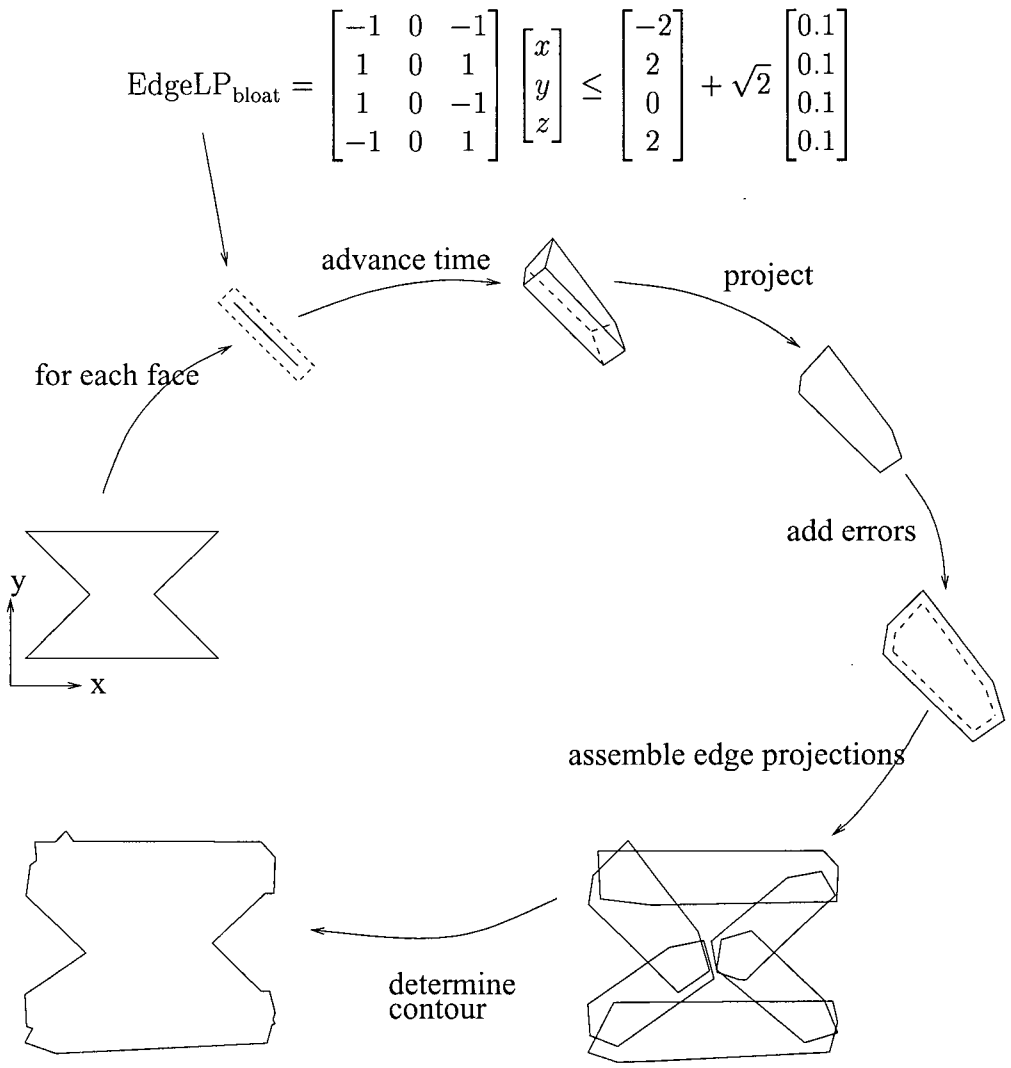

Figure 2.3: A time step of Coho

1. The time step begins by loading a polygon and its convex hull for each projection of the system. The convex hulls are then bloated outward slightly to ensure that they contain all possible trajectories for the next time step. Each projection's bloated convex hull can be translated into a set of linear inequalities in the projection's two coordinates. The conjunction of all the projections' linear inequalities describes a convex region containing the projectahedron. At this point, the movement of each edge of each projection's polygon can be computed independently. Each edge corresponds to a face of the projectahedron, and the objective is to compute the furthest outward that points on the face could move during a time step. For each face, the following computations occur:

2. (a) *Restriction:* The convex region computed from the convex hulls is further restricted to a box around the edge in the coordinates of the edge as described by four more linear inequalities. In the full dimensional space this is equivalent to constructing a slab around the face being examined. The slab is a conservative estimate of the convex hull of the bloated face.

   (b) *Linearize Model:* The slab's description is available in terms of the collection of linear inequalities computed in the previous step. The derivative function for the model is assumed to be autonomous (i.e. independent of time) and finitely piecewise continuous (therefore locally bounded). A linearization of the system derivatives that is valid in the slab is computed. This model includes linear and constant terms, and gives bounds on the error introduced by the linearization within the slab. Typically, this linearization is based on bounds for the variables in the model and bounds on linear combinations of these variables. These bounds are computed by solving the corresponding linear programming problem. More formally, let W be the slab represented as a set of inequalities. The

non-linear model is approximated with the differential inclusion:

$$x \in W \Rightarrow \dot{x} \in Ax + b + U \tag{2.1}$$

where $A \in \mathbf{R}^{d \times d}$ is a matrix, $b \in \mathbf{R}^d$ is a vector and $U \in (\mathbf{R} \times \mathbf{R})^d$ is a hypercube (i.e. a Cartesian product of intervals).

(c) *Advance Time:* The linear model is used to move the slab forward in time according to the first two terms of equation (2.1). The U term is handled as an inhomogeneous stimulus to the system as described in step 2e. The forward time transformation is performed by exponentiating the A matrix. This new convex region contains any point reachable from the convex hull of the face at the end of the time step (ignoring U). Moving the slab model forward in time amounts to right multiplying the left-hand side of its inequalities by a matrix that transforms points at the end of the time step back to their location at the beginning of the step and also modifying their right-hand side. The application of matrix multiplication to the left-hand side would lead to the modification of its structure. For reasons of computational efficiency, the implementation transforms the cost function instead of the inequalities. This is described in detail in chapter 6.

(d) *Project Back:* The slab's end-of-step shape is described by a collection of linear inequalities after time is advanced. Building a polygon from these inequalities requires projecting the region that they contain back onto the basis for the projection polygon corresponding to the face. This projection is computed by running a series of linear programs on the time-advanced set of inequalities. The cost functions used by the linear programs are directions contained in the plane of the polygon.

(e) *Add Errors:* So far, the slab's movement is entirely controlled by the linearized model. To treat the error, we add a constant derivative offset

within the error bounds throughout the time step, in such a way as to bloat the slab's projection outward as much as possible. This involves the computation of bounds on $\|\dot{x}\|_1$ over the slab. An over-approximation of $\|\dot{x}\|_1$ is computed based on the extremes of each $\dot{x}_i$ over the slab. Computing these extremes again involves solving linear programs.

3. Each edge of each projection's polygon therefore produces an "edge polygon" at the end of the time step; this polygon contains the projection of all points that could be reached from the corresponding face within the time step. The outer boundary of the union of all such polygons is the projection of an over-approximation of the projectahedron at the end of the time step.

Clearly the use of linear programming by Coho is heavy. In fact, a major limitation to the applicability of this tool stems from the failure of the classical Simplex algorithm to compute sufficiently accurate solutions to the linear programs that arise in Coho.

Each vertex of the feasible region of a linear program lies at the intersection of d hyperplanes, where d is the dimension of the space. If the normal to at least one of these hyperplanes is almost a linear combination of the normals to the other hyperplanes, the vertex is ill-conditioned: the use of a typical implementation of Simplex or other LP algorithm for the determination of the vertex position leads to a result likely to be affected by large errors.

However, the feasible region of any Coho linear program is special. As any convex polyhedron, the feasible region can be described by the matrix inequality:

$$Ax \geq b$$

where $A \in \mathbf{R}^{d \times d}$ and $b \in \mathbf{R}^d$. Each row of this inequality represents a halfspace that corresponds to at most one face of the polyhedron. As the feasible region represents the intersection of back-projections into $\mathbf{R}^d$ of two-dimensional convex polygons, each row of matrix A contains at most two non-zero elements.

The above observations lead to the idea of exploiting the special structure of the feasible region of Coho linear programs in order to solve them more accurately, thus enhancing the usability of the system.

Chapter 3

Linear Programs

Linear programs play an important role in the Coho system. This thesis presents a method for computing better solutions to the particular category of linear programs that arise in Coho.

This chapter introduces the mathematical description of linear programs and of an algorithm to solve them. Then, section 3.3 describes the particular linear programs that arise in Coho.

3.1 Problem Definition

Definition 1 *Let m, n be positive integers, $A \in \mathbf{R}^{m \times n}$ an $m \times n$ matrix of reals, $b \in \mathbf{R}^m$ an m-vector of reals, $c \in \mathbf{R}^n$ an n-vector of reals, $M \subseteq \{1, \ldots, m\}$ a set of indices of rows of matrix A, $N \subseteq \{1, \ldots, n\}$ a set of indices of columns of matrix A. Let $\overline{M} = \{1, \ldots, m\} \setminus M$ and $\overline{N} = \{1, \ldots, n\} \setminus N$. Let $s \in \{+1, -1\}$.*

Then the following problem is an instance of a general linear program:

$$\begin{aligned} &\min_{x \in \mathbf{R}^n} sc^T x \\ &\textit{subject to:} \\ A_{i,:}x &= b_i \quad i \in M \\ A_{i,:}x &\geq b_i \quad i \in \overline{M} \\ x_j &\geq 0 \quad j \in N \\ x_j &\textit{ unconstrained} \quad j \in \overline{N} \end{aligned} \tag{3.1}$$

Such an instance of the general linear program is denoted by $\mathrm{LP}(A, b, c, M, N, s)$.

Column matrix c *is called the* cost vector *or the* optimization direction *of the linear program.*

The value x_{opt} *for which the minimum is attained, if it exists, is called the* optimal solution *of the linear program.*

The value $sc^T x_{\mathrm{opt}}$ *is called the* optimal cost *of the linear program.*

The sign s specifies whether the problem is one of minimization $(+1)$ or one of maximization(-1).

The following trivial transformations enable the reduction of other forms of linear programs to the one above:

- A maximization problem can be turned into one of minimization by negating the cost vector:

$$\max c^T x = -\min -c^T x$$

- An inequality of the form:

$$a^T x \leq b, \quad a, x \in \mathbf{R}^n,\ b \in \mathbf{R}$$

is equivalent to:

$$-a^T x \geq -b$$

A point $x \in \mathbf{R}^n$ that satisfies all the constraints of the linear program is called a *feasible* point of the linear program. The set of all feasible points represents the *feasible region* of the linear program, denoted by feas(LP). A linear program is called feasible if its feasible region is non-empty. Otherwise it is called infeasible.

As an intersection of convex sets (hyperplanes and closed halfspaces), the feasible region of an LP is a convex set. An optimal solution lies on the boundary of the feasible region.

In general, the optimal solution might not be unique. Consider the trivial case with $c = 0$: all feasible points are optimal.

Let x_{opt} be an optimal solution to a linear program. If the affine subspace that is normal to the cost vector and contains x_{opt} contains other feasible points, they too are optimal.

Recall that a general linear program was defined as a problem of minimization. This means that the optimal point is the feasible point that lies the farthest in the negative direction of the cost vector (see fig. 3.1).

If a linear program consists of a minimization problem and its feasible region is unbounded in the negative direction of the cost vector, then the cost function can take arbitrarily large negative values and the linear program is said to be unbounded. If the feasible region of a linear program is non-empty and bounded in the negative direction of the cost vector, then the linear program has a finite optimum.

An important particular case of a general program is when $M = \{1, \ldots, m\}$ and $N = \{1, \ldots, n\}$, i.e. when all constraints are equalities and all variables must be positive:

$$\begin{aligned} &\min c^T x \\ &Ax = b \\ &x \geq 0 \end{aligned} \tag{3.2}$$

Such a linear program is said to be in standard form and is denoted by $\text{SLP}(A, b, c)$:

$$\text{SLP}(A, b, c) = \text{LP}(A, b, c, \{1, \ldots, m\}, \{1, \ldots, n\}, +1)$$

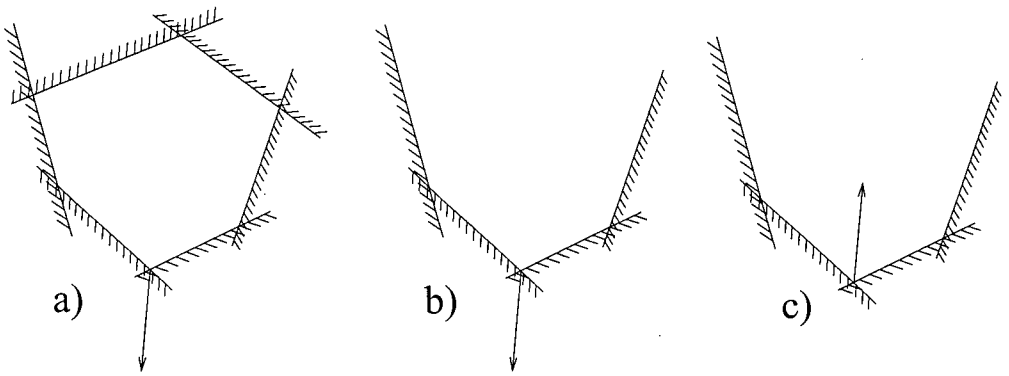

Figure 3.1: Types of maximization linear programs: a) bounded, non-empty feasible region; b) unbounded feasible region bounded in the optimization direction; c) unbounded feasible region unbounded in the optimization direction; The arrow indicates the optimization direction, which for maximization problems coincides with that of the cost vector. The shading indicates the outer (infeasible) side of each line.

Throughout the rest of the discussion about linear programs it is assumed that $m < n$ and $rank(A) = m$. If $rank(A) < m$, then $m - rank(A)$ rows of $[A|b]$ can be deleted without changing the problem.

A linear program in standard form is amenable to solution using the Simplex algorithm [PS82, p.26]. A linear program in general form can be reduced to standard form by using the following straightforward transformations:

- A variable x_j that is unrestricted as to sign can be replaced with the difference of two non-negative variables:

$$x_j = x_j^+ - x_j^-, \quad x_j^+ \geq 0, \; x_j^- \geq 0$$

- An inequality constraint $\sum_{i=1}^{n} A_{ij}x_j \geq b_i$ can be converted into the equation:

$$\sum_{i=1}^{n} A_{ij}x_j + s_i = b_i, \quad s_i \geq 0$$

  The variable s_i is called a *surplus* variable. The similar transformation for a "less-than" constraint introduces a *slack* variable.

All linear programs that arise in Coho are of the form:

$$\begin{aligned} &\max c^T x \\ &Ax \geq b \\ &x \text{ unconstrained} \end{aligned} \tag{3.3}$$

Throughout this thesis this form of linear program is termed Coho form. A linear program in this form is called a Coho linear program, denoted by $\mathrm{CLP}(A, b, c)$. The following equation relates a Coho linear program to a general linear program:

$$\mathrm{CLP}(A, b, c) = \mathrm{LP}(A, b, c, \emptyset, \emptyset, -1)$$

A Coho LP can obviously be reduced to an LP in the standard form. Consider a Coho LP that has f inequalities and d variables. Each inequality and each variable in the original system requires the addition of an extra variable in the equivalent LP in standard form. The equivalent system would have $d + f + d$ variables and f equations. Moreover, the special structure of the original LP would be destroyed by the transformation.

However, the Coho LP can be solved without reducing it to the standard form by using a general characteristic of linear programs called duality.

For an LP in general form, called the primal, the following construction defines another LP, called its *dual*:

Primal		Dual
$\min sc^T x$		$\max sb^T y$
$A_{i,:}x = b_i$	$i \in M$	y_i unconstrained
$A_{i,:}x \geq b_i$	$i \in \overline{M}$	$y_i \geq 0$
$x_j \geq 0$	$j \in N$	${A_{:,j}}^T y \leq c_j$
x_j unconstrained	$j \in \overline{N}$	${A_{:,j}}^T y = c_j$

(3.4)

The dual can also be rewritten in the following way:

$$
\begin{array}{rl}
 & -s \min -sb^T y \\
(-A^T)_{j,:}\, y \geq -c_j & j \in N \\
(-A^T)_{j,:}\, y = -c_j & j \in \overline{N} \\
y_i \geq 0 & i \in M \\
y_i \text{ unconstrained} & i \in \overline{M}
\end{array}
\tag{3.5}
$$

This is to say that:

$$\text{dual}(\text{LP}(A, b, c, M, N, s)) = \text{LP}(-A^T, -c, b, \overline{N}, \overline{M}, -s)$$

The attributes "primal" and "dual" are interchangeable: the dual of the dual is the primal.

Any primal-dual pair of linear programs has the following remarkable property:

- If the dual has a finite optimum, then so does the primal and their optimal costs are equal.

  The optimal point of the dual can be easily computed from the optimal point of the primal and vice versa.

- If the dual is infeasible, then the primal is either infeasible or unbounded.

- If the dual is unbounded, then the primal is infeasible.

This property means that the solving of an LP can be replaced with the computation of the solution to its dual, with almost no loss of information. The only case where a precise verdict cannot be given for the primal is when the dual is infeasible. However, in many cases, knowing that a linear program doesn't have a finite optimum suffices. In fact, in the systems we are examining, the linear programs that arise cannot be unbounded.

The dual of a Coho linear program is easily seen to be a linear program in standard form:

$$\begin{aligned}
\text{dual}(\text{CLP}(A,b,c)) &= \text{dual}(\text{LP}(A,b,c,\emptyset,\emptyset,-1)) \\
&= \text{LP}(-A^T,-c,b,\{1,\ldots,f\}\setminus\emptyset,\{1,\ldots,d\}\setminus\emptyset,+1) \\
&= \text{LP}(-A^T,-c,b,\{1,\ldots,f\},\{1,\ldots,d\},+1) \qquad (3.6) \\
&= \text{SLP}(-A^T,-c,b) \\
&= \text{SLP}(A^T,c,b)
\end{aligned}$$

Therefore the solution to $\text{CLP}(A,b,c)$ can be obtained by solving $\text{SLP}(A^T,c,b)$. It is clear that no variables are added and the structure of matrix A remains intact.

3.2 Linear Programs in Standard Form

3.2.1 Feasible Region

Consider an instance of a linear program in standard form $\text{SLP}(A,b,c)$ with f variables and d equations. It will be seen later that this SLP corresponds to a polyhedron with f faces in the d-dimensional space, hence the new choice of letters for the dimensions of the linear program. The feasible region of SLP is the portion of a d-dimensional affine subspace of $\mathbf{R}^f$ that lies inside the non-negative orthant. If the feasible region is non-empty, then an optimal point exists at the intersection of this subspace with one of the positive semiaxes of $\mathbf{R}^d$.

3.2.2 Bases

A set of d linearly independent columns of matrix A is called a *basis*. A basis is described either as a set of column indices, more precisely called the *basic set* corresponding to the basis:

$$\mathcal{B} = \{j_1,\ldots,j_d\}$$

or through the restriction of the linear program's matrix A to the basic set of columns:

$$B = A_{:,\mathcal{B}}$$

The terms "basic set" and "basis" are used interchangeably when there is no chance of confusion.

The columns that belong to a given basis are called *basic columns*, whereas the others are called *non-basic*. Each column of matrix A of an LP in standard form corresponds to a variable. The attribute "basic" extends to variables in the natural way. The values of the basic variables are:

$$t_0 = B^{-1}b$$

The *basic solution* corresponding to a basis $\mathcal{B}$ is a vector $x \in \mathbf{R}^f$ obtained by expanding the vector of basic variables in the natural way:

$$x_j = \begin{cases} 0 & \text{if } j \notin \mathcal{B} \\ t_{0,k} & \text{if } j = \mathcal{B}_k \end{cases}$$

If an LP in standard form has an optimal solution, it also has a basic optimal solution.

A basic solution that has no negative components represents a feasible point for the LP and is called *feasible*. Otherwise it is called *infeasible*. The attribute "feasible" extends to bases in the natural way.

The situation in which a basic variable is equal to 0 is called *degeneracy*. The corresponding basis and the basic solution are said to be *degenerate*. More than one basis can correspond to the same degenerate solution, all such bases being degenerate.

3.2.3 Pivoting

A well-known algorithm for solving linear programs is called Simplex. Simplex operates on linear programs in standard form.

In addition to the description of the linear program to be solved, Simplex must be supplied with a feasible basis for that program. Finding a feasible basis is non-trivial, but it will be dealt with later.

Simplex is a greedy algorithm. During each step, it tries to replace one of current basic columns with a new column in order to obtain a new feasible basis of lower cost.

The search ends at the optimal basis, which is the cheapest feasible basis.

If a non-degenerate feasible basis is not optimal, then there exists at least one non-basic column whose introduction into the basis results in a decrease in the cost. Let t_j be the column vector defined by:

$$t_j = B^{-1}A_j$$

The quantity

$$\bar{c}_j = c_j - c_B^T t_j$$

is called the *relative cost* of column j with respect to basis $\mathcal{B}$. The introduction of column j in basis $\mathcal{B}$ might be favorable (reduce the cost) if the relative cost of the column is negative.

Once a column with negative relative cost is found, the algorithm must determine which column to evict from the basis. The decision is guided by the requirement that the new basis must be feasible and is accomplished by the following computation:

$$k = \arg\min_i \frac{t_{0,i}}{t_{j,i}}$$

where k is the index of the column to be evicted. In the presence of degeneracy k may not be uniquely defined.

The action of moving from one feasible basis to another is called *pivoting.*

The cost decrease achieved by pivoting as described by the computations above is

$$\frac{t_{0,k}}{t_{j,k}}\bar{c}_j$$

The following observations are in order regarding pivoting:

- If $t_{j,i} < 0, \forall i = 1, \ldots, d$, the feasible region of the linear program is unbounded in the negative direction of the cost vector. Feasible points of arbitrary low cost exist.
- In the presence of degeneracy, the quantity $\frac{t_{0,k}}{t_{j,k}}$ can be 0 and so can the decrease in cost:

$$\exists i, j \text{ s.t. } t_{0,i} = 0 \wedge \bar{c}_j < 0 \wedge t_{j,i} > 0 \Rightarrow \frac{t_{0,k}}{t_{j,k}} = 0 \Rightarrow \frac{t_{0,k}}{t_{j,k}} \bar{c}_j = 0$$

  This is equivalent to a change of basis without a change in the basic solution. It is said that the new column *enters the basis at zero level.*

In the absence of degeneracy, Simplex works because:

- There exists a finite number of feasible bases.
- At every step the cost decreases monotonically, ensuring that a basis is never visited again.
- The optimal solution is among the basic feasible solutions.

Simplex is by no means guaranteed to produce the shortest path from an initial feasible basis to the optimal basis, but it typically performs well in practice.

3.2.4 Cycling

In the presence of degeneracy, Simplex is no longer guaranteed to work with any choice of a favorable column. It is possible that, once arrived at a degenerate basis, the algorithm takes a sequence of favorable pivots to subsequent degenerate bases. These pivots do not decrease the cost and the algorithm can eventually return to the first degenerate basis. Obviously, the algorithm can loop indefinitely through a set of degenerate bases that all yield the same solution unless special precautions are taken. This phenomenon is called cycling.

Cycling occurs when the cost function is a positive combination of less that d columns of a basis.

Cycling avoidance is achieved by Bland's anticycling algorithm [Bla77], [PS82, p.50] that chooses the lexically first pivot at first step:

- The columns that enters the basis is the lowest numbered one.
- In case of a tie in the computation of the column that leaves the basis, the lowest numbered column is selected.

3.2.5 The Simplex Tableau

At each step, the Simplex algorithm makes pivoting decisions based on the values in the matrix

$$T = B^{-1}[A|b]$$

The matrix T is called the Simplex tableau. For simplicity, its last column is indexed by 0. Computing the Simplex tableau from the input data every time pivoting occurs would render Simplex prohibitively expensive: solving a $d \times d$ linear system with $f - d$ right-hand sides takes $O(d^2(f-d))$ in the general case.

In practice, this expensive solution is replaced with the computation of each new tableau from the previous one at the lower cost of adding one row to each of the others ($O(d(f-d))$). The tableau corresponding to the initial basis still needs to be computed from the initial data.

Let $\mathcal{B}'$ be the basis obtained by replacing column $j = \mathcal{B}(l)$ with column j' in basis $\mathcal{B}$ and T and T' the tableaus corresponding to bases $\mathcal{B}$ and $\mathcal{B}'$, respectively. Then T' can be obtained from T through the following operations:

$$\begin{aligned} T'_{l,:} &= T_{l,:}/T_{l,j'} \\ T'_{i,:} &= T_{i,:} - T_{i,j'}T'_{l,:} \end{aligned} \tag{3.7}$$

The use of a tableau presents the disadvantage that numerical errors accumulate as the algorithm proceeds.

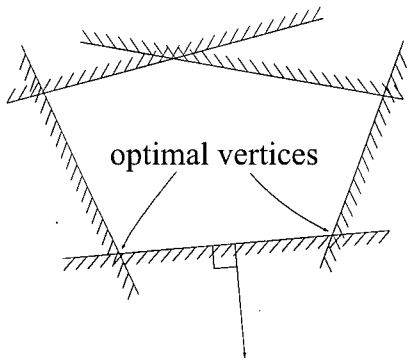

Figure 3.2: Linear program with two optimal vertices

3.3 Linear Programs in Coho Form

The Coho form of a linear program can offer more insight, particularly as regards the geometric meaning of linear programs.

Consider a primal linear program in Coho form with d variables and f inequalities $\mathrm{CLP}(A^{\mathcal{C}}, b^{\mathcal{C}}, c^{\mathcal{C}})$ and its dual in standard form $\mathrm{SLP}(A^{\mathcal{S}}, b^{\mathcal{S}}, c^{\mathcal{S}})$, where $A^{\mathcal{S}} = (A^{\mathcal{C}})^T$, $b^{\mathcal{S}} = c^{\mathcal{C}}$, $c^{\mathcal{S}} = b^{\mathcal{C}}$.

The feasible region of a Coho LP is a closed convex polyhedron:

$$\mathrm{feas}(\mathrm{CLP}(A^{\mathcal{C}}, b^{\mathcal{C}}, c^{\mathcal{C}})) = \mathrm{PH}(A^{\mathcal{C}}, b^{\mathcal{C}})$$

The optimal point of a Coho LP is a vertex of the feasible region. As illustrated by fig. 3.2, the optimal vertex might not be unique. For example, all the points in a hyperplane normal to the cost vector have the same cost. If the vector of the cost function is normal to a face of the polyhedron and oriented towards its interior, all the vertices on that face are optimal.

Each constraint in the primal defines a halfspace whose boundary is a hyperplane. Each such hyperplane contains a face of the feasible region, unless it is redundant. Each row of the primal $A^{\mathcal{C}}$ (i.e. each column of the dual $A^{\mathcal{S}}$) represents a normal to a face of the feasible region oriented towards the interior of the feasible region. Such a normal is called an inward face normal.

A basis in the standard-form dual represents a set of d halfspaces in the Coho primal. The intersection of their boundaries determines a point in $\mathbf{R}^d$, which is the primal solution associated with that basis. The *basic primal (Coho) solution* corresponding to a basic set $\mathcal{B}$ is:

$$x^{\mathcal{C}} = (A^{\mathcal{C}}{}_{\mathcal{B},:})^{-1}\, b^{\mathcal{C}}{}_{\mathcal{B}}$$

Basic primal solutions will be termed *vertices* by abuse of terminology, as in general they do not represent vertices of the feasible region of the Coho LP. Those of them that are actual vertices of the feasible region will be termed *proper vertices.*

The intersection of the basic halfspaces of the Coho LP is a cone whose vertex is the basic primal solution. This cone represents the feasible region with respect to the constraints comprised in the basis and is called the *basic feasible cone.*

A feasible basis of the standard dual represents a set of halfspaces of the Coho primal such that the primal cost vector is a positive combination of their inward face normals. In other words, the primal cost vector must lie inside the cone generated by the basic inward face normals. This cone is called the *basic cost cone.* It is natural to consider both the basic cost cone and the primal cost vector originated at the basic primal solution.

The bases that Simplex visits along the way to an optimal solution (other than the optimal one) represent feasible suboptimal, solutions in the standard form. In the Coho form, they represent infeasible supraoptimal solutions (see fig. 3.3 b). Monotonicity is preserved, however: the cost decreases monotonically in the standard form, whereas the infeasibility (expressed as the distance to the closest feasible vertex along the cost vector) decreases monotonically in the Coho form.

The optimal vertex of an LP in Coho form is an intersection point of d halfspace boundaries that satisfies both of the following properties (see fig. 3.3 a):

- The cost vector is a positive combination of the inward face normals. All non-optimal vertices lead to at least one negative component.

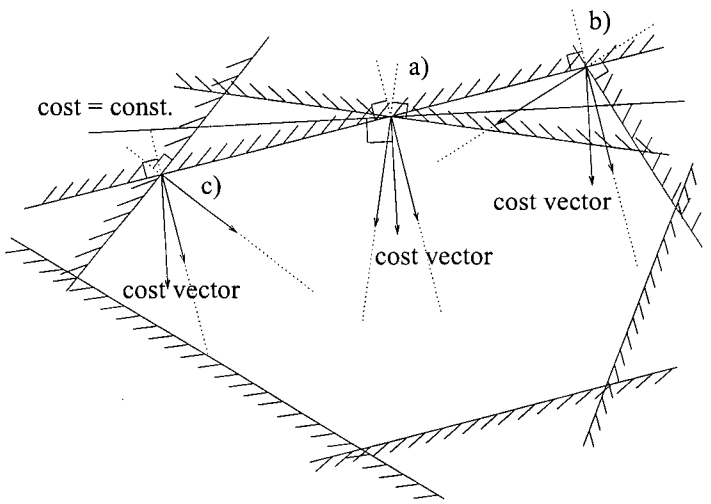

Figure 3.3: Types of vertices in a Coho LP: a) optimal; b) Coho-infeasible and standard-suboptimal, at least one constraint is violated; c) Coho-suboptimal and standard-infeasible, the cost vector does not lie within the cost cone

- It satisfies all the constraints, i.e. it belongs to the feasible region. Any non-vertex intersection point breaks at least one constraint.

Chapter 4

Combinatorial Simplex for Coho

An important obstacle in the way of the verification of systems with moderately high dimensionality (5-20 variables) by Coho is the need to solve linear programs with sufficient accuracy. Coho allows that the solution to any LP be an overapproximation of the feasible region in the direction of the cost function, like in figure 3.3 b. Underapproximation (figure 3.3 c), however, is not allowed - otherwise Coho might incorrectly label faulty systems as correct. The amount of overapproximation must be kept low, or Coho might fail to verify correct systems.

Coho has previously employed an implementation of the classical Simplex algorithm for its linear programming needs. Oftentimes, the optimal solution to a Coho LP represents the solution to a highly ill-conditioned linear system. In such cases the solutions computed by classical Simplex tend to contain substantial errors for which no bounds are provided, thus preventing Coho from functioning. In the previous implementation, these large errors often led to arithmetic exceptions preventing Coho from generating any results.

From a purely mathematical point of view, Simplex works by taking favorable pivots until it reaches a basis from which no favorable pivot can be taken. This basis is optimal and the solution that corresponds to it is the optimal solution of the linear program.

The mathematical view of Simplex implies that the arithmetic operations with real numbers are performed with infinite precision. Computers, however, use floating-point arithmetic, which uses only limited precision.

The favorability of a pivot is determined based on the values in the Simplex tableau. The errors that affect these values can lead to incorrect decisions about the favorability of a pivot. This in turn can result in incorrect determination of the optimal basis or in numerical cycling.

Even if the optimal basis is determined correctly, the optimal solution, which is itself a tableau column, is affected by errors for which no bound is available.

These problems are addressed as follows: The special structure of the bases that arise in Coho linear programs is exploited in order to make the computation of tableau columns directly from the input data feasible, thus reducing errors. Two methods for determining error bounds on tableau columns are presented, one relying on the use of running error analysis and the other analytical. The analytical method is presented in the next chapter, with the rest of the aforementioned material forming the topic of the current chapter.

Even with improved accuracy in the computation of the tableau columns and availability of error bounds, it is still possible that all the pivots from a basis are neither clearly favorable nor clearly unfavorable, which renders the basis neither clearly optimal nor clearly suboptimal. In such a case, branching of the computation path is used in order to guarantee the visitation of the optimal basis. This is presented in detail in chapter 6.

As discussed in chapter 3, it is advantageous to solve the dual of a Coho linear program instead of reducing the primal problem to standard form at the price of altering its structure. This better solution is assumed to be used throughout the rest of this chapter. At the same time, the fact that the matrices of the Coho linear program and of its standard form dual are identical up to a transposition enables us to refer to the original Coho LP when that is advantageous to understanding the

system.

The remainder is structured as follows: The first section presents the idea of computing tableau columns only when access to them is required by the program. Then the linear-time computation of such a column is examined. The description of a technique called running error analysis, used in order to obtain error bounds on the solution, concludes the chapter.

4.1 Lazy Tableau Generation

Simplex arrives at the optimal solution by taking a series of favorable pivots. Taking a pivot amounts to identifying a non-basic column that replaces a column in the basis, thus producing a lower-cost feasible basis. Depending on how the selection of the column that enters the basis is made, the need to know some of the tableau columns might not arise during a particular pivoting operation. In particular, it is enough to discover the lowest-numbered favorable column: if this column is chosen to enter the basis, then knowledge of the higher-numbered columns of the tableau is unnecessary in the current step of Simplex and their computation can be omitted. This policy, called lazy tableau generation, is the one followed in the version of Simplex tailored for Coho. The computation of a part of the tableau is thus avoided.

It must be emphasized that with incomplete tableau generation, the necessary tableau columns are computed from the input data rather than from the incomplete tableau of the previous pivot. The only data that is passed from one pivot to the next is the new basis. A basis is a collection of integers, so error propagation and accumulation across pivots is eliminated.

As mentioned in subsection 3.2.5, the computation of tableau columns from the input data is in general undesirable because of the high computational cost. For Coho linear programs however, a more efficient algorithm is available: a tableau column can be computed in linear rather than cubic time. This makes the algorithm very practical.

4.2 Efficient Computation of Tableau Columns

Each inequality in a Coho linear program CLP$(A^{\mathcal{C}}, b^{\mathcal{C}}, c^{\mathcal{C}})$ represents the halfplane corresponding to the backprojection of a side of a two dimensional polygon back into the full-dimensional space. As a result, each row of matrix $A^{\mathcal{C}}$ contains either one or two non-zero elements. The one non-zero case occurs when the polygon side is parallel to one of the coordinate axes that determine the plane that contains it. Hence matrix $A^{\mathcal{S}} = (A^{\mathcal{C}})^T$ of the standard-form dual of a Coho LP contains either one or two non-zeros in each column.

Let B be an arbitrary basis of the linear program in standard form. As B represents a subset of the columns of matrix $A^{\mathcal{S}}$, B is a square matrix that, like $A^{\mathcal{S}}$, has either one or two non-zero elements per column.

A tableau column is described by the equation:

$$T_{:,j} = \begin{cases} B^{-1}A^{\mathcal{S}}{}_{:,j}, & \text{if } j \neq 0 \\ B^{-1}b^{\mathcal{S}}, & \text{if } j = 0 \end{cases}$$

The task at hand is to solve a linear system whose left-hand side is B. Such a linear system is henceforth referred to as a Coho linear system. Its solution can be determined in two stages that are described in the subsections that follow.

4.2.1 Reduction to a Cycle

Whereas the columns of B are restricted to containing 1 or 2 non-zero elements, the rows of B are not under a similar constraint. However, if a row of matrix B contains no non-zero elements, B is trivially singular and a solution cannot be determined.

If a row i of matrix B contains exactly one non-zero element, which lies in column j, the value of the variable x_j can be determined immediately. If column j contains another non-zero element in a row i', variable x_j can be eliminated from row i' by the appropriate substitution. Row i and column j of matrix B can then

be deleted, as the value of x_j has been determined and no other equation depends on this value.

The rows of B momentarily containing one non-zero element each can thus be eliminated one by one. The key to keeping the running time of this computation linear is to check the number of non-zero elements left in row i' after deleting row i. If there are zero non-zero elements left in row i', matrix B is trivially singular and the algorithm terminates. If there is one non-zero left, the algorithm proceeds with the solving and deletion of row i'.

In the end, no rows of B with less than 2 elements are left. It is possible that matrix B has become empty, in which case a solution to the problem has already been found.

Now consider the case where B is non-empty. One row and one column have been deleted from matrix B during each step of the algorithm, so B must still be a square matrix. Let its dimension be n.

- By the termination condition of this part of the algorithm, each row contains at least 2 non-zero elements. Consequently, B must contain at least $2n$ non-zero elements.

- By hypothesis, each column of B contains at most 2 non-zero elements at the start of this part of the algorithm. As the algorithm proceeds, the number of non-zero elements in columns that remain in the matrix is unchanged. Consequently, B must contain at most $2n$ non-zero elements.

Clearly, both inequalities can be satisfied only if each column and each row of B contains exactly 2 non-zero elements.

Consider a graph where the vertices correspond to the rows and there is an edge from vertex i_1 to vertex i_2 if and only if there exists a column of matrix B whose non-zero elements are in rows i_1 and i_2. Because every row has exactly 2 non-zero elements in it, every vertex in the corresponding graph has degree 2. Therefore the graph is a collection of disjoint simple cycles.

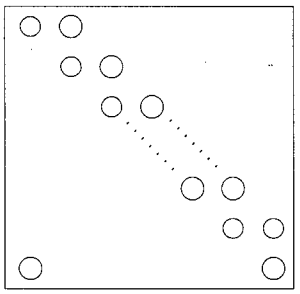

Figure 4.1: Non-zero structure of a cycle

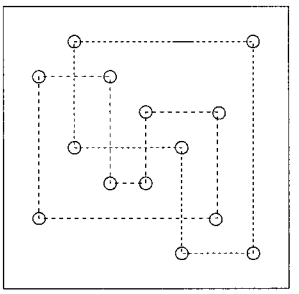

Figure 4.2: Cycles in a matrix

Each simple cycle corresponds to a linear system that can be solved independently from the others. By a suitable permutation, matrix B can be rearranged such that all its non-zero elements are grouped in square blocks along the main diagonal. Blocks are henceforth termed cycles, as each block represents a simple cycle in the graph derived from the matrix.

Each $n \times n$ block A can be permuted such that its non-zero elements are on the main diagonal, right above the main diagonal and in the lower left corner, as shown in Fig. 4.1:

$$A_{i,j} \neq 0 \Leftrightarrow j = i \vee j = (i \bmod n) + 1 \tag{4.1}$$

The partitioning of matrix B into cycles is achieved by a greedy walk through the graph corresponding to B. This is easily doable in linear time.

4.2.2 Solving a Cycle

In order to complete the solving of a Coho linear system, solutions to its cycles must be found.

Let A be a cycle with the structure described by (4.1) and let

$$Ax = y \tag{4.2}$$

be the corresponding linear subsystem to be solved.

The rows of matrix A can be scaled to obtain its *normalized form*:

$$A = \begin{bmatrix} 1 & -\alpha 1 & 0 & \ldots\ldots & 0 \\ 0 & 1 & -\alpha_2 & 0 \ldots & 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0 & \ldots\ldots\ldots\ldots & 0 & 1 & -\alpha_{n-1} \\ -\alpha_n & 0 & \ldots\ldots\ldots & 0 & 1 \end{bmatrix} \tag{4.3}$$

which is equivalent to:

$$A_{i,j} = \begin{cases} 1 & \text{if } j = i \\ -\alpha_i & \text{if } j = i \bmod n + 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Let

$$P_k = \begin{cases} \prod_{i=1}^{k} \alpha_i & \text{if } k = 1, \ldots, n \\ 1 & \text{if } k = 0 \end{cases} \tag{4.5}$$

It is obvious that:

$$P_k = \alpha_k P_{k-1}, \quad \forall k = 1, \ldots, n \tag{4.6}$$

The first row in equation (4.2) yields:

$$x_2 = \frac{x_1 - y_1}{\alpha_1} \tag{4.7}$$

More generally, rows $1, \ldots, i$ yield:

$$x_{i+1} = \frac{x_1 - \sum_{j=1}^{i} (y_j P_{j-1})}{P_i} \tag{4.8}$$

Finally, combining this with the last row of equation (4.2) gives the formula for x_1:

$$x_1 = \frac{\sum_{j=1}^{n} (y_j P_{j-1})}{1 - P_n} \tag{4.9}$$

Thus, the solution for x_1 is ill-conditioned if P_n is close to 1. Chapter 5 presents a more detailed error analysis. Clearly, equation (4.9) can be rewritten to obtain similar formulas for the other x_i's, all with the same denominator. Rather then computing each x_i separately, it is more efficient to compute x_1 as per formula (4.9), and then use the recurrence:

$$x_{i+1} = \frac{x_i - y_i}{\alpha_i}, \quad i \in \{1, \ldots, n-1\} \tag{4.10}$$

Clearly, the above algorithm runs in $O(n)$ time. Other algorithms that employ elimination, like LU decomposition, can be used in order to achieve the same running time. However, the one presented in this subsection has the advantage of yielding the result in a concise form that is appropriate for the analysis of the numerical stability of the system.

4.3 Running Error Analysis

An important problem with the Coho LPs is the need for an error bound on the computed solutions. In addition to the solution proper to the Coho LP, error bounds on the solution are necessary if the verification is to be sound. More precisely, it is important to never underapproximate the feasible region of the LP and the optimal cost.

Pivoting decisions in the Simplex algorithm involve comparisons between tableau elements. In the presence of ill-conditioning, these comparisons might yield uncertain results. The availability of error bounds on tableau elements and other quantities enables the algorithm to recognize cases of uncertainty in the result of a comparison.

The Simplex algorithm and the algorithm that computes tableau columns make use of elementary operations only. This leads to the approach of computing an error bound on the result of each elementary operation. The computation of an error bound along with the result proper of each operation leads to an error bound

on the final outcome. This approach to error analysis is called running error analysis [Hig96, p.72]. A slight modification of the method is called interval arithmetic.

In many cases, interval arithmetic doesn't work because of the explosion of the interval as the computation proceeds through a long algorithm. The slightly better running error analysis does work for Simplex because, with the computation of tableau columns from input data proposed in section 4.1, there is no floating-point data propagated from one pivot of the algorithm to the next. Although running error analysis tends to lead to overly pessimistic error bounds when applied to a long algorithm, it can in some cases provide sharper, *a posteriori* bounds than an *a priori* analysis can provide [Hig96, p.73].

The arithmetic operations on floating-point numbers are generally subject to rounding errors when executed on digital computers. This is caused essentially by the fact that floating-point numbers are stored with only a fixed number of digits, whereas the result of an operation might require more digits than the particular numeric format has available.

All the arithmetic operations executed on a computer follow the fundamental rule of the arithmetic of the computer:

$$\mathrm{fl}(x \,\mathrm{op}\, y) = (x \,\mathrm{op}\, y)(1+\delta), \quad |\delta| \le u, \quad \mathrm{op} \in \{+, -, \times, \div\}$$

where $\mathrm{fl}(x \,\mathrm{op}\, y)$ is the result of $x \,\mathrm{op}\, y$ computed by that arithmetic and u is a constant for a particular arithmetic called the unit roundoff. For the IEEE double precision arithmetic, $u = 2^{-53} \approx 1.1 \times 10^{-16}$.

When the operands are themselves affected by errors, they can be regarded as intervals on the real axes. Consider the pair (x, e) to be the representation of the interval $[x - e, x + e]$. Let $(x, e) = (x_1, e_1)\,\mathrm{op}(x_2, e_2)$. Then $x = \mathrm{fl}(x_1 \,\mathrm{op}\, x_2)$ and

$$e = \begin{cases} e_1 + e_2 + u|x| & \text{if op} \in \{+, -\} \\ e_1|x_2| + e_2|x_1| + u|x| + e_1 e_2 & \text{if op} \in \{\times\} \end{cases}$$

Division can be regarded as the inversion of denominator followed by the multipli-

cation of its result with the numerator. Let $(x, e) = (x_0, e_0)^{-1}$. Then $x = 1/x_0$ and

$$e = \begin{cases} \frac{e_0}{|x_0|(|x_0|-e_0)} & \text{if } |x_0| > e_0 \\ +\infty & \text{otherwise} \end{cases}$$

Clearly, the computation of the error bounds is itself affected by rounding errors. Fortunately, the error bound does not need to be known with high precision: its order of magnitude will often suffice. Moreover, disastrous cancellation cannot occur in the computation of error bounds: all the numbers involved are positive and subtraction does not occur.

There are some problems with the use of running error analysis. The first is that ideally one would like to have a simple formula to compute error bounds for the solutions of a linear system. The other issue is that the computation of error bounds on the result of each arithmetic operation along with the actual result increases the running time of the program. However, the penalty is a constant factor, not a deterioration of the asymptotic running time. As verification is executed as an off-line process, an increase of the running time by a constant factor can be seen as a reasonable price to pay if the algorithm would otherwise fail.

Chapter 5

Analytical Attack on the Error

As emphasized in previous chapters, the success of Coho verification depends strongly on the accuracy with which the linear programs that it produces are solved. The main floating-point (hence error-prone) computation that is performed as part of Simplex is the determination of the tableau columns corresponding to a basis.

However, for bases other than the optimal one, only enough accuracy is needed to be able to determine the departing and the entering basic variables. This is the case because no floating-point values computed at a basis are subsequently reused in our version of Simplex. For most of the bases encountered during a run of Simplex, the computation of tableau columns using the linear-time algorithm presented in section 4.2 along with running error analysis produces satisfactory results.

On the other hand, there can exist suboptimal bases at which the error bounds on the tableau columns are not tight enough to establish whether the basis is optimal and, if it is not, which pivot leads closer to optimality. Whereas such a situation can be dealt with by branching the computation path, it is clear that obtaining sufficiently sharp error bounds to be able to decide with certainty is preferable.

The combinatorial solution to a linear program consists of its optimal basis.

Once the optimal basis has been found, the optimal solution of the standard-form dual is determined as column 0 of the tableau. However, if the solution to the Coho primal is sought after, a slightly different linear system has to be solved. Whereas the solution to the standard dual is:

$$x^{\mathcal{S}} = (A^{\mathcal{S}}{}_{:,\mathcal{B}})^{-1}\, b^{\mathcal{S}} \tag{5.1}$$

the solution to the Coho primal is:

$$x^{\mathcal{C}} = (A^{\mathcal{C}}{}_{\mathcal{B},:})^{-1}\, b^{\mathcal{C}}{}_{\mathcal{B}} = ((A^{\mathcal{S}}{}_{:,\mathcal{B}})^{T})^{-1}\, b^{\mathcal{C}}{}_{\mathcal{B}} \tag{5.2}$$

The left-hand sides of two linear systems differ only through a transposition. The conditioning of any matrix is the same as that of its transpose. Moreover, only trivial changes are needed to an algorithm that solves linear systems of the first type to make it work for the second type. Thus it is sufficient to analyze linear systems of the first type, under the implicit assumption that the results also apply to the extraction of the optimal solution to the primal.

Whereas in some cases the optimal solution of a linear program is the result of interest, there are instances where the optimal cost is the sought-after answer to the problem. In such a case, the components of the error in the optimal solution that are orthogonal to the optimization direction are harmless. This opens the possibility of trading accuracy of the optimal solution in directions that are orthogonal to the optimization direction for accuracy in the optimization direction.

Ideally, we would like to be able to characterize the accuracy of the computed optimal solution through a closed-form expression depending on the machine precision and on the matrix structure.

In the case of the optimal cost, the comparative flexibility of the constraints suggests that an error bound can be established that depends only on the machine precision and on the dimension of the system.

Whereas these problems have not been solved completely, some inroads have been made into them. These advances form the subject of this chapter. As the

reduction of an independent Coho linear subsystem to independent cycles has been described in the previous chapter, the focus here will be on the solving of the cycles. The first section presents a new algorithm for solving cycles as part of computing tableau columns and the error bound that is thus achieved. The second section is concerned with a way of obtaining a better estimate of the optimal cost in cases where the optimal basis is ill-conditioned.

5.1 Error Bound on Cycle Solution

Section 4.2 presents a linear-time algorithm for solving the cycles that appear in Coho linear systems. The key to keeping the running time linear is the calculation of only one component of the solution by means of a direct formula that takes $O(n)$ time. The other components are computed recursively starting from the first component at the cost of $O(1)$ each. Clearly, the recursive computation accumulates error. This can be avoided by the direct use of formulas similar to (4.9) in order to determine each component of the solution. Obviously this increases the running time of the algorithm to $O(n^2)$. However, even with this modification, only running error bounds on the solution to the cycle are available.

This section examines an alternative way of solving a cycle and of computing a bound on the error in the solution.

5.1.1 Algorithm for Solving Cycles

Let the cycle to solve be described by the equation:

$$Ax = y \tag{5.3}$$

where $A \in \mathbf{R}^{n \times n}$, $y \in \mathbf{R}^n$. Matrix A is supposed to be in normalized form as per equation (4.4). Then it is possible to express A as:

$$A = SBS^{-1} \tag{5.4}$$

where S is a diagonal matrix:

$$S = \text{diag}(s) \tag{5.5}$$

and B is a particular case of a cyclic matrix in which all off-diagonal elements are equal:

$$B_{i,j} = \begin{cases} 1 & \text{if } j = i \\ -\beta & \text{if } j = (i \bmod n) + 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

From equation (5.4) it results immediately that:

$$B = S^{-1}AS \tag{5.7}$$

Simple computations show that:

$$(S^{-1}AS)_{i,j} = \begin{cases} 1 & \text{if } j = i \\ -\alpha_i \frac{s_{(i \bmod n)+1}}{s_i} & \text{if } j = (i \bmod n) + 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

Equation (5.7) implies that:

$$B_{i,(i \bmod n)+1} = (S^{-1}AS)_{i,(i \bmod n)+1} \quad \forall i = 1, \ldots, n \tag{5.9}$$

which is equivalent to:

$$\beta = \alpha_i \frac{s_{(i \bmod n)+1}}{s_i} \quad \forall i = 1, \ldots, n \tag{5.10}$$

Memberwise multiplication of the equations above for $i = 1, \ldots, n$ yield:

$$\beta^n = \prod_{i=1}^{n} (\alpha_i \frac{s_{(i \bmod n)+1}}{s_i}) = (\prod_{i=1}^{n} \alpha_i) \frac{\prod_{i=1}^{n} s_{(i \bmod n)+1}}{\prod_{i=1}^{n} s_i} = \prod_{i=1}^{n} \alpha_i \tag{5.11}$$

Thus,

$$\beta = \sqrt[n]{\prod_{i=1}^{n} \alpha_i} \tag{5.12}$$

The combination of equations (4.5) and (5.12) yields:

$$\beta = \sqrt[n]{P_n} \tag{5.13}$$

The values of s can be obtained from (5.10):

$$s_k = \begin{cases} 1 & \text{if } k = 1 \\ \frac{s_{k-1}\beta}{\alpha_{k-1}} = \prod_{i=1}^{k-1} \frac{\beta}{\alpha_i} & \text{if } 1 < k \leq n \end{cases} \tag{5.14}$$

Matrix S is not unique: multiplication of S by any non-zero real number yields another matrix that satisfies (5.3).

Matrix B with the structure as defined by (5.6) falls within the category of *circulant* matrices or, for short, circulants [Hig96, p.469]. As a circulant, matrix B has the property that it is diagonalized by the Fourier transform matrix F_n:

$$F_n B F_n^{-1} = \Lambda = \text{diag}(\lambda) \tag{5.15}$$

Also as a property of circulants, vector λ contains the eigenvalues of matrix B and it satisfies:

$$\lambda = F_n b \tag{5.16}$$

where b is the first column of matrix B:

$$b = B_{:,1} = \text{col}(1, 0, \ldots, 0, -\beta) \tag{5.17}$$

By definition:

$$(F_n)_{i,j} = e^{\sqrt{-1}\ 2\pi(i-1)(j-1)/n} \tag{5.18}$$

As b contains only two non-zero elements, a closed form of the eigenvalues of B is easy to obtain:

$$\lambda_i = 1 - \beta e^{\sqrt{-1}\ 2\pi(i-1)/n} \tag{5.19}$$

Equation (5.15) implies that B can be expressed as:

$$B = F_n^{-1} \Lambda F_n \tag{5.20}$$

and

$$B^{-1} = F_n^{-1}\Lambda^{-1}F_n \tag{5.21}$$

From (5.4) and because S is diagonal it follows that:

$$A^{-1} = SB^{-1}S^{-1} \tag{5.22}$$

Equations (5.21) and (5.22) combined yield:

$$A^{-1} = SF_n^{-1}\Lambda^{-1}F_nS^{-1} \tag{5.23}$$

and, for the solution to the cycle:

$$x = A^{-1}y = SF_n^{-1}\Lambda^{-1}F_nS^{-1}y \tag{5.24}$$

The computations of s and λ take linear time. Fast Fourier transform algorithms take $O(n \log n)$ time. Consequently (5.24) defines an $O(n \log n)$ time algorithm for solving a cycle.

5.1.2 Estimation of Cycle Condition Number

The direct and inverse Fourier transforms are known to be quite stable. The algorithm defined by (5.24) might offer higher accuracy than the linear-time one presented earlier on. As it is more expensive, it will be used only when the linear-time algorithm fails to give satisfactory results.

The expression of cycle matrix A as the product of matrices with simpler structure enables the determination of a closed-form expression for the error in the solution to the cycle. Error analysis is straightforward for diagonal matrices.

Numerical stability properties are also known for circulant matrices. For one thing, the singular values of a circulant are the absolute values of its eigenvalues. Quantity P_n is a product of real numbers, so it is itself a real number. Quantity β is the n^{th} root of P_n, so it can be expressed as:

$$\beta = |\beta| e^{\sqrt{-1}((p+2k)\pi/n)}, \quad \text{where } k \in \{0, \ldots, n-1\} \tag{5.25}$$

and

$$p = \begin{cases} 0 & \text{if } P_n > 0 \\ 1 & \text{if } P_n < 0 \end{cases} \tag{5.26}$$

The eigenvalues of matrix B are given by (5.19). By combining it with (5.25) the following equation results for its singular values:

$$\sigma_i = \sqrt{1 + |\beta|^2 - 2|\beta| \cos\theta} \qquad i = 1, \ldots, n \tag{5.27}$$

where

$$\theta = \frac{(p + 2k + 2(i-1))\pi}{n} \tag{5.28}$$

A classical measure of the sensitivity of a linear system to numerical errors is the condition number of its left-hand side, which is the ratio of its largest to its smallest singular value:

$$K(B) = \frac{\sigma_{\max}}{\sigma_{\min}} \tag{5.29}$$

where $\sigma_{\max} = \max_i \sigma_i$ and $\sigma_{\min} = \min_i \sigma_i$.

It is easily seen that $\sigma_{\max}$ as a function of θ is realized for the lowest value of $\cos\theta$, i.e the value of θ that is closest to $\pi \pmod{2\pi}$:

$$\sigma_{\max} = \begin{cases} 1 + |\beta| & \text{if } (P_n > 0) = (n \text{ is even}) \\ \sqrt{(1 + |\beta|)^2 - 2|\beta|(1 - \cos\frac{\pi}{n})} & \text{if } (P_n > 0) = (n \text{ is odd}) \end{cases} \tag{5.30}$$

Equation (5.30) can be expressed more simply as:

$$\sigma_{\max} \leq 1 + |\beta| \tag{5.31}$$

The value $\sigma_{\min}$, as a function of θ, is realized for the highest value of $\cos\theta$, i.e the value of θ that is closest to $0 \pmod{2\pi}$:

$$\sigma_{\min} = \begin{cases} |1 - |\beta|| & \text{if } P_n > 0 \\ \sqrt{|\beta|^2 - 2|\beta| cos\frac{\pi}{n} + 1} & \text{if } P_n < 0 \end{cases} \tag{5.32}$$

For the $P_n < 0$ case, the value of $K(B)$ can be determined from equations (5.31) and (5.32):

$$K(B) = \sqrt{\frac{|\beta|^2 + 2|\beta| + 1}{|\beta|^2 - 2|\beta| \cos\frac{\pi}{n} + 1}} \tag{5.33}$$

The minimum value of this expression for $|\beta| > 0$ is:

$$K(B) = \frac{1}{\sin\frac{\pi}{2n}} \tag{5.34}$$

which shows that the system cannot be ill-conditioned when $P_n < 0$ for the values of n of interest in Coho ($n \leq 20$). This agrees with equation (4.9), which suggests that ill-conditioning is related to P_n being close to 1. Only the case $P_n > 0$ is considered henceforth. For this case, $\sigma_{\min}$ simplifies to:

$$\sigma_{\min} = |1 - |\beta|| \tag{5.35}$$

From equations (5.31) and (5.35) it follows that:

$$K(B) \leq \frac{1 + |\beta|}{|1 - |\beta||} \tag{5.36}$$

The condition number of a diagonal matrix is the ratio of its largest diagonal element in absolute value to its smallest:

$$K(S) = \frac{\max_i |s_i|}{\min_i |s_i|} \tag{5.37}$$

As with any matrix, $K(S) = K(S^{-1})$.

This enables us to establish an upper bound on the condition number of A:

$$K(A) \leq K(S)K(B)K(S^{-1}) = K(B)K(S)^2 \tag{5.38}$$

5.1.3 Error Bound on Solution to Cycle

Formula (5.38), although it has the merit of establishing an upper bound on the error in the solution to a cycle, can be rather pessimistic. Known results on circulant systems will be combined with error computations for the scaling matrices to obtain a tighter error bound on the cycle solution.

Forward Error Bound for the Cycle Circulant

In addition to the properties of circulant matrices presented in the previous section, more results about them can be found in [Lin92]. More specifically, a normwise forward error bound is established for circulant systems. The result, which holds if the input data is free from errors, the only source of errors being roundoff in the algorithm, is the following:

$$\frac{\|\,\mathrm{fl}(x) - x\|}{\|x\|} \leq u(\Psi_{\mathrm{FFT}}(n)(K_{F,2}(B) + 2) + c_{C0}) \stackrel{\mathrm{def}}{=} f_B(n, u) \tag{5.39}$$

where $\mathrm{fl}(x)$ denotes the floating-point approximation of x, u denotes the machine precision, $c_{C0} = \sqrt{2} + 4$ is a small constant, $K_{F,2}(B)$ is a pseudo-condition number defined as:

$$K_{F,2}(B) = \frac{\sqrt{\sum_k |\lambda_k|^2}}{\min_k |\lambda_k|} \tag{5.40}$$

and $\Psi_{\mathrm{FFT}}(n)$ is a function that characterizes the stability of the particular FFT algorithm employed to solve the circulant system. For example, the radix 2, Cooley-Tukey algorithm has [Lin92]:

$$\Psi_{\mathrm{FFT}}(n) \leq c_\Psi \log_2 n \tag{5.41}$$

where $c_\Psi = 1.06 \times 4^{3/2}$. From (5.19) it follows that:

$$|\lambda_k|^2 = 1 + \beta^2 - 2\beta \cos \frac{2\pi(k-1)}{n} \tag{5.42}$$

so

$$\begin{aligned} \textstyle\sum_{k=1}^n |\lambda_k|^2 &= \textstyle\sum_{k=1}^n \left(1 + \beta^2 - 2\beta \cos \frac{2\pi(k-1)}{n}\right) \\ &= n(1+\beta^2) - 2\beta \textstyle\sum_{k=1}^n \cos \frac{2\pi(k-1)}{n} \end{aligned} \tag{5.43}$$

A well-known trigonometric result is that:

$$\sum_{k=1}^{n} \cos \frac{2\pi(k-1)}{n} = 0 \tag{5.44}$$

which, introduced in (5.43), yields:

$$\sum_{k=1}^{n} |\lambda_k|^2 = n(1+\beta^2) \tag{5.45}$$

Similarly to (5.35), we have that:

$$\min_k |\lambda_k| \geq |1 - |\beta|| \tag{5.46}$$

The substitution of (5.45) and (5.35) in (5.40) yields:

$$K_{F,2}(B) \leq \frac{\sqrt{n(1+\beta^2)}}{|1-|\beta||} \tag{5.47}$$

The introduction of equation (5.41) and inequality (5.47) into inequality (5.39) yields the following forward error bound for matrix B:

$$f_B(n,u) \leq u\left(c_\Psi \log_2 n\left(\frac{\sqrt{n(1+\beta^2)}}{|1-|\beta||}+2\right)+c_{C0}\right) \tag{5.48}$$

Errors Affecting the Scaling Matrix

As a cycle matrix A consists of matrix B pre- and postmultiplied with matrix S and S^{-1}, respectively, the errors introduced by S and S^{-1} have to be considered as well. The coefficients α_i result from row scaling, so they will be affected by error:

$$\frac{|\,\mathrm{fl}(\alpha_i) - \alpha_i|}{|\alpha_i|} \leq u \tag{5.49}$$

The application of the error composition rule for multiplication leads to:

$$\frac{|\,\mathrm{fl}(P_k) - P_k|}{|P_k|} \leq (2k-1)u \leq (2n-1)u \quad \forall k = 1,\ldots,n \tag{5.50}$$

The computation of the quantity:

$$\beta^k = (P_n)^{\frac{k}{n}} \tag{5.51}$$

involves the power operation, which is not elementary. In general, the quantity x^y, where x and y are real numbers, is computed as $e^{y \ln x}$ and, when the inputs are free from errors, is affected by the following error [Mul97, p.179]:

$$\frac{|\,\mathrm{fl}(x^y) - x^y|}{|x^y|} = |e^{uy \ln x} - 1| \tag{5.52}$$

or, to the first order:

$$\frac{|\,\mathrm{fl}(x^y) - x^y|}{|x^y|} \approx uy \ln x \tag{5.53}$$

In order to account for the error in the inputs to the power, we use its linearization:

$$(x(1+\delta_x))^{y(1+\delta_y)} \approx x^y(1 + y(\delta_x + \ln x\ \delta_y)) \tag{5.54}$$

Both k and n are error-free integers, so the error that affects their computed ratio is:

$$\frac{|\,\mathrm{fl}(\frac{k}{n}) - \frac{k}{n}|}{|\frac{k}{n}|} \leq u \tag{5.55}$$

Based on the rules expressed by (5.53) and (5.54) and on the error bounds on the inputs given by (5.50) and (5.55), the following bound is obtained for β^k computed by formula (5.51):

$$\frac{|\,\mathrm{fl}(\beta^k) - \beta^k|}{|\beta^k|} \leq \frac{k}{n}((2k-1)u + |\ln|P_n||\ u) + \frac{k}{n}|\ln|P_n||\ u \tag{5.56}$$

which, because $\frac{k}{n} < 1$, simplifies to:

$$\frac{|\,\mathrm{fl}(\beta^k) - \beta^k|}{|\beta^k|} \leq (2n + 2|\ln|P_n||)u \tag{5.57}$$

Formula (5.14) of s_k can be rewritten as:

$$s_k = \frac{\beta^k}{P_k} \tag{5.58}$$

When computed by the above formula, the error that affects s_k is:

$$\begin{aligned} \frac{|\,\mathrm{fl}(s_k) - s_k|}{|s_k|} &\leq \frac{|\,\mathrm{fl}(\beta^k) - \beta^k|}{|\beta^k|} + \frac{|\,\mathrm{fl}(P_k) - P_k|}{|P_k|} + u \\ &\leq (2n + 2|\ln|P_n||)u + (2k-1)u + u \\ &\leq (4n + 2|\ln|P_n||)u \end{aligned} \tag{5.59}$$

The introduction of (5.13) into (5.59) leads to:

$$\frac{|\,\mathrm{fl}(s_k) - s_k|}{|s_k|} \leq (4n + 2n|\ln|\beta||)u \tag{5.60}$$

It is easy to see that the same error bounds can be attained for the components of S^{-1}.

Error Bound for Cycle Solution

Let us now establish an error bound on the solution of the cycle. Consider that the linear system is solved in the following steps:

$$\begin{aligned} t' &= S^{-1}y \\ t'' &= B^{-1}t' \\ x &= St'' \end{aligned} \tag{5.61}$$

For all three steps of the algorithm we have to consider two types of error in the output. The first type is due to unavoidable roundoff, which is present even if the input is error-free. The second type is due to the propagation of input error into the output. The relative error in the input appears in the output magnified by the condition number of the system matrix:

$$x_{\text{out}} = Mx_{\text{in}} \Rightarrow \frac{\|\Delta x_{\text{out}}\|}{\|x_{\text{out}}\|} \leq K(M)\frac{\|\Delta x_{\text{in}}\|}{\|x_{\text{in}}\|} \tag{5.62}$$

The first step of the algorithm consists of elementwise multiplication:

$$t'_k = s_k y_k \tag{5.63}$$

Quantity y is considered free from errors. A bound on the componentwise relative error affecting t' follows immediately:

$$\frac{|\,\text{fl}(t'_k) - t'_k|}{|t'_k|} \leq \frac{|\,\text{fl}(s_k) - s_k|}{|s_k|} + u \tag{5.64}$$

The introduction of inequality (5.60) into the one above yields:

$$\frac{|\,\text{fl}(t'_k) - t'_k|}{|t'_k|} \leq ((4n+1) + 2n|\ln|\beta||)u \stackrel{\text{def}}{=} f_S(n,u) \tag{5.65}$$

Both types of error can occur during multiplication by B^{-1}:

$$\frac{\|\,\text{fl}(t'') - t''\|}{\|t''\|} \leq f_B(n,u) + K(B)\frac{\|\,\text{fl}(t') - t'\|}{\|t'\|} \tag{5.66}$$

Componentwise bounds translate directly into normwise bounds, so it is possible to introduce (5.65) in the inequality above:

$$\frac{\|\,\text{fl}(t'') - t''\|}{\|t''\|} \leq f_B(n,u) + K(B)f_S(n,u) \tag{5.67}$$

The error bound for the final result is obtained by considering the multiplication by S:

$$\frac{\|\,\mathrm{fl}(x) - x\|}{\|x\|} \leq f_S(n,u) + K(S)\frac{\|\,\mathrm{fl}(t'') - t''\|}{\|t''\|} \tag{5.68}$$

By combining the equation above with (5.67) it results that:

$$\frac{\|\,\mathrm{fl}(x) - x\|}{\|x\|} \leq K(S)f_B(n,u) + (K(S)K(B)+1)f_S(n,u) \stackrel{\text{def}}{=} g(B,S,n,u) \tag{5.69}$$

The introduction of (5.36), (5.37), (5.48), and (5.65) into (5.69) leads to the error bound for the cycle solution:

$$\begin{aligned} g(B,S,n,u) &= \frac{\max\limits_i |s_i|}{\min\limits_i |s_i|}\left(c_\Psi \log_2 n\left(\frac{\sqrt{n(1+\beta^2)}}{|1-|\beta||} + 2\right) + c_{C0}\right)u \\ &+ \left(\frac{\max\limits_i |s_i|}{\min\limits_i |s_i|}\,\frac{1+|\beta|}{|1-|\beta||} + 1\right)((4n+1) + 2n|\ln|\beta||)u \end{aligned} \tag{5.70}$$

From (5.70), we see that errors can be large if $|\beta| \approx 1$ or $\frac{\max_i |s_i|}{\min_i |s_i|}$ is large. As mentioned earlier, the characterization of the errors caused by solving the circulant matrix assumes that β^n is a positive number. When $\beta^n \approx 1$, the denominator of (4.9) is close to zero, reflecting the near singularity of the system. When $\frac{\max_i |s_i|}{\min_i |s_i|}$ is large, the system is ill-conditioned due to extreme scaling that can lead to serious cancellation.

This analysis has not considered the fact that matrix B is affected by errors in the calculated value of β. This can be achieved by computing $\frac{d\|t''\|}{d\beta}$. A first step towards this can be the evaluation of $\frac{d\|B\|}{d\beta}$. This remains a topic for future research.

5.1.4 Summary

This section has analyzed the errors that affect the solutions of Coho linear systems. The left-hand side of such a linear system is a cyclic matrix A. In this section we have made progress on getting a more detailed characterization of the conditioning of A.

Cyclic matrices have been shown to be expressible as the product $A = S^{-1}BS$, where S is a diagonal matrix and B is a circulant matrix. Closed-form formulas for the eigenvalues, eigenvectors, singular values, and singular vectors of matrices B and S have been presented.

The singular values of matrix B are proportional to the eigenvalues of the same matrix. Matrix B has at most one small eigenvalue and at most one small singular value. The eigenvalues and eigenvectors of matrix A coincide with those of matrix B. Therefore matrix A has at most one small eigenvalue.

The conditioning of matrix A is determined by the conditioning of both B and S. Unlike B, matrix S can have several small singular values. The precise characterization of the conditioning of matrix A requires knowledge of its singular values. Whereas a time-effective method for this computation has not been found, the condition number of A can be bounded by the product of the condition numbers of B and S, which we know exactly.

An analysis of the forward error that affects the solution of a cycle has been presented. However, this analysis does not lead to a closed-form expression for the error bound, which means that further research is necessary in this direction. The most significant improvement can potentially result from the determination of the singular values of matrix A. This too remains a topic for further research.

5.2 Estimation of Optimal Cost

In this section we seek ways to compute the optimal cost of a Coho linear program as accurately as possible. The computation of the optimal cost continues naturally from that of the optimal solution.

Consider a Coho linear program CLP($A^{\mathcal{C}}, b^{\mathcal{C}}, c^{\mathcal{C}}$) whose optimal basis is $\mathcal{B}$. The optimal cost can be computed as the dot product of the cost vector and the

optimal solution either in the Coho primal:

$$(c^{\mathcal{C}})^T x^{\mathcal{C}} = (c^{\mathcal{C}})^T \left(A^{\mathcal{C}}{}_{\mathcal{B},:}^{-1} b^{\mathcal{C}}{}_{\mathcal{B}}\right) \tag{5.71}$$

or in the standard-form dual:

$$(c^{\mathcal{S}})^T x^{\mathcal{S}} = (b^{\mathcal{C}}{}_{\mathcal{B}})^T \left((A^{\mathcal{C}}{}_{\mathcal{B},:}^{T})^{-1} c^{\mathcal{C}}\right) \tag{5.72}$$

The second method will be considered as it builds directly on the results of the previous section. Moreover, it is not difficult to show that the same results apply to the first method.

Of the two operands of the dot product in (5.71), the dual cost vector is free from errors. The dual optimal solution, however, is affected by an error that has been bounded in the previous section.

The following forward error bound is known for the dot product of two vectors [Hig96, p.69]:

$$|\,\mathrm{fl}(x^T y) - x^T y| \leq \frac{nu}{1-nu} |x|^T |y| \quad \text{where } x, y \in \mathbf{R}^n \tag{5.73}$$

This shows that high accuracy is not guaranteed if $|x^T y| \ll |x|^T |y|$. On the other hand, the accuracy of the computed dot product is high if all $x_i y_i$ terms have the same sign.

An optimal solution of a linear program in standard form is feasible, so all the components of $x^{\mathcal{S}}$ must be non-negative. On the other hand, the signs of $b^{\mathcal{C}}$ can in general be arbitrary, holding the potential for disastrous cancellation in the dot product with the dual optimal solution.

This problem can be circumvented by making a change of variables such that the origin lies inside every projection polygon. This will result in the origin lying inside the feasible region of the linear program. The feasible region of the Coho LP is defined by the inequality:

$$A^{\mathcal{C}} x^{\mathcal{C}} \geq b^{\mathcal{C}} \tag{5.74}$$

As the origin is inside the feasible region, the above inequality must hold for it:

$$A^{\mathcal{C}}0 \geq b^{\mathcal{C}} \tag{5.75}$$

i.e.:

$$b^{\mathcal{C}} \leq 0 \tag{5.76}$$

Because $c^{\mathcal{S}} = b^{\mathcal{C}}$, this implies that:

$$|(c^{\mathcal{S}})^T x^{\mathcal{S}}| = |(c^{\mathcal{S}})^T||x^{\mathcal{S}}| \tag{5.77}$$

The application of rule (5.73) to equation (5.77) leads to the following formula for the relative error e_{DP} in the computed cost due to the dot product operation:

$$e_{DP} = \frac{|\,\mathrm{fl}((c^{\mathcal{S}})^T x^{\mathcal{S}}) - (c^{\mathcal{S}})^T x^{\mathcal{S}}|}{|(c^{\mathcal{S}})^T x^{\mathcal{S}}|} \leq nu \tag{5.78}$$

The approximation $\frac{1}{1-nu} = 1$ introduces negligible errors for for the values of $n \leq 20$ typical of Coho.

Formula (5.78) bounds the error that would affect the computed optimal cost if the box c and x were free from errors. As it has been seen, x is a computed quantity affected by an error bounded by formula (5.69). The error in the optimal solution translates into the following absolute error in the optimal cost:

$$E_x = |c^T\,\mathrm{fl}(x) - c^T x| = |c^T(\mathrm{fl}(x) - x)| \leq \|c\|\ \|\mathrm{fl}(x) - x\| \tag{5.79}$$

As the magnitude of the cost vector is irrelevant, we can assume $\|c\| = 1$ and (5.79) becomes:

$$E_x \leq \|\mathrm{fl}(x) - x\| \tag{5.80}$$

The introduction of (5.69) into (5.80) yields:

$$E_x \leq g(B, S, n, u)\,\|x\| \tag{5.81}$$

The total absolute error E in the cost is obtained by adding errors e_{DP} and E_x:

$$E = E_x + e_{DP}\,c^T x \leq E_x + nu\,c^T x \tag{5.82}$$

Equation (5.82) says that, assuming a unit cost vector, the error in the computed cost is bounded by the error in computing the optimal point plus nu times the exact cost. If the optimal basis is ill-conditioned, the error in calculating the optimal point may be large. This motivates considering nearby branches where the error can be reduced.

5.2.1 Use of Nonoptimal Basis

When the optimal basis of a linear program is highly ill-conditioned, the error bound on the optimal cost computed by the methods discussed so far might be unacceptably large. Therefore a better overapproximation of the optimal cost may be obtainable by computing it for a slightly primal-infeasible well-conditioned basis.

More concretely, consider a linear program with cost vector c and optimal vertex x. Let the optimal cost be approximated by the cost of the vertex x'. Let the computed costs of x and x' be $\mathrm{fl}(c^T x)$ and $\mathrm{fl}(c^T x')$, respectively. The computed cost of x' differs from the true optimal cost by:

$$\begin{aligned} |\,\mathrm{fl}(c^T x') - c^T x| &= |(\mathrm{fl}(c^T x') - c^T x') + (c^T x' - c^T x)| \\ &\leq |\,\mathrm{fl}(c^T x') - c^T x'| + |c^T x' - c^T x| \end{aligned} \tag{5.83}$$

If this quantity is less than the error $\mathrm{fl}(c^T x) - c^T x$ that affects the computed cost of x, then x' offers a better approximation of the true optimal cost of the linear program.

The error component $\mathrm{fl}(c^T x') - c^T x'$ depends on the conditioning of the basis associated with x'. The error component due to the difference between the true costs is:

$$|c^T x' - c^T x| = \|x' - x\| \left\|\mathrm{proj}_{(x'-x)}\, c\right\| \tag{5.84}$$

Therefore in order for a vertex x' to yield a good approximation of the optimal cost, x' must satisfy the following properties:

- x' is not far from the optimal vertex x

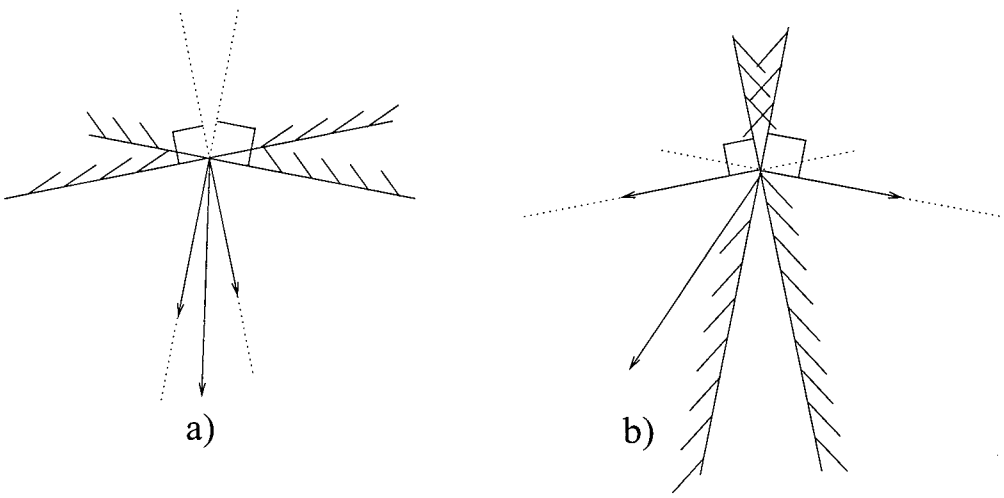

Figure 5.1: Types of optimal 2D vertices: a) highly obtuse optimal vertex and highly acute cost cone; b) highly acute optimal vertex and highly obtuse cost cone

- x' is on a near-normal to the cost vector through the optimal vertex
- the basis corresponding to x' is well-conditioned

A square matrix is ill conditioned if at least one of its rows is nearly equal to a linear combination of the other rows. The maximum number of rows that are nearly equal to linear combinations of other rows represents the degree of ill conditioning. Clearly the degree of ill conditioning depends on what is meant by "nearly equal".

A matrix and its transpose share the same conditioning. This is reflected by the fact that the characterization of matrix ill-conditioning with respect to rows is also true of columns.

In the case of Coho linear programs, basic columns in the dual represent inward halfspace normals in the primal. The feasibility of a basis means that the primal cost vector is a positive combination of basic columns. Therefore at any feasible basis the primal cost vector lies inside the cone generated by the basic inward halfspace normals, also called the basic cost cone.

Let us now consider the simple case of a two-dimensional linear program

whose optimal basis is ill-conditioned. Ill conditioning has a simple geometric interpretation in this case: the two lines that determine the optimal vertex are nearly parallel. This means that the optimal vertex is either highly obtuse or highly acute.

It is easily seen that the angle of the optimal vertex is the supplement of the angle of the optimal cost cone. Therefore when the optimal cost cone is highly acute the optimal vertex is highly obtuse and vice versa.

The optimal basis is feasible, so the optimization direction lies inside the optimal cost cone. Consequently the angle between the optimization direction and either side of the optimal vertex cannot differ from $\pi/2$ by more than the angle of the optimal cost cone. Therefore, when the optimal cost cone is very acute, i.e. when the optimal vertex is very obtuse, the sides of the optimal vertex are nearly perpendicular to the optimization direction.

This suggests that vertices that lie on the lines that determine the optimal vertex are good choices for approximating the optimal cost. Moreover, the more ill-conditioned the system is, the better the approximation offered by such points.

A vertex on one of the lines that determine the optimal vertex is obtained by replacing the constraint corresponding to the other line with some other constraint.

If the optimal vertex is highly acute, then the optimal cost cone is very wide, so the orientation of the optimization direction relative to the boundary lines cannot be characterized. However, highly acute vertices can be precluded by an appropriate scaling of the LP, not discussed further in this thesis.

Let us now try to extend the approximation idea presented above to higher dimensions. Consider a Coho linear program of dimension d whose optimal basis is ill-conditioned. For simplicity, we assume its degree of ill-conditioning to be 1, but we expect the generalization to higher degrees to be straightforward.

The ill conditioning of the basis means that there exists a subset of its columns such that any column in the subset is almost a linear combination of the other columns in the subset. We term the columns in the aforesaid subset interde-

pendent and the other columns independent.

Let G_k be the linear subspace generated by all the basic columns except k. This subspace has dimension $d-1$, so it is, in fact, a hyperplane. Let k be one of the interdependent columns. The fact that column k is almost a linear combination of the columns that generate G_k means that the projection of column k onto the normal to G_k is very small. This implies that, for any two interdependent columns k_1 and k_2, hyperplanes G_{k_1} and G_{k_2} are nearly parallel.

Each of the d hyperplanes generated by a set of $d-1$ inward face normals contains a face of the optimal cost cone. Hence the faces of the optimal cone corresponding to interdependent basic columns are nearly parallel to one another. Therefore the wedge formed by two such faces can be either very thin or very wide.

The cost vector must lie inside the wedge formed by a pair of adjacent faces of the optimal cost cone. If the wedge is very wide, no inference can be made about the direction of the cost vector. This is the case for highly acute vertices, and can be excluded by scaling.

For a highly obtuse vertex, the wedge is very thin, and the cost vector is nearly parallel to either face of the wedge. A normal to a face of the wedge will be almost perpendicular to the cost vector. Therefore the points on such a normal may suitably approximate the optimal cost.

A vertex on the normal to the face of the optimal cost cone contained in hyperplane G_k can be obtained by replacing basic column k with another column. In the Coho LP, this amounts to replacing one of the constraints in the optimal basis.

If an ill-conditioned optimal basis is characterized by the existence of a pair of faces of the optimal cost cone that form a very thin wedge, then the optimal vertex is said to be highly obtuse and the optimal cost cone is termed highly acute. Otherwise, the vertex is said to be highly acute and the optimal cost cone is termed highly obtuse.

The error introduced by using a non-optimal basis is proportional to the distance of the approximating vertex to the optimal vertex and to the projection of the cost vector onto the line passing through the two vertices. In order to bound the approximation error, we have to bound these two quantities.

5.2.2 Error Introduced by Dropping One Constraint

This subsection considers the case of a d-dimensional cone and a vector inside the cone. A bound on the projection of this vector onto the normal to an arbitrary face of the cone is established. Practically, we are interested in very flat and narrow cones as described in the proposed approximation solution.

Lemma 1 *Let* $\mathbf{u}$ *and* $\mathbf{v}$ *be vectors in* $\mathbf{R}^n \setminus \{\mathbf{0}\}$.

Let

$$\theta = \measuredangle(\mathbf{u}, \mathbf{v}) \tag{5.85}$$

Let $w = w(\theta)$ *be a function defined as:*

$$w(\theta) = \begin{cases} 1 & \text{if } \theta \in [0, \pi/2] \\ \sin\theta & \text{if } \theta \in (\pi/2, \pi] \end{cases} \tag{5.86}$$

Then

$$\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|} \leq w(\theta) \tag{5.87}$$

Proof It is simple to establish that:

$$\|\mathbf{u}+\mathbf{v}\| = \sqrt{\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 + 2\|\mathbf{u}\|\|\mathbf{v}\|\cos\theta} \tag{5.88}$$

Division by $\|\mathbf{u}\|$ turns equation (5.88) into:

$$\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|} = \sqrt{1 + \left(\frac{\|\mathbf{v}\|}{\|\mathbf{u}\|}\right)^2 + 2\frac{\|\mathbf{v}\|}{\|\mathbf{u}\|}\cos\theta} \tag{5.89}$$

Let:

$$\mu = \frac{\|\mathbf{v}\|}{\|\mathbf{u}\|} \tag{5.90}$$

Substitution of (5.90) into (5.89) yields:

$$\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|} = \sqrt{1+\mu^2+2\mu\cos\theta} \tag{5.91}$$

The value of μ for which the expression of $\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|}$ is minimized is:

$$\mu'_{\min} = -\cos\theta \tag{5.92}$$

Because both $\|\mathbf{u}\|$ and $\|\mathbf{v}\|$ are positive, μ is restricted to positive values. By bringing in this restriction the value of μ for which $\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|}$ is minimized becomes:

$$\mu_{\min} = \max(\mu'_{\min}, 0) = \max(-\cos\theta, 0) \tag{5.93}$$

$$\mu_{\min} = \begin{cases} 0 & \text{if } \theta \leq \pi/2 \\ -\cos\theta & \text{if } \theta > \pi/2 \end{cases} \tag{5.94}$$

If $\theta \leq \pi/2$, then $\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|}$ is minimized with $\mu = 0$ and:

$$\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|} = 1 \tag{5.95}$$

If $\theta > \pi/2$, then $\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|}$ is minimized with $\mu = \cos\theta$ and:

$$\frac{\|\mathbf{u}+\mathbf{v}\|}{\|\mathbf{u}\|} = \sqrt{(-\cos\theta)^2 + 2\cos\theta(-\cos\theta) + 1} = \sqrt{1-\cos^2\theta} = \sin\theta \tag{5.96}$$

Equations (5.95) and (5.96) establish the result of the lemma.

□

The proposed approximation relies on the cost vector being nearly parallel to any face of the optimal cone. This nearness is measured by the projection of the cost function on the normal to a face of the optimal cone. The following theorem establishes a bound on this projection:

Theorem 1 *Let* $\{\mathbf{u}_i : i = 1, \ldots, n\}$ *be a set of n-dimensional* unit *vectors such that* $\mathbf{u}_1, \ldots, \mathbf{u}_{n-1}$ *are linearly independent.*

Let $\mathbf{d}_\perp$ *be a* unit *vector such that*

$$\mathbf{d}_\perp \cdot \mathbf{u}_i = 0, \quad \forall i = 1, \ldots, n-1 \tag{5.97}$$

and

$$\mathbf{u}_n \cdot \mathbf{d}_\perp \geq 0 \tag{5.98}$$

Let

$$u_{n_\perp} = \mathbf{u}_n \cdot \mathbf{d}_\perp \tag{5.99}$$

and

$$\mathbf{u}_{n_\parallel} = \mathbf{u}_n - u_{n_\perp} \mathbf{d}_\perp \tag{5.100}$$

Let

$$\varphi = \max_{(b_1, \ldots, b_{n-1}) \in \mathbf{R}_+^{n-1}} \measuredangle \left(\mathbf{u}_{n_\parallel}, \sum_{i=1}^{n-1} b_i \mathbf{u}_i \right) \tag{5.101}$$

Let $\mathbf{f} \in \mathbf{R}^n$ *be a* unit *vector such that*

$$\mathbf{f} = \sum_{i=1}^{n} a_i \mathbf{u}_i \tag{5.102}$$

where

$$a_i \geq 0, \quad \forall i = 1, \ldots, n-1 \tag{5.103}$$

Let

$$f_\perp = \mathrm{proj}_{\mathbf{d}_\perp} \mathbf{f} \tag{5.104}$$

Then

$$0 \leq f_\perp \leq \frac{u_{n_\perp}}{\sqrt{u_{n_\perp}^2 + w^2(\varphi)\left(1 - u_{n_\perp}^2\right)}} \tag{5.105}$$

Proof The combination of equations (5.102) and (5.104) yields:

$$f_\perp = \mathbf{f} \cdot \mathbf{d}_\perp = (\sum_{i=1}^{n} a_i \mathbf{u}_i) \cdot \mathbf{d}_\perp = \sum_{i=1}^{n} a_i (\mathbf{u}_i \cdot \mathbf{d}_\perp) \tag{5.106}$$

The introduction of definition (5.97) into equation (5.106) yields:

$$f_\perp = a_n (\mathbf{u}_n \cdot \mathbf{d}_\perp) \tag{5.107}$$

By definition (5.99) equation (5.107) becomes:

$$f_\perp = a_n u_{n_\perp} \tag{5.108}$$

By definitions (5.98) and (5.99):

$$u_{n_\perp} \geq 0 \tag{5.109}$$

Thus, definition (5.103) and inequality (5.109) yield $f_\perp \geq 0$, establishing the first inequality of the theorem.

Let

$$\mathbf{f}_{||} = \mathbf{f} - f_\perp \mathbf{d}_\perp \tag{5.110}$$

The introduction of (5.102) and (5.108) into (5.110) yields:

$$\mathbf{f}_{||} = \sum_{i=1}^{n} a_i \mathbf{u}_i - a_n u_{n_\perp} \mathbf{d}_\perp = \sum_{i=1}^{n-1} a_i \mathbf{u}_i + a_n \mathbf{u}_n - a_n u_{n_\perp} \mathbf{d}_\perp = \sum_{i=1}^{n-1} a_i \mathbf{u}_i + a_n (\mathbf{u}_n - u_{n_\perp} \mathbf{d}_\perp) \tag{5.111}$$

The combination of (5.100) and (5.111) leads to:

$$\mathbf{f}_{||} = a_n \mathbf{u}_{n_{||}} + \sum_{i=1}^{n-1} a_i \mathbf{u}_i \tag{5.112}$$

The application of lemma 1 to the vectors in equation (5.112) results in:

$$\left\| \mathbf{f}_{||} \right\| \geq \left\| a_n \mathbf{u}_{n_{||}} \right\| w \left(\measuredangle \left(a_n \mathbf{u}_{n_{||}}, \sum_{i=1}^{n-1} a_i \mathbf{u}_i \right) \right) \tag{5.113}$$

By (5.103) a_n must be positive, so:

$$\measuredangle \left(a_n \mathbf{u}_{n_{||}}, \sum_{i=1}^{n-1} a_i \mathbf{u}_i \right) = \measuredangle \left(\mathbf{u}_{n_{||}}, \sum_{i=1}^{n-1} a_i \mathbf{u}_i \right) \tag{5.114}$$

By definition (5.101):

$$\measuredangle\left(\mathbf{u}_{n_{\|}}, \sum_{i=1}^{n-1} a_i \mathbf{u}_i\right) \leq \varphi \tag{5.115}$$

The function w is monotonically decreasing, so the inequality above can be turned into:

$$w\left(\measuredangle\left(\mathbf{u}_{n_{\|}}, \sum_{i=1}^{n-1} a_i \mathbf{u}_i\right)\right) \geq w\left(\varphi\right) \tag{5.116}$$

Combined, inequalities (5.113) and (5.116) yield:

$$\left\|\mathbf{f}_{\|}\right\| \geq \left\|a_n \mathbf{u}_{n_{\|}}\right\| w\left(\varphi\right) \tag{5.117}$$

In general, for any 2 vectors $\mathbf{v}$ and $\mathbf{w}$, the Pythagorean theorem holds:

$$\|\mathbf{v}\|^2 = \|\mathrm{proj}_{\mathbf{w}}\, \mathbf{v}\|^2 + \|\mathbf{v} - \mathrm{proj}_{\mathbf{w}}\, \mathbf{v}\|^2 \tag{5.118}$$

For $\mathbf{f}$ and $\mathbf{d}_{\perp}$ it becomes:

$$\left\|\mathbf{f}_{\|}\right\|^2 + f_{\perp}^2 = \|\mathbf{f}\|^2 = 1 \tag{5.119}$$

or:

$$f_{\perp} = \sqrt{1 - \left\|\mathbf{f}_{\|}\right\|^2} \tag{5.120}$$

By combining inequality (5.117) with equation (5.120) we get:

$$f_{\perp} \leq \sqrt{1 - \left(a_n w(\varphi) \left\|\mathbf{u}_{n_{\|}}\right\|\right)^2} \tag{5.121}$$

The introduction of (5.108) into (5.121) yields:

$$a_n u_{n_{\perp}} \leq \sqrt{1 - \left(a_n w(\varphi) \left\|\mathbf{u}_{n_{\|}}\right\|\right)^2} \tag{5.122}$$

By the Pythagorean theorem for $\mathbf{u}_n$ and $\mathbf{d}_{\perp}$:

$$\left\|\mathbf{u}_{n_{\|}}\right\|^2 = \|\mathbf{u}_n\|^2 - u_{n_{\perp}}^2 = 1 - u_{n_{\perp}}^2 \tag{5.123}$$

The introduction of (5.123) into (5.122) yields:

$$a_n u_{n_{\perp}} \leq \sqrt{1 - a_n^2 w^2(\varphi)\left(1 - u_{n_{\perp}}^2\right)} \tag{5.124}$$

As the left side of inequality (5.124) has been shown to be positive, we can square it to get:

$$a_n^2 u_{n_\perp}^2 \leq 1 - a_n^2 w^2(\varphi)\,(1 - u_{n_\perp}^2) \tag{5.125}$$

This leads to the following bound on quantity a_n:

$$a_n \leq \frac{1}{\sqrt{u_{n_\perp}^2 + w^2(\varphi)\,(1 - u_{n_\perp}^2)}} \tag{5.126}$$

According to (5.109), quantity $u_{n_\perp}$ is positive. The multiplication of inequality (5.126) with $u_{n_\perp}$ results in:

$$a_n u_{n_\perp} \leq \frac{u_{n_\perp}}{\sqrt{u_{n_\perp}^2 + w^2(\varphi)\,(1 - u_{n_\perp}^2)}} \tag{5.127}$$

The introduction of equation (5.108) into the left hand of inequality (5.127) leads to:

$$f_\perp \leq \frac{u_{n_\perp}}{\sqrt{u_{n_\perp}^2 + w^2(\varphi)\,(1 - u_{n_\perp}^2)}} \tag{5.128}$$

This establishes the second inequality of the proof.

□

The quantity $u_{n_\perp}$ measures how close to singularity, i.e. ill-conditioned, the system is, with lower values representing worse conditioning. The cases in which the proposed approximation works are the ones where $f_\perp$ is very low. The theorem shows that, if $\varphi < \pi/2$, then:

$$f_\perp \leq u_{n_\perp} \tag{5.129}$$

The condition $\varphi < \pi/2$ represents an obtuseness requirement for the associated vertex. This requirement is not very restrictive. We assume this condition to hold during the rest of this discussion.

5.2.3 Use of the Bounding Box

In order to bound the approximation error, a well-conditioned overapproximating basis within suitable distance of the true optimal basis is needed. Such a basis can be guaranteed to exist by adding bounding box constraints to the linear program.

The fact that each time step of the Coho algorithm starts from a set of projection polygons makes the computation of a bounding box for the feasible region of the linear program trivial.

The feasible region of a face undergoes bloating and intersection with other projectahedra, but these operations have obvious equivalents for the bounding box. Intersection operations might result in a slight overapproximation of the bounding box, but this is deemed acceptable.

The optimal solution of the linear program lies inside the bounding box. Consider a line that is the intersection of $d-1$ of the hyperplanes that define the optimal vertex. The optimal point divides the line into two halflines. The points that belong to one of the halflines are characterized by higher-than-optimal cost, as needed for the type of approximation that we are seeking.

This halfline intersects (at most) half of the bounding box hyperplanes. We shall establish that at least one bounding box hyperplane exists such that its intersection with the halfline is well-conditioned and their intersection point is either on the bounding box or very close to it.

In the arguments that follow, the conditioning of the intersection of the halfline with the hyperplane is measured by the projection of the direction of the halfline onto the normal to the hyperplane. Low values indicate that the hyperplane and the halfline are nearly parallel, so their intersection is ill-conditioned.

Definition 2 *A vector* $\mathbf{f} \in \mathbf{R}^d$ *is* ϵ*-perpendicular to another vector* $\mathbf{g} \in \mathbf{R}^d$ *iff* $\frac{|\mathbf{f} \cdot \mathbf{g}|}{\|\mathbf{f}\| \, \|\mathbf{g}\|} \leq \epsilon$. *We write* $\mathbf{f} \perp_\epsilon \mathbf{g}$.

The above definition gives a quantitative measure for how close two vectors are to

being perpendicular. It is easy to see that $\mathbf{f} \perp_0 \mathbf{g} \Leftrightarrow \mathbf{f} \perp \mathbf{g}$.

Lemma 2 *Let* $\mathbf{f}$ *be a vector in* $\mathbf{R}^d \setminus \{\mathbf{0}\}$.

Let ϵ *be a non-negative real number such that*

$$\epsilon < \frac{1}{\sqrt{d}} \tag{5.130}$$

Then $\mathbf{f}$ *cannot be* ϵ*-perpendicular to all the directions of an orthogonal basis.*

Proof Let $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$ be an orthonormal basis of $\mathbf{R}^d$. Then:

$$\mathbf{f} = \sum_{i=1}^{d} f_i \mathbf{e}_i, \quad \text{where } f_i = \mathbf{f} \cdot \mathbf{e}_i \tag{5.131}$$

Because basis E is orthonormal:

$$f_i = \mathbf{f} \cdot \mathbf{e}_i \tag{5.132}$$

and

$$\|\mathbf{f}\| = \sqrt{\sum_{i=1}^{d} f_i^2} \tag{5.133}$$

Suppose $\mathbf{f} \perp_\epsilon \mathbf{e}_i, \forall i = 1, \ldots, d$. By (5.132), this implies that:

$$f_i < \epsilon \|\mathbf{f}\|, \quad \forall i = 1, \ldots, d \tag{5.134}$$

The introduction of (5.134) into (5.133) leads to:

$$\|\mathbf{f}\| < \sqrt{\sum_{i=1}^{d} (\epsilon \|\mathbf{f}\|)^2} = \epsilon\sqrt{d}\, \|\mathbf{f}\| \tag{5.135}$$

or

$$1 < \epsilon\sqrt{d} \tag{5.136}$$

which contradicts hypothesis (5.130).

□

Lemma 3 *Let $f \in \mathbf{R}^d$ be a unit vector such that $f_i \geq 0,\ \forall i = 1, \ldots, d$.*
Let $b \in \mathbf{R}^d_+$ be a d-dimensional point.
Let $\epsilon \in \mathbf{R}_+$ be a non-negative real such that $\epsilon < \frac{1}{\sqrt{d}}$.
Then $\exists k \in \{1, \ldots, d\}$ and $\exists \lambda \geq 0$ such that:

$$\lambda f_k = b_k \tag{5.137}$$

and

$$f_k > \epsilon \tag{5.138}$$

and

$$\lambda < \frac{\|b\|}{\sqrt{1 - (d-1)\epsilon^2}} \tag{5.139}$$

Proof All the numbers used in the proof are non-negative, either by definition or by being a sum, product, or quotient of non-negative numbers. Non-negativity will not be stated explicitly in the rest of the proof.

Without loss of generality, the unit vectors of the basis can be renumbered such that

$$\frac{b_i}{f_i} \leq \frac{b_{i+1}}{f_{i+1}},\ \forall i = 1, \ldots, d \tag{5.140}$$

Let

$$k = \min_{1 \leq i \leq d} i : \neg(f \perp_\epsilon \mathbf{e}_i) \tag{5.141}$$

By lemma 2, such a k exists.

Let

$$\lambda = \frac{b_k}{f_k} \tag{5.142}$$

By hypothesis:

$$\|f\| = 1 \ \Rightarrow\ \sum_{i=k}^{d} f_i^2 = 1 - \sum_{i=1}^{k-1} f_i^2 \tag{5.143}$$

By the definition of k:

$$\forall i = 1, \ldots, k :\ f_i \perp_\epsilon \mathbf{e}_i \ \Rightarrow\ f_i < \epsilon \tag{5.144}$$

The introduction of (5.144) in (5.143) yields:

$$\sum_{i=k}^{d} f_i^2 > 1 - (k-1)\epsilon^2 \tag{5.145}$$

By the definition of the Euclidean norm:

$$\sum_{i=k}^{d} b_i^2 \leq \|b\|^2 \tag{5.146}$$

Combining (5.145) and (5.146):

$$\frac{\sum_{i=k}^{d} b_i^2}{\sum_{i=k}^{d} f_i^2} < \frac{\|b\|^2}{1-(k-1)\epsilon^2} \tag{5.147}$$

By the definition of k:

$$\frac{b_k}{f_k} \leq \frac{b_i}{f_i} \quad \forall i = k, \ldots, d \tag{5.148}$$

This can be rewritten as:

$$\frac{b_k}{f_k} = \nu_i \frac{b_i}{f_i}, \text{ where } \nu_i \leq 1, \quad \forall i = k, \ldots, d \tag{5.149}$$

The square of the equation above is:

$$\frac{b_k^2}{f_k^2} = \frac{\nu_i^2 b_i^2}{f_i^2}, \quad \forall i = k, \ldots, d \tag{5.150}$$

Summing the numerators and the denominators of the right-hand side of the equation above for $i = k, \ldots, d$ yields:

$$\frac{b_k^2}{f_k^2} = \frac{\sum_{i=k}^{d} \nu_i^2 b_i^2}{\sum_{i=k}^{d} f_i^2} \tag{5.151}$$

The definition of ν_i by (5.149) as a subunitary number leads to:

$$\sum_{i=k}^{d} \nu_i^2 b_i^2 \leq \sum_{i=k}^{d} b_i^2, \quad \forall i = k, \ldots, d \tag{5.152}$$

By combining (5.151) and (5.152) it follows that:

$$\left(\frac{b_k}{f_k}\right)^2 \leq \frac{\sum_{i=k}^{d} b_i^2}{\sum_{i=k}^{d} f_i^2} \tag{5.153}$$

From (5.147) and (5.153) we obtain:

$$\left(\frac{b_k}{f_k}\right)^2 < \frac{\|b\|^2}{1-(k-1)\epsilon^2} \tag{5.154}$$

The introduction of (5.142) in the inequality above yields:

$$\lambda < \frac{\|b\|}{\sqrt{1-(k-1)\epsilon^2}} \tag{5.155}$$

The maximum of the right-hand side of the equation above is achieved for $k = d$:

$$\lambda < \frac{\|b\|}{\sqrt{1-(d-1)\epsilon^2}} \tag{5.156}$$

□

A halfline HL that emanates from the origin in a direction whose components are all positive is contained in the positive orthant of the space. Each point b in $\mathbf{R}^d$ can be seen as the corner of the box whose diagonally opposite corner is at the origin. Each face of the box is included in a hyperplane that is normal to a coordinate axis. If point b is situated in the positive orthant, then halfline HL must intersect at least one of the box hyperplanes that pass through b. Lemma 3 shows that such a hyperplane exists such that its intersection with HL is both well-conditioned and situated not much farther from the origin than the diameter of the box.

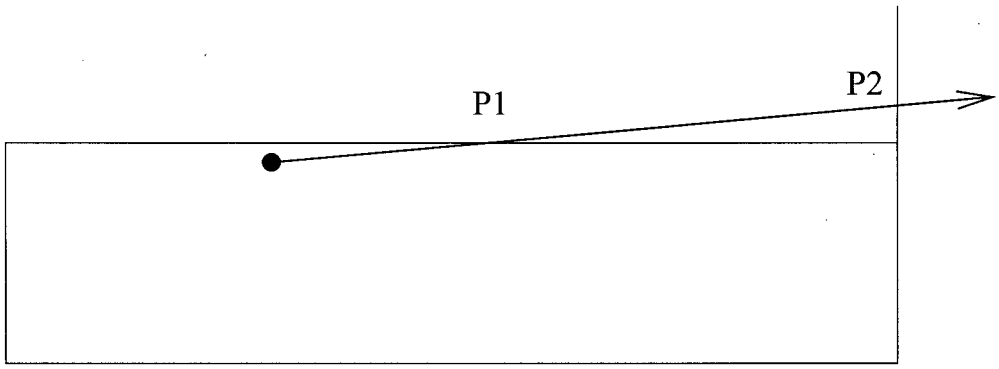

Figure 5.2: Halfline emanating from inside a box: first intersection with a box line is P1, which is ill-conditioned; second intersection is P2, which is well-conditioned and close to the bounding box.

Theorem 2 *Let* HC *be a d-dimensional hypercube of diameter D.*

Let HL *be a halfline emanating from a point* x_0 *inside the hypercube in direction* f*:*

$$\mathrm{HL} = \{x \in \mathbf{R}^d : x = x_0 + \lambda f,\ \lambda \geq 0\} \tag{5.157}$$

Let $\epsilon \in \mathbf{R}_+$ *be a non-negative real such that* $\epsilon < \frac{1}{\sqrt{d}}$.

Then an intersection point x_e *of* HL *with a hyperplane* HP *of the hypercube* HC *exists such that:*

$$\neg(f \perp_\epsilon n_{\mathrm{HP}}) \tag{5.158}$$

where n_{HP} *is the normal to hyperplane* HP*, and:*

$$\|x_e - x_0\| < \frac{D}{\sqrt{1-(d-1)\epsilon^2}} \tag{5.159}$$

Proof Trivial transformations that do not modify distances (translations and reflections) can move x_0 to the origin and f into the positive orthant. Let b the corner of the hypercube now situated in the positive orthant. Lemma 3 can be applied to the current values of f and b to obtain a real number λ with the properties stated by the lemma.

Let:

$$x_e = \lambda f \tag{5.160}$$

Because $x_0 = 0$ and $\lambda \geq 0$, $x_e \in \text{HL}$ by the definition of HL.

Let:

$$\text{HP} = \{x \in \mathbf{R}^d : x_k = b_k\} \tag{5.161}$$

By the definition of x_e:

$$(x_e)_k = \lambda f_k \tag{5.162}$$

By the definition of λ (lemma 3, (5.137)), (5.162) can be rewritten as:

$$(x_e)_k = b_k \tag{5.163}$$

which means that $x_e \in \text{HP}$. The point x_e belongs to both HP and HL, so it is an intersection point of these two sets.

One normal to HP is $n_{\text{HP}} = \mathbf{e}_k$. The angle between n_{HP} and f is characterized by the quantity:

$$\frac{|f \cdot n_{\text{HP}}|}{\|f\| \, \|n_{\text{HP}}\|} = \frac{|f_k|}{1 \cdot 1} = f_k \tag{5.164}$$

The introduction of (5.138) into (5.164) leads to:

$$\frac{|f \cdot n_{\text{HP}}|}{\|f\| \, \|n_{\text{HP}}\|} > \epsilon \tag{5.165}$$

from which (5.158) immediately follows.

As the point x_0 is now at the origin, the distance between x_0 and x_e is:

$$\|x_e - x_0\| = \|x_e\| = \|\lambda f\| = \lambda \, \|f\| = \lambda \tag{5.166}$$

By requirement (5.139) of lemma 3:

$$\lambda < \frac{\|b\|}{\sqrt{1 - (d-1)\epsilon^2}} \tag{5.167}$$

The distance from a point inside a hypercube to a corner of the hypercube cannot be larger than the diagonal of the hypercube. As the origin is contained inside HC and $\|b\|$ is in fact the distance from the origin to corner b of HC, it follows that:

$$\|b\| \leq D \tag{5.168}$$

The combination of equation (5.166) and inequalities (5.167) and (5.168) establishes inequality (5.159).

□

The theorem shows that any halfline emanating from a point inside a bounding box has an intersection with a bounding box hyperplane that is well-conditioned and situated not far from the point (fig. 5.2).

Formula (5.159) expresses the tradeoff that exists between the conditioning of the intersection point x_e of the halfline with the bounding box, measured by ϵ, and the distance from the origin x_0 of the halfline to x_e. Characteristic values of d do not exceed 20. Consider that ϵ is required to be at least 0.1, which guarantees the good conditioning of the intersection. Then theorem 2 guarantees that:

$$\|x_e - x_0\| \leq \gamma D \tag{5.169}$$

where:

$$\gamma = \frac{1}{\sqrt{1-(d-1)\epsilon^2}} = 1.(1) \tag{5.170}$$

The value of ϵ could be set to a much lower value and still represent a well-conditioned intersection while driving γ very close to 1. That, however, wouldn't introduce a qualitative change to the argument.

In the case of Coho LPs, well-conditioned bases that approximate the optimal solution more closely than any basis that contains a bounding box hyperplane may exist. However, it is the use of the bounding box that provides an upper bound on the distance from the true optimal vertex.

5.2.4 Error Bound for Coho Cycles

Having described a way to approximate the cost function for a particular class of ill-conditioned optimal bases, we shall now examine the application of this idea to Coho linear programs. A basis represents a matrix that may contain several independent cycles, each of which can be considered separately.

In order to bound the approximation error, a bound on the projection of an arbitrary basic column onto the normal to all the other basic columns is needed.

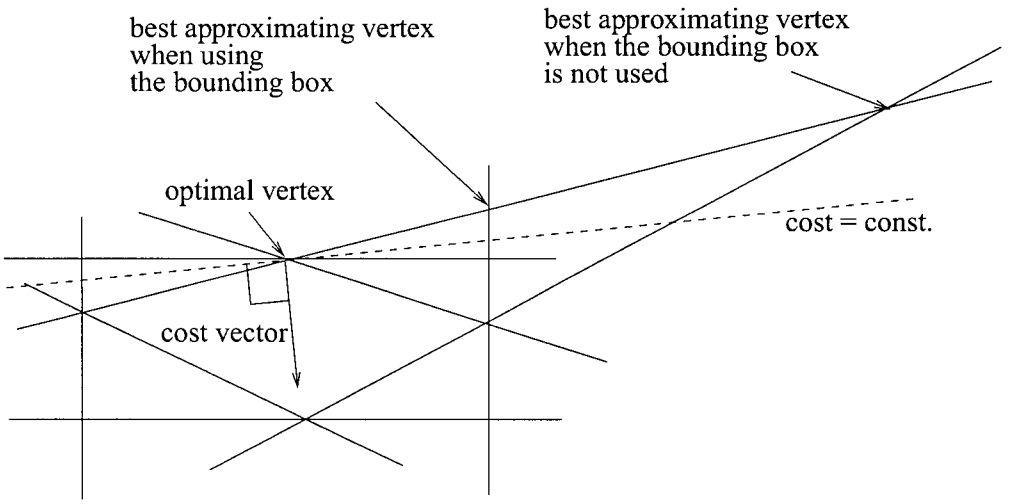

Figure 5.3: The best approximating vertex, with and without the bounding box

First the direction of a normal to face of the basic cost cone is determined:

Lemma 4 *Let A be a matrix representing a cycle in normalized form as described by* (4.4).

Let d be a vector defined as $d = \mathrm{col}(P_0, \ldots, P_{n-1})$, i.e.:

$$d_k = P_{k-1}, \ \forall k = 1, \ldots, n \tag{5.171}$$

Then d is orthogonal to columns $2, \ldots, n$ of matrix A:

$$d \perp A_{:,j}, \quad \forall j = 2, \ldots, n \tag{5.172}$$

Proof By the definition of the dot product:

$$d^T A_{:,j} = \sum_{i=1}^{n} d_i A_{i,j} \tag{5.173}$$

The introduction of the definition of $A_{:,j}$ results in:

$$d^T A_{:,j} = \sum_{i=1}^{j-2} d_i 0 + d_{j-1}(-\alpha_{j-1}) + d_j 1 + \sum_{i=j+1}^{n} d_i 0 = d_j - \alpha_{j-1} d_{j-1} \tag{5.174}$$

From the definition of d it follows that:

$$d^T A_{:,j} = P_{j-1} - \alpha_{j-1} P_{j-2} \tag{5.175}$$

The use of (4.6) leads to the final result:

$$d^T A_{:,j} = 0 \tag{5.176}$$

□

The next step is to determine the projection of a basic column onto the normal to the others:

Theorem 3 *In the conditions of lemma 4:*

$$\|\text{proj}_d \operatorname{dir} A_{:,1}\| = \frac{|P_n - 1|}{\sqrt{1+\alpha_n^2}\sqrt{\sum_{i=0}^{n-1} P_i^2}} \tag{5.177}$$

Proof The columns $A_{:,j}, j = 2, \ldots, n$ are linearly independent. Together with d, which is normal to each $A_{:,j}$, they form a basis for $\mathbf{R}^n$. Consequently it is possible to express $A_{:,1}$ as:

$$A_{:,1} = \sum_{j=2}^{n} c_j A_{:,j} + \delta d \tag{5.178}$$

where $c_j \in \mathbf{R}, \forall j \in \{2, \ldots, n\}$ and $\delta \in \mathbf{R}$.

The first row of equation (5.178) yields:

$$1 = c_2(-\alpha_1) + \delta P_0 \tag{5.179}$$

Considering that $\alpha_1 = P_1$ and $P_0 = 1 = P_0^2$, this can be rewritten as:

$$1 = -c_2 P_1 + \delta P_0^2 \tag{5.180}$$

From rows $2 \ldots n-1$ of equation (5.178) it follows that:

$$0 = c_i + c_{i+1}(-\alpha_i) + \delta P_{i-1}, \quad i = 1, \ldots, n-2 \tag{5.181}$$

Multiplication of the above with P_{i-1} and the equality $\alpha_i P_{i-1} = P_i$ lead to:

$$0 = c_i P_{i-1} - c_{i+1} P_i + \delta P_{i-1}^2, \quad i = 1, \ldots, n-2 \tag{5.182}$$

The memberwise summation of all of the above equations yields:

$$\begin{aligned} 0 &= \sum_{i=2}^{n-1} (c_i P_{i-1} - c_{i+1} P_i + \delta P_{i-1}^2) \\ &= \sum_{i=2}^{n-1} (c_i P_{i-1}) - \sum_{i=2}^{n-1} (c_{i+1} P_i) + \delta \sum_{i=2}^{n-1} P_{i-1}^2 \\ &= \sum_{i=2}^{n-1} (c_i P_{i-1}) - \sum_{i=3}^{n} (c_i P_{i-1}) + \delta \sum_{i=1}^{n-2} P_i^2 \\ &= c_2 P_1 - c_n P_{n-1} + \delta \sum_{i=1}^{n-2} P_i^2 \end{aligned} \tag{5.183}$$

The last row of equation (5.178) can be expressed as:

$$-\alpha_n = c_n + \delta P_{n-1} \tag{5.184}$$

Multiplication with P_{n-1} and the equality $\alpha_n P_{n-1} = P_n$ enable us to rewrite the above :

$$-P_n = c_n P_{n-1} + \delta P_{n-1}^2 \tag{5.185}$$

The memberwise summation of (5.180), (5.183), and (5.185) is:

$$\begin{aligned} 1 + 0 - P_n = &(-c_2 P_1 + \delta P_0^2) \\ &+ (c_2 P_1 - c_n P_{n-1} + \delta \sum_{i=1}^{n-2} P_i^2) \\ &+ (c_n P_{n-1} + \delta P_{n-1}^2) \end{aligned} \tag{5.186}$$

This simplifies to:

$$1 - P_n = \delta \sum_{i=0}^{n-1} P_i^2 \tag{5.187}$$

or, equivalently:

$$\delta = \frac{1 - P_n}{\sum_{i=0}^{n-1} P_i^2} \tag{5.188}$$

The orthogonality of d onto $c_j \in \mathbf{R}, \forall j = 2, \ldots, n$ implies that:

$$\text{proj}_d A_{:,j} = \delta d \tag{5.189}$$

and for the direction of $A_{:,1}$:

$$\text{proj}_d \,\text{dir}\, A_{:,1} = \frac{\text{proj}_d A_{:,1}}{\|A_{:,1}\|} = \frac{\delta d}{\|A_{:,1}\|} \tag{5.190}$$

The substitution of the norms of d and $A_{:,1}$ yields:

$$\|\text{proj}_d \,\text{dir}\, A_{:,1}\| = \frac{|P_n - 1|}{\sum_{i=0}^{n-1} P_i^2} \sqrt{\sum_{i=0}^{n-1} P_i^2} \frac{1}{\sqrt{1+\alpha_n^2}} = \frac{|P_n - 1|}{\sqrt{1+\alpha_n^2}\sqrt{\sum_{i=0}^{n-1} P_i^2}} \tag{5.191}$$

□

A low value of quantity $\|\text{proj}_d \,\text{dir}\, A_{:,1}\|$ means that a bounding-box approximation of the cost achieved by dropping column 1 of matrix A will be accurate.

Equation (5.4) associates a circulant matrix B to matrix A. Equation (5.191) shows that the ill conditioning of matrix B, which is reflected in a low value of $|P_n - 1|$, implies that a pivot to the bounding box will yield a satisfactory approximation of the cost. The following theorem establishes a bound on the error introduced by pivoting to the bounding box in a Coho LP:

Theorem 4 *Let* CLP$(A^{\mathcal{C}}, b^{\mathcal{C}}, c^{\mathcal{C}})$ *be a Coho linear program with n variables and m constraints, where the cost vector,* $c^{\mathcal{C}}$*, is a unit vector.*
Let $\mathcal{B}$ *be an optimal basis of* CLP *and* x *be the optimal vertex of* CLP *corresponding to* $\mathcal{B}$*.*
Let

$$A = \left(A^{\mathcal{C}}{}_{\mathcal{B},:}\right)^T \tag{5.192}$$

Suppose that A *is a cycle in normalized form as per equation* (4.4)*. Moreover, suppose that the maximum angle between* $A_{:,1}$ *and any positive combination of* $A_{:,2}, \ldots, A_{:,n}$ *does not exceed* $\frac{\pi}{2}$*.*

Let D be the diameter of the bounding box of the feasible region of the LP. The inequalities representing the bounding box are supposed to be among the constraints that define the feasible region of the linear program.

Let a Coho boundary hyperplane be defined as:

$$\mathrm{CHP}(k) = \{x \in \mathbf{R}^n : {A^{\mathcal{C}}}_{k,:}\, x = {b^{\mathcal{C}}}_k\} \tag{5.193}$$

Let d_1 be the intersection of boundary hyperplanes $\mathcal{B}(2), \ldots, \mathcal{B}(n)$:

$$d_1 = \bigcap_{k=2}^{n} \mathrm{CHP}(\mathcal{B}(k)) \tag{5.194}$$

Let ϵ be a positive number such that

$$\epsilon < \frac{1}{\sqrt{n}} \tag{5.195}$$

Then there exists a basis $\mathcal{B}'$ with corresponding basic solution x' such that:

$$\mathcal{B}'(k) = \mathcal{B}(k) \quad \forall k = 2, \ldots, n \tag{5.196}$$

and

$$\neg(d_1 \perp_\epsilon A_{\mathcal{B}'(1),:}) \tag{5.197}$$

and

$$|(c^{\mathcal{C}})^T x' - (c^{\mathcal{C}})^T x| \leq \frac{1}{\sqrt{1-(n-1)\epsilon^2}} \frac{|P_n - 1|}{\sqrt{1+\alpha_n^2}\sqrt{\sum_{i=0}^{n-1} P_i^2}} \tag{5.198}$$

where P_i is defined in equation (4.5).

Proof From the definition of CHP, it follows immediately that:

$${A^{\mathcal{C}}}_{k,:} \perp \mathrm{CHP}(k) \quad \forall k = 1, \ldots, m \tag{5.199}$$

The definition of A implies that $A_{:,k} = {A^{\mathcal{C}}}_{\mathcal{B}(k),:}$, which, introduced into (5.199), yields:

$$A_{:,k} \perp \mathrm{CHP}(\mathcal{B}(k)) \quad \forall k = 2, \ldots, n \tag{5.200}$$

Definition (5.194) can be restated as:

$$d_1 \subset \mathrm{CHP}(\mathcal{B}(k)) \quad \forall k = 2, \ldots, n \tag{5.201}$$

The combination of (5.200) and (5.201) yields:

$$d_1 \perp A_{:,k} \quad \forall k = 2, \ldots, n \tag{5.202}$$

Feasibility requires that the cost vector $c^{\mathcal{C}}$ be a positive combination of the columns of matrix A. Each column of matrix A can be turned into a unit vector by scaling it by a positive factor. Therefore $c^{\mathcal{C}}$ is also a positive combination of the unit vectors of the columns of matrix A. Along with (5.202) and with the obtuseness hypothesis, this enables the application of theorem 1 to bound the projection of vector $c^{\mathcal{C}}$ onto the normal d_1 to the hyperplane generated by $\operatorname{dir} A_{:,2}, \ldots, \operatorname{dir} A_{:,n}$ by the projection of vector $\operatorname{dir} A_{:,1}$ onto the same direction:

$$\left\|\mathrm{proj}_{d_1} c^{\mathcal{C}}\right\| \leq \left\|\mathrm{proj}_{d_1} \operatorname{dir} A_{:,1}\right\| \tag{5.203}$$

The right-hand side of the above inequality is determined by theorem 3. The introduction of its result into (5.203) leads to:

$$\left\|\mathrm{proj}_{d_1} c^{\mathcal{C}}\right\| \leq \frac{|P_n - 1|}{\sqrt{1 + \alpha_n^2}\sqrt{\sum_{i=0}^{n-1} P_i^2}} \tag{5.204}$$

Let HL be one of the two halflines that point x determines on line d_1. The fact that the optimal vertex must lie inside the bounding box of the feasible region, along with (5.195), means that the conditions of theorem 2 are met. By hypothesis, the bounding box inequalities are part of the linear program. Let the hyperplane in theorem 2 be:

$$\mathrm{HP} = \mathrm{CHP}(r) \tag{5.205}$$

a normal to which is:

$$n_{\mathrm{HP}} = {A^{\mathcal{C}}}_{r,:} \tag{5.206}$$

Let the intersection point of HP and HL be x'. Then:

$$\neg(d_1 \perp_\epsilon A^{\mathcal{C}}{}_{r,:}) \tag{5.207}$$

and

$$\|x' - x\| \leq \frac{D}{\sqrt{1-(n-1)\epsilon^2}} \tag{5.208}$$

A vertex of a Coho LP represents the intersection of the n boundary hyperplanes that correspond to its basis. Line d_1 represents the intersection of $n-1$ boundary hyperplanes. A vertex of the LP is to be found at the intersection of d_1 with any other boundary hyperplane, the corresponding basis being formed by the hyperplanes that contain d_1 and the hyperplane that d_1 intersects. Vertex x is to be found the intersection of d_1 with CHP($\mathcal{B}(1)$). Vertex x' represents the intersection of d_1 and CHP(r). This is to say that x' is the vertex that corresponds to the basis obtained by replacing $\mathcal{B}(1)$ with r:

$$\mathcal{B}'(k) = \begin{cases} r & \text{if } k = 1 \\ \mathcal{B}(k) & \text{if } k = 2, \ldots, n \end{cases} \tag{5.209}$$

This definition satisfies (5.196) and, introduced in (5.207), yields (5.197).
The difference in cost between vertices x' and x is:

$$|(c^{\mathcal{C}})^T x' - (c^{\mathcal{C}})^T x| = \|x' - x\| \, \|\text{proj}_{x'-x} \, c^{\mathcal{C}}\| \tag{5.210}$$

The fact that $x' \in d_1$ and $x \in d_1$ implies that $(x' - x) \parallel d_1$, so equation (5.210) can be rewritten as:

$$|(c^{\mathcal{C}})^T x' - (c^{\mathcal{C}})^T x| = \|x' - x\| \, \|\text{proj}_{d_1} \, c^{\mathcal{C}}\| \tag{5.211}$$

The introduction of inequalities (5.208) and (5.204) into (5.211) establish result (5.198) of the theorem.

□

Theorem 4 establishes an upper bound on the error introduced into the cost by pivoting to the bounding box of the feasible region. In general, the solution to an n-dimensional linear system can be seen as the intersection of n hyperplanes. If the linear system is non-degenerate, the intersection of $n-1$ of these hyperplanes represents a line. The solution of the system is the intersection of this line with the other hyperplane. Theorem 4 establishes a lower bound on the angle between the intersection line of $n-1$ hyperplanes and the normal to the n^{th} hyperplane as a measure of the conditioning of the system. Good conditioning and a fairly low distance between the optimal and the approximating vertex can be obtained simultaneously. The theorem also shows that the approximation error decreases as the conditioning of the circulant matrix corresponding to the optimal basis worsens, i.e. as $|P_n - 1|$ approaches 0. The approximating solution was required to be a basic solution of the LP in order to enable its discovery by the Simplex algorithm.

Ill-conditioning due to the scaling matrix S does not appear to be curable by pivoting to the bounding box. However, we suspect that obtuse ill-conditioning of the optimal basis should always be nearly orthogonal to the cost vector. This gives us hope for the discovery of better methods resulted from the exploitation of this property.

Theorem 4 refers to the case of a matrix with one cycle and so do the results presented earlier in this chapter. An n-dimensional square matrix can have at most $n/2$ non-trivial cycles. By the triangle inequality, the total error affecting the computed cost is bounded by the sum of the errors for the cycles. This observation, along with the error bounds established for cycles, leads immediately to an error bound for the computed cost of a general, multicycle matrix.

5.2.5 Summary

In this section we have studied the approximation of the optimal cost through a pivot to the bounding box. A class of optimal vertices for which such an approximation

introduces small errors has been identified. Then a bound on the component of the cost vector which is proportional to the approximation error has been determined.

The approximation technique relies on the existence of a well-conditioned basis formed by replacing an optimal constraint with a bounding box constraint. The existence of such a basis has been proven. The approximation error is proportional to the distance from the true optimal vertex to the approximating vertex. A bound on this distance has been established when a bounding box for the feasible region is used.

The case of Coho linear systems has been examined in the final part of the section, with the computation of a bound on the quantity that measures the error introduced in the cost by pivoting to the bounding box. The formula for this quantity has shown that the bounding box approximation method works for one type of ill conditioning, whereas the other type remains an open problem.

Chapter 6

Implementation

The previous chapters described how the special structure of the linear programs arising in Coho can be exploited to produce an efficient and robust version of Simplex. This chapter addresses three remaining issues for a practical implementation. First, the problem of finding an initial feasible basis is addressed. Second, the way the algorithm deals with uncertainty in the results of intermediate computations is presented. Third, the solution of linear programs whose feasible region is an arbitrary linear transformation of an actual projectahedron is described.

6.1 Finding an Initial Invertible Basis

Let SLP(A_S, b_S, c_S) be an instance of a linear program in standard form whose matrix A_S exhibits the Coho-specific structure (either one or two non-zero elements in each column). Let d be the number of rows and f be the number of columns of matrix A_S, where $d < f$. The Simplex algorithm needs a feasible basis $\mathcal{B}_0$ from which to start pivoting. In order for a selection of columns of A_S to represent a feasible basis, it must first represent a non-singular matrix. Due to the sparsity of the matrix A_S of a Coho LP, finding an structurally non-singular column selection is not trivial.

The structure of matrix A_S can be seen as a graph G: each row corresponds

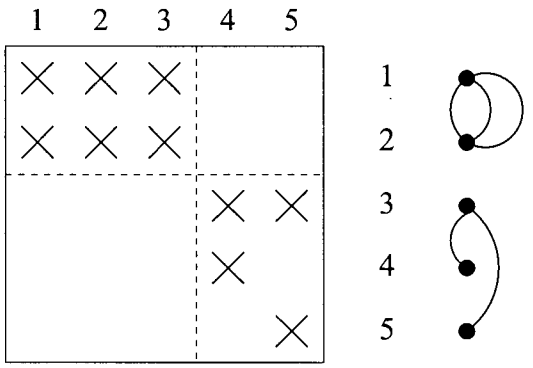

Figure 6.1: Subgraph that corresponds to an structurally singular matrix.

to a vertex, whereas each column turns into an edge. The number of non-zero elements in any column must be either 1 or 2. A column whose non-zeros are in rows i_1 and i_2 represents an edge between vertices i_1 and i_2. A column whose only non-zero element is in row i represents an edge between vertex i and itself. Graph G has d vertices and f edges. Clearly, more that one edge can exist between a pair of vertices.

Let $G_{\mathcal{B}_0}$ be the subgraph of G that corresponds to the submatrix $(A_S)_{:,\mathcal{B}_0}$. Matrix $(A_S)_{:,\mathcal{B}_0}$ contains all the d rows and d of the f columns of A_S, so $G_{\mathcal{B}_0}$ contains all the d vertices and d of the f edges of graph G.

Consider the linear system LS defined by the equation $(A_S)_{:,\mathcal{B}_0}x = y$. Each connected component of $G_{\mathcal{B}_0}$ corresponds to an independent subsystem of LS. The structural non-singularity of matrix $(A_S)_{:,\mathcal{B}_0}$ means that LS must be neither under- nor overdetermined. In turn, this implies that any independent subsystem of LS must have a square left-hand side: one with more rows that columns is overdetermined, whereas one with more columns than rows is underdetermined. Therefore, any connected component G_p of $G_{\mathcal{B}_0}$ must represent a square matrix. By the construction rules for G and $G_{\mathcal{B}_0}$, this means that the graph has equal numbers (d_p) of edges and vertices. The connectedness of G_p requires the use of $d_p - 1$ edges to link the d_p vertices together in a tree structure. The d_p-th edge can join two arbitrary

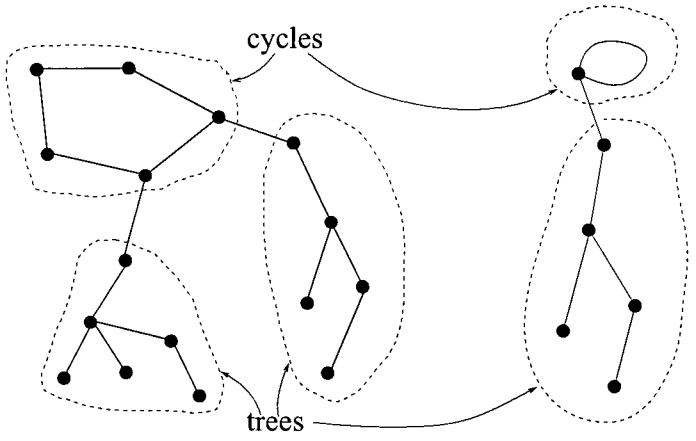

Figure 6.2: Subgraph that corresponds to an invertible matrix.

vertices, thus creating a cycle with some overhanging trees.

This suggests the following way of finding an invertible column selection: a set of trees is constructed by depth-first search such that every vertex is assigned to a tree; then a cycle is introduced in every tree.

Any edge between two different vertices represents a column of matrix A_S. In turn, each column of A_S corresponds to a side of a projection polygon. Moreover, all sides of a projection polygon represent edges between the same pair of vertices of G. So for each edge there will exist at least two more edges between the same vertices. Therefore it is always possible to create a 2×2 cycle in any existing tree by adding one more edge between two vertices that are already connected to one another.

In addition to satisfying the condition described above, each cycle must give rise to a well-conditioned linear system. We assume that all projection polygons are of a low enough degree that it cannot be the case that all vertices of one of them are highly obtuse. This assumption does not exclude any polygons that other components of Coho would handle conveniently: in order for all its vertices to be highly obtuse, a polygon must be at least of degree 10^9, which would render the computational cost of its manipulation prohibitive.

As shown in chapter 5, the cycles embedded in the linear systems that arise in Coho can be affected by two types of ill-conditioning: one is caused by the quantity P_n being close to 1; the other is the result of the bad scaling of the coefficients α_i of the cycle. The following method of picking the initial basis guarantees the avoidance of ill conditioning of the first type: Suppose that the cycle is to be created between vertices i_1 and i_2 between which an edge is known to exist. Vertices i_1 and i_2 define a projection plane and the edges between i_1 and i_2 represent the edges of the corresponding projection polygon. The projection polygons are assumed to have no highly acute vertices. Therefore, no projection polygon encountered in Coho can have the property that every two of its sides are nearly parallel to one another. It follows that it must be possible to pick a pair of edges of the projection polygon corresponding to the pair (i_1, i_2) such that the angle between them is not close to either 0 or π. Then the cycle formed by this pair of edges will of necessity be free from ill conditioning of the first type.

It is also possible to create a cycle in a tree by adding an edge from a node to itself, if one such edge, i.e. a constraint with only one variable, is present. The conditioning is guaranteed to be good in this case.

The method of finding an initial feasible basis described above does not guarantee the absence of ill conditioning caused by the bad scaling of the α_i coefficients of the cycle. Instead, this problem is handled by the branching method described in section 6.3.

6.2 Finding an Initial Feasible Basis

The basis $\mathcal{B}$ identified in the previous section might not have all the properties required by the Simplex algorithm: some components of $T_{:,0} = B^{-1}b$ might be negative, i.e. $\mathcal{B}$ might be infeasible. This calls for another computational step towards a feasible basis.

Classical solutions to the problem do exist, but they involve the introduction

of auxiliary variables that increase the overall computational cost and, worse, destroy the special structure of the LP's matrix A_S. These disadvantages render a solution tailored specifically for the case at hand more desirable. This new solution employs a helper linear program $\text{SLP}^H(A_S^H, b_S^H, c_S^H)$ constructed as follows:

$$
\begin{aligned}
(A_S^H)_{:,j} &= \begin{cases} -(A_S)_{:,j} & \text{if } \mathcal{B}(i) = j \text{ and } T_{i,0} < 0 \\ (A_S)_{:,j} & \text{otherwise} \end{cases} \\
b_S^H &= b_S \\
(c_S^H)_j &= \begin{cases} 1 & \text{if } \mathcal{B}(i) = j \text{ and } T_{i,0} < 0 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

The columns of A_S corresponding to the negative components of $T_{:,0}$ appear negated in A_S^H, thus ensuring that $\mathcal{B}$ is a feasible basis for SLP^H. The same construction ensures that the distribution of the non-zero elements stays the same in A_S^H as in A_S. SLP^H being completely similar to SLP, the same computational techniques developed for SLP can be applied to SLP^H.

A variable that appears negated in the helper LP is called undesirable. Our objective is to obtain a feasible basis that contains no undesirable variable.

The cost function of the helper LP makes the undesirable variables expensive, whereas all the other variables have 0 cost. In order for a pivot to be favorable, it must drive one of the undesirable variables out of the basis.

Given a linear program and a feasible basis for it, the basis remains feasible if arbitrary changes are made to the non-basic columns and to the cost function. This enables us to flip the sign of the undesirable variable that has left the basis while keeping the basis feasible. A variable ceases to be undesirable when its sign is flipped. The cost of the variable is made 0, as the variable doesn't need to be kept out of the basis anymore.

Eventually all the undesirable variables are driven out of the basis or the original LP was infeasible. The optimal basis $\mathcal{B}^H$ that we end up with contains only

variables that appear with the same sign in the original linear program SLP. As the right-hand sides b_S and b_S^H are also identical, $\mathcal{B}^H$ must be a feasible basis for SLP as well. The problem of finding an initial feasible basis for SLP(A_S, b_S, c_S) is thus solved.

6.3 Dealing with Uncertainty and Avoidance of Cycling

Earlier on in this thesis as well as in most textbook presentations, the description of the Simplex algorithm assumes that the numeric operations executed on various floating-point numbers are free of errors. Unfortunately, on real machines, this is not the case. Unavoidable rounding errors introduce an uncertainty with which any floating-point number is known. Floating-point numbers in computations can be thought of more accurately as real intervals: $x \pm e$.

Errors in the result of a computation can diminish the usefulness of the result, occasionally rendering it worthless. The problem can be even more complicated when a decision in the program has to be made based on the computed value of some floating-point number. Any comparison between floating-point numbers can be reduced to the comparison between their difference and 0:

$$x \circ y \Leftrightarrow (x - y) \circ 0, \ \forall \circ \in \{<, \leq, >, \geq, =, \neq\}$$

so the case of comparisons of computed floating-point numbers with 0 will be considered henceforth. The result of the comparison $x \pm e \circ 0$ is undetermined if $e > |x|$. This is tantamount to saying that the sign of x is uncertain.

Comparisons between computed floating-point numbers are used in Simplex in order to decide whether a column shall enter the basis and which column shall leave the basis. In the presence of ill-conditioning, computed quantities occasionally have uncertain signs. Clearly, deciding to take the wrong branch can make the algorithm fail:

- If the wrong column is evicted from the basis, an infeasible basis is reached. Upon checking the sign of the new basic variables, the algorithm signals failure.
- If the wrong column enters the basis, a (slightly) more costly basis is reached. The algorithm may end up caught in a cycle of bases: at some of them a correct pivot is taken, at at least one other base the pivot is wrong, the overall outcome being cycling of a type different that the one treated by Bland's anticycling algorithm.

  Bland's anticycling algorithm essentially provides ways of dealing with the fact that two or more computed quantities are equal. With floating-point computer arithmetic, two computed quantities are very seldom equal, even in cases where error-free computation would have lead to equal results.

An obvious solution to the problems raised by uncertainty in the results of comparisons is to try both possible paths of the computation. The arrival at an infeasible basis or an increase in the cost mean that the current path of computation is in fact wrong and must be abandoned. Clearly this could potentially lead to an explosion in the running time of the algorithm, as each of the n comparisons that the algorithm would effect on an error-free machine can in principle turn into a node of a computation tree with 2^n nodes.

In practice, however, the number of uncertain comparisons is expected to be low. When selecting a column to enter the basis, a clearly favorable column shall always be chosen even if possibly but unclearly favorable columns do exist. Similarly, if a clearly favorable column is found but the identity of the column that must leave the basis is uncertain, another clearly favorable column can be given preference if the column to be evicted is clear in its case. Overall, exploring multiple branches is likely to be necessary only in the neighborhood of an ill-conditioned optimal basis.

A record of the visited bases is maintained such that various paths of the computations are not explored more than once. Clearly, this also solves the problem of cycling.

When the optimal basis of an LP is ill-conditioned, it might not be possible to label it as "clearly optimal", but only as "possibly optimal" instead. If the ill-conditioning of the optimal basis is particularly strong, the optimal basis may seem to be numerically singular: the error bound on some of the basic variables simply becomes infinite. Moreover, bases whose cost is close to but different from the optimal one, may not appear to be suboptimal or infeasible, but "possibly optimal" instead.

The algorithm terminates the search for an optimal basis when a clearly optimal basis is detected or when the exploration of the paths arising from comparisons with unclear result is finished. In order to compute a good estimate of the optimal cost, the algorithm must visit the optimal basis or, if this one is ill-conditioned, a standard-suboptimal well-conditioned basis whose cost is very close to the optimum.

The algorithm guarantees that, for any visited feasible basis, a neighboring basis of lower cost will be visited if one exists. This holds true even if the true sign of the cost difference between the bases is not clear from the computation. Applied recursively, this invariant leads to the fact that the algorithm is guaranteed to visit the optimal basis, which has no feasible and less costly neighbor.

Each type of basis provides the following information about the primal LP in standard form:

- A feasible, clearly standard-suboptimal basis provides an upper bound on the cost.
- An infeasible basis that yields a feasible solution to the dual provides a lower bound on the cost.
- A clearly optimal basis provides both an upper and a lower bound on the cost.
- A maximal set of potentially optimal bases provides an upper and a lower bound on the cost. The cost of each basis in the set can be computed as an

interval. The ends of the union of all these intervals represent bounds on the true optimal cost.

- A numerically singular basis $\mathcal{B}$ that has been reached by taking a clearly favorable pivot from another basis $\mathcal{B}_{\text{pred}}$ must have a lower cost than the basis that preceded it ($\mathcal{B}_{\text{pred}}$).

An upper (and sometimes also a lower) bound on the optimal cost can be computed by examining the record of visited bases. The upper bound and the corresponding solution is the information that Coho expects. The accuracy of the bound depends directly on the closeness to optimality of the visited bases and on the error bounds on the solutions that correspond to these bases.

Clearly a truly optimal basis is the best result of the linear program as regards cost. When the optimal basis is ill-conditioned and consequently produces a solution with large (possibly infinite) error bounds, it is necessary to replace it with the feasible suboptimal well-conditioned basis that has the lowest cost. This slightly suboptimal solution to the standard LP under consideration represents a slightly infeasible solution to its Coho dual. So the true feasible region of the Coho LP is over approximated as required.

Ill conditioning can affect the optimal basis of the helper LP used for determining an initial feasible basis for the actual LP. In this case the optimal basis rather than the optimal cost is of interest. Ill conditioning translates into the discovery of more than one possibly optimal basis rather than one clearly optimal one. The solution to this type of uncertainty is to try out all the possibly optimal bases of the helper LP as initial feasible bases of the actual LP.

6.4 Conserving Structure after Moving Forward in Time

Consider a Coho projectahedron described by the following equation at the beginning of the time step:

$$Ax_0 \geq b \tag{6.1}$$

where $A \in \mathbf{R}^{d \times d}$, $b \in \mathbf{R}^d$ and x_0 represents a feasible point at the beginning of the time step.

The linearized model for the time step has the form:

$$\dot{x} = Mx + q \tag{6.2}$$

where $M \in \mathbf{R}^{d \times d}$, $q \in \mathbf{R}^d$. The duration of the time step is Δt. Let x_e be the position position at the end of the time step of a point whose position at the beginning of the time step was x_0. By integrating (6.2) for the time step we obtain:

$$x_e = e^{M\Delta t}x_0 + (e^{M\Delta t} - I)M^{-1}q \tag{6.3}$$

The combination of (6.1) and (6.3) yields the equation of the region resulted from moving the projectahedron forward in time:

$$AEx_e \geq b_e \tag{6.4}$$

where $E = e^{-M\Delta t}$ and $b_e = b + A(I - e^{-M\Delta t})M^{-1}q$. Matrix E results from matrix exponentiation and is therefore non-singular.

At the end of the time step, the region described by $AEx_e \geq b_e$ needs to be projected back onto various projection planes. This amounts to solving several Coho linear programs of the form

$$\mathrm{CLP}(AE, b_e, c) \tag{6.5}$$

where c is some cost function.

In general, the postmultiplication of A by E yields a matrix that no longer presents the two non-zeros per row structure that describes a projectahedron. Clearly,

the machinery developed for linear programs whose feasible regions are projectahedra cannot be applied directly to linear programs of the type described by (6.5). However, the following transformation enables the reduction of the latter type to the former: in the original problem:

$$\min_{AEx \geq b} c^T x \tag{6.6}$$

the following change of variable is effected:

$$y = Ex \Leftrightarrow x = E^{-1}y \tag{6.7}$$

which yields:

$$\min_{Ay \geq b} d^T y \tag{6.8}$$

where $d = E^{-1}c$ is the transformed cost function. As matrix A is the left-hand side of the inequality that describes a projectahedron, this transformed problem can be solved using the techniques presented earlier.

Let y_{opt} be the optimal solution to the transformed problem. The optimal solution to the original problem is:

$$x_{\text{opt}} = E^{-1}y_{\text{opt}} \tag{6.9}$$

The multiplication of the cost vector and of the optimal solution by the quantity $E^{-1} = e^{M\Delta t}$ are the only operations that need to be added to the Coho LP solver in order to enable it to deal with the kind of LPs described by (6.5).

Chapter 7

Conclusions

7.1 What has been Accomplished

The verification performed by Coho makes heavy use of linear programming. The applicability of the tool depends critically on the accuracy of the solutions to the linear programs that it has to solve.

Linear programming has a well-known mathematical solution that is the Simplex algorithm. However, numerical errors that are inherent to floating point computation sometimes make this algorithm produce unacceptably imprecise results. This thesis studied ways of exploiting the special structure of the linear programs that arise in Coho in order to compute better solutions to them.

A Coho linear program cannot be solved by Simplex directly. The standard solution of introducing additional variables in order to bring it to standard form would have destroyed its special structure. This has been circumvented by observing that the dual of a Coho linear program is a linear program in standard form that exhibits the same special structure and to which Simplex can be applied directly. Moreover, the solution of the Coho LP is straightforward to compute from that of its dual.

As part of solving Coho linear programs, it is necessary to compute Simplex tableau columns. The computation of a tableau column implies solving a linear

system. The structure of such linear systems is closely related to that of the linear programs from which they arise. Because of their structure it was possible to devise a linear-time algorithm for solving Coho linear systems. Based on the linear-time solver, Simplex was modified such as to prevent the propagation of numerical errors between steps.

The main numerical problem that affects the Simplex algorithm as modified for Coho is the errors that affect the solutions to linear systems. Therefore research effort has been directed towards computing more accurate solutions to these systems.

A characterization of the numerical stability of the Coho linear systems has been obtained. Whereas in quantitative terms it is not complete, it does shed light on the possible sources of ill-conditioning. A possible way of determining a forward error bound on the solution to a Coho linear system has been presented.

In the case of most of the LPs encountered in Coho, their optimal cost, rather than their optimal solution, is of interest. The research has identified a class of linear programs for which the computation of the optimal cost can be achieved with a small error, although the optimal vertex is replaced with the vertex obtained by pivoting from the optimal basis to the bounding box of the feasible region. This class has been shown to contain a part of the Coho linear programs.

The error bounds that are computed on the results of floating-point operations permit the linear program solver to identify the situations where the use of a computed value could potentially lead to an incorrect computation path. In such situations, the solver tries out both possible computation paths if the first one fails. This strategy prevents the solver from failing by reaching an infeasible basis and guarantees that the optimal basis of the linear program to solve is eventually visited.

An implementation of Simplex that incorporates the modifications proposed in the thesis was written in Java and integrated within the Coho verification tool. The implementation raises several non-trivial issues that have been presented in

detail in the body of the thesis.

In conclusion, this research has resulted in progress towards better solutions to Coho linear systems and, by way of consequence, to Coho linear programs. Consequently, this is expected to increase the usability of Coho. However, some important questions have remained open.

7.2 Suggestions for Further Research

The study of the Coho linear programs is not finished yet. As mentioned in the body of the thesis, some further explorations are necessary.

The fact that the optimal cost is computed in two ways which yield errors that vary in opposite directions with the conditioning of the optimal basis suggests that it should be possible to determine an error bound on the optimal cost of a Coho linear program that depends only on the unit roundoff and on the dimension of the space.

In order to get a tighter bound on the error in the optimal solution, a better estimate of the condition number of cyclic matrices is needed. Experimental evidence suggests that the available estimate overapproximates the true condition number by several orders of magnitude when the ill conditioning is high.

Moreover, there are linear systems for which the condition number is an overly conservative error predictor [CF88]. The identification of such a situation requires knowledge of the singular value decomposition (SVD) of the system's matrix. General methods for SVD computation are not linear time. A linear-time or nearly linear-time SVD method tailored for Coho cyclic matrices is a desirable first step towards more precise error prediction.

If highly acute ill-conditioned vertices can be shown to occur, a method for dealing with them is certainly needed. Rescaling the system appears to be a promising approach. This is a topic for future research.

When the optimal basis $\mathcal{B}$ of a linear program represents a highly obtuse and

ill-conditioned vertex, a slightly suboptimal basis $\mathcal{B}'$ is guaranteed to exist if the bounding box of the feasible region is made part of the linear program. Unfortunately, a theoretic proof that $\mathcal{B}'$ is reached during the search for optimality is yet to be found. The experimental evidence currently available is inconclusive.

Alternatively, the algorithm could be modified to ensure the visitation of basis $\mathcal{B}'$. The slightly suboptimal basis $\mathcal{B}'$ becomes the optimal basis if the cost vector is tilted slightly in an appropriate way. Such a slight modification to the cost function would guarantee that basis $\mathcal{B}'$ is discovered as the optimal basis.

Running error analysis is presently used for all the Simplex operations that involve real numbers. In the absence of ill-conditioning, this is wasteful. A conservative estimate of the condition number of a left-hand side of a Coho linear system can be computed in linear time. The ability to recognize cases where error analysis is unnecessary based on the condition number estimate can improve the running time of a program by a constant factor.

The use of Givens rotations [Hig96, p.371] for the solving of the Coho linear systems has been suggested by scientific computations experts [Hab01]. The problem seems to be that Givens rotations work best when various rotations are independent, which is not true in our case.

Another suggestion was the replacement of Simplex with the *interior point method* as the algorithm used for solving linear programs. This avenue is entirely unexplored.

Bibliography

[AL94] Martín Abadi and Leslie Lamport. An old-fashioned recipe for real time. *ACM Transactions on Programming Languages and Systems*, 16(5):1543–1571, September 1994.

[Bla77] R. Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2:103–107, 1977.

[CF88] Tony F. Chan and David E. Foulser. Effectively well-conditioned linear systems. *SIAM Journal on Scientific and Statistical Computations*, 9(6):963–969, November 1988.

[GM98] Mark R. Greenstreet and Ian Mitchell. Integrating projections. In Thomas A. Henzinger and Shankar Sastry, editors, *Proceedings of the First International Workshop on Hybrid Systems: Computation and Control*, pages 159–174, Berkeley, California, April 1998.

[GM99] Mark R. Greenstreet and Ian Mitchell. Reachability analysis using polygonal projections. In *Proceedings of the Second International Workshop on Hybrid Systems: Computation and Control*, pages 103–116, Berg en Dal, The Netherlands, March 1999. Springer. LNCS 1569.

[Hab01] Eldad Haber. Personal communication, 2001.

[Hig96] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, 1996.

[Lin92] Elliot Linzer. On the stability of transform-based circular deconvolution. *SIAM Journal on Numerical Analysis*, 29(5):1482–1492, October 1992.

[Mul97] Jean-Michel Muller. *Elementary Functions: Algorithms and Implementation.* Birkhauser, Boston, Basel, Berlin, 1997.

[PS82] Christos H. Papadimitriou and Keneth Steiglitz. *Combinatorial Optimization.* Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1982.

[TPS98] Claire Tomlin, George Pappas, and Shankar Sastry. Conflict resolution in air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.

Appendix A

Definitions and Notations

A.1 Notations

The notation for matrix element selection follows the Matlab style, with the selector appearing as a subscript rather than as an argument.

matrix element

$A_{i,j}$: the element of matrix A at the intersection of column i and row j

matrix row

$A_{i,:}$: row i of matrix A

matrix column

$A_{:,j}$: column j of matrix A

set of matrix columns

$A_{:,J}$, where $J = \{j_1, \ldots, j_C\}$ is a set of indices of columns of A : a matrix S with C columns such that $S_{:,c} = A_{:,j_c}$, $\forall c = 1, \ldots, C$

element of row or column

b_k : element k of the row or column b

set of column elements

b_J, where $J = \{j_1, \ldots, j_C\}$ is a set of indices of elements of b : a column matrix S with C elements such that $S_c = b_{j_c}$, $\forall c = 1, \ldots, C$

vertical matrix join

$[A|B]$: a matrix obtained by appending each row of matrix B to the same-index row of A; A and B must have the same number of rows

identity matrix

$\mathcal{I}_n$: the $n \times n$ identity matrix; n is omitted when it results from the context

A.2 Definitions

linear subspace

A linear subspace S of $\mathbf{R}^d$ is a subset of $\mathbf{R}^d$ closed under vector addition and scalar multiplication.

affine subspace

An affine subspace A of $\mathbf{R}^d$ is a linear subspace S translated by a vector u:

$$A = \{u + x : x \in S\}$$

orthogonal complement of a subspace

The orthogonal complement of a subspace $S \subseteq \mathbf{R}^n$ is defined by:

$$S^{\perp} = \{y \in \mathbf{R}^n : y^T x = 0 \text{ for all } x \in S\}$$

orthogonality of a vector to a subspace

A vector v is orthogonal to a linear subspace W if v is orthogonal to every vector in W.

projection of a vector onto a subspace

Let W be a subspace of $\mathbf{R}^d$. Let $\{u_1, \ldots, u_h\}$ be an orthonormal basis for W.

If v is a vector in $\mathbf{R}^d$, the projection of v onto W is denoted $\text{proj}_W v$ and is defined by

$$\text{proj}_W v = \sum_{i=1}^{h} (v \cdot u_i) u_i$$

distance from a point to a subspace

The distance from a point x to a subspace W is defined as:

$$d(x, W) = \min_{y \in W} d(x, y)$$

An important property is that:

$$d(x, W) = \|x - \text{proj}_W x\|$$

positive combination

Given p vectors $x_1, \ldots, x_p \in \mathbf{R}^d$, a positive combination of them is a vector $x \in \mathbf{R}^d$ of the form:

$$x = \sum_{i=1}^{p} \lambda_i x_i \qquad \lambda_i \geq 0$$

cone

The cone generated by a set of vectors $x_1, \ldots, x_p \in \mathbf{R}^d$ is the set of all their positive combinations.

hyperplane

A hyperplane in $\mathbf{R}^d$ is a set of points defined by:

$$\text{HP}(a, b) = \{x \in \mathbf{R}^d : a^T x = b\} \qquad a \in \mathbf{R}^d \setminus \{\mathbf{0}\},\ b \in \mathbf{R}$$

A hyperplane is an affine subspace of $\mathbf{R}^d$ of dimension $d - 1$.

hyperplane normal

Any vector λa, where $\lambda \in \mathbf{R} \setminus \{0\}$, is a normal to the hyperplane $\text{HP}(a, b)$.

halfspace

A hyperplane $\mathrm{HP}(a, b)$ in $\mathbf{R}^d$ represents the common boundary of two closed halfspaces, which are defined as:

$$\mathrm{HS}_{\geq}(a, b) = \{x \in \mathbf{R}^d : a^T x \geq b\}$$

$$\mathrm{HS}_{\leq}(a, b) = \{x \in \mathbf{R}^d : a^T x \leq b\}$$

halfspace normal

A halfspace normal is a normal to the boundary of a halfspace.

inward halfspace normal

A inward halfspace normal n of a halfspace HS is a halfspace normal such that

$$\forall x \in \mathrm{HS},\ \forall \lambda \in [0, \infty) \qquad x + \lambda n \in \mathrm{HS}$$

Intuitively, a inward normal points from the boundary of the halfspace towards its interior.

The vector a is a inward normal for the halfplane $\mathrm{HS}_{\geq}(a, b)$.

The vector $-a$ is a inward normal for the halfplane $\mathrm{HS}_{\leq}(a, b)$.

closed convex polyhedron

A closed convex polyhedron is the intersection of a finite number of halfspaces:

$$\mathrm{PH} = \bigcap_{i=1}^{f} \mathrm{HS}_{\geq}(a_i, b_i),\ \text{where } a_i \in \mathbf{R}^d \setminus \{\mathbf{0}\} \text{ and } b_i \in \mathbf{R}$$

In matrix form:

$$\mathrm{PH}(A, b) = \{x \in \mathbf{R}^d : Ax \geq b\}$$

where $A \in \mathbf{R}^{f \times d}$ and $b \in \mathbf{R}^d$.

Each hyperplane may correspond to a face of the polyhedron, although some of them might be redundant.

The polyhedron may be empty or unbounded in some directions.

hypercube

A hypercube in $\mathbf{R}^d$ is the Cartesian product of d closed real intervals.

standard basis

The standard basis of the $\mathbf{R}^d$ vector space is the basis

$$\{\mathbf{e}_i : i = 1, \ldots, d\}$$

where

$$(\mathbf{e}_i)_j = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases}$$

vector norm

The norm of a vector v in $\mathbf{R}^d$ is defined as:

$$\|v\| = \sqrt{\sum_{i=1}^{d} v_i^2}$$

unit vector

A unit vector is a vector whose norm is 1.

The unit vector, i.e. the direction, of a vector v is defined by:

$$\operatorname{dir} v = \frac{v}{\|v\|}$$

projection of a vector onto another vector

The projection of a vector v onto another vector u is defined by:

$$\operatorname{proj}_u v = \frac{v \cdot u}{u \cdot u} u = (v \cdot \operatorname{dir} u) \operatorname{dir} u$$

vector angle

The angle between two vectors u and v is defined by:

$$\measuredangle(u, v) = \arccos \frac{u \cdot v}{\|u\| \, \|v\|} = \arccos(\operatorname{dir} u \cdot \operatorname{dir} v)$$

distance between points

The distance between two points x and y in $\mathbf{R}^d$ is:

$$d(x, y) = \|x - y\|$$