Geometric Optimization

Equinoctial School on Geometric Computing

ETH Zürich, 15. – 26. September 1997

Bernd Gärtner

Contents

Chapter 1

Introduction

1.1 Geometric Optimization

In all generality, a geometric optimization problem consists in finding a geometric object of some type which is optimal according to some criterion, among all objects of this type that satisfy a certain geometric condition. Let's consider some examples.

Smallest enclosing ball. Given n points in $\mathbb{R}^d$, find the ball of smallest volume that contains all the points (Figure 1.1).

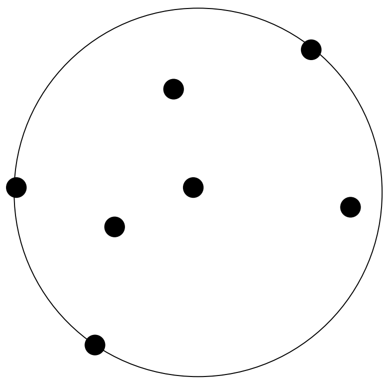


Figure 1.1: Smallest enclosing ball of point set

Smallest enclosing ellipsoid. Given n points in $\mathbb{R}^d$, find the ellipsoid of smallest volume that contains all the points (Figure 1.2).

Distance of polytopes. Given two polytopes P_1, P_2 in $\mathbb{R}^d$, defined by n_1 resp. n_2 points, find the shortest line segment connecting P_1 and P_2 (Figure 1.3).

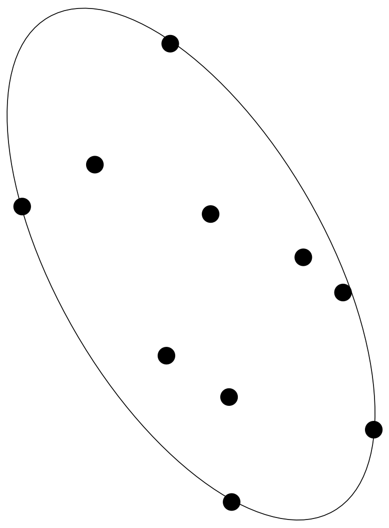


Figure 1.2: Smallest enclosing ellipse of point set

Smallest enclosing annulus. Given n points in the plane, find the annulus of smallest area (or width) that contains all the points (Figure 1.4).

Largest ball in polyhedron. Given a polyhedron defined by n halfspaces in $\mathbb{R}^d$, find the largest ball contained in the polyhedron (Figure 1.5).

All these problems have the property in common that they are defined by n objects in some dimension d (where $d = 2$ in case of the smallest enclosing annulus problem). The scenario in which these problems are typically studied in computational geometry is that n is large while d is small or even constant. After all, computational geometry finds its applications in computer graphics, CAD etc. where the dimension is usually two or three.

In this scenario, the above mentioned problems have something more in common, namely that the optimal solution is defined by only few of the input objects – most of them are redundant. To make it more formal, the optimal object is defined by a set of input objects whose size is only a function of d. Let's check this for one specific example, namely the smallest enclosing ellipsoid, for the case $d = 2$.

It is quite obvious that points which do not lie on the optimal ellipse can be removed from the point set without changing the optimal ellipse. This means, the optimal ellipse is already determined by the points that lie on it. Still, this might be all (or nearly all) points if we have bad luck. But recall from the talk of Jürgen Richter-Gebert that five points already determine a unique conic. This means, if more than five points lie on the ellipse, we can remove all but five, and still the ellipse will not change, because there is only one ellipse through the five points. This means, the smallest enclosing ellipse of n points is already determined by at most five points. If we only knew them, then the problem would be easy, but of course, the main problem is to find this subset of the n points that

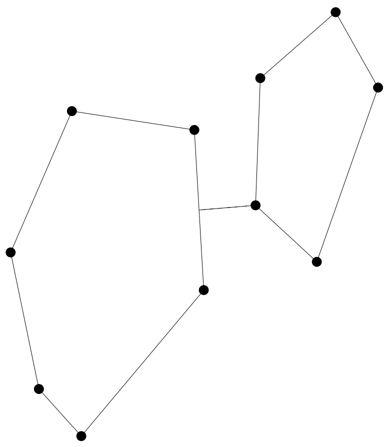


Figure 1.3: Distance of polytopes

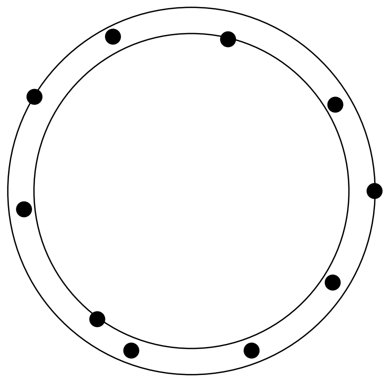


Figure 1.4: Smallest enclosing annulus of point sets

determine the solution.

In case of the other examples we have given above, it is also easy to determine the number of input objects which are necessary to define the optimal solution, and you can check that this number is indeed only a function of d.

It turns out that quite a number of geometric optimization problems can be solved by formulating them as problems of optimizing some *objective function* over a polyhedron, and if the function is reasonably well-behaved, existing and well-studied algorithms for solving such problems can be applied. For example, the *Largest Ball in Polyhedron* problem is of this type: we want to find a point in the polyhedron such that its distance to the boundary is maximized. In this case, the function to optimize (in fact, maximize) assigns to every point in the polyhedron the distance to the boundary.

We can also formulate the problem in another way, obtaining a simpler objective func-

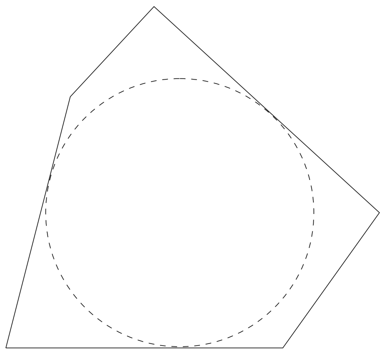


Figure 1.5: Largest ball in polyhedron

tion: find the maximum value of r such that there exists a point in the polyhedron which has distance at least r to the boundary. To write out this problem formally, let's assume the polyhedron is given as the intersection of halfspaces

$$\{x \mid a_j^T x \leq b_j\},$$

$j = 1, \dots n$. The vectors a_j are normal vectors of the hyperplanes that delimit the halfspaces. Moreover, after scaling a_j, b_j accordingly, we can assume the a_j are unit vectors. Now consider a point $\tilde{x}$ in the polyhedron, i.e. $\tilde{x}$ satisfies $a_j^T \tilde{x} \leq b_j$ for all j. It is well-known that the distance of $\tilde{x}$ to the hyperplane $a_j^T x = b_j$ is given by $b_j - a_j^T \tilde{x}$. Then the *Largest Ball in Polyhedron* problem can be written in the form

$$\begin{array}{lll} \text{(LBIP)} & \text{maximize} & r \\ & \text{subject to} & b_j - a_j^T x \geq r. \end{array} \tag{1.1}$$

This is now a problem of maximizing a linear function in $d+1$ variables x, r over a polyhedron in $I\!R^{d+1}$, defined by the halfspaces

$$\{x \mid a_j^T x + r \leq b_i\}.$$

Such problems are known as *Linear Programming problems*, and a particularly nice and efficient method exists for solving them, namely the *Simplex Method.*

Let's look at another example, the *Distance of polytopes.* Assume the polytopes P_1, P_2 are specified as the convex hulls of n_1 points $\{p_1, \dots p_{n_1}\}$ resp. n_2 points $\{q_1, \dots, q_{n_2}\}$. Then we can formulate the problem as follows.

$$\begin{array}{lll} \text{(DOP)} & \text{minimize} & \|p - q\| \\ & \text{subject to} & p = \sum_{i=1}^{n_1} x_i p_i, \\ & & q = \sum_{j=1}^{n_2} y_j q_j, \\ & & \sum_{i=1}^{n_1} x_i = 1, \\ & & \sum_{j=1}^{n_2} y_j = 1, \\ & & x_i, y_j \geq 0, \quad \forall i, j. \end{array}$$

The constraints of this problem just encode the requirement that p is in the convex hull of the $\{p_i\}$, and q is in the convex hull of the $\{q_j\}$. The problem has $2d + n_1 + n_2$ variables, and the subset of $\mathbb{R}^{2d+n_1+n_2}$ defined by the (in)equalities of the problem does again form a polyhedron. The objective function can – without changing the optimal points p and q – be replaced by

$$\|p - q\|^2 = (p - q)^T(p - q),$$

and this is a quadratic function. Problems of this type are called *Quadratic Programming problems*, and an adaptation of the simplex method exists that can also solve these problems.

As it turns out, the *Smallest enclosing annulus* problem (area version) can also be formulated as a linear programming problem (exercise), while the *Smallest enclosing ball* problem is a quadratic programming problem again, which we will only be able to prove at the end of this manuscript.

The *Smallest enclosing ellipse* problem does not quite fit in here. It is neither an instance of linear nor of quadratic programming. This also explains the fact that in dimensions higher than two, this problem is not solvable as easily as our other example problems are.

In the following chapters we will introduce the by now classical simplex method for linear programming which has just celebrated the 50th anniversary of its invention by George Dantzig back in 1947. We will then describe the above mentioned adaptation of it to quadratic programming. This adaptation relies on the important *Kuhn-Tucker* optimality conditions for minimzing convex functions over a polyhedron. We will prove these conditions here.

Finally, we go back to our example problems and briefly discuss how the methods we have developed so far can efficiently be applied to them.

1.2 Linear Programming

Linear Programming (LP) in a quite general form is the problem of minimizing a linear function in n variables subject to m linear inequalities. If, in addition, we require all variables to be nonnegative, we have an LP in *standard form* which can be written as follows.

$$\begin{array}{lll} \text{(LP)} & \text{minimize} & \sum_{j=1}^n c_j x_j \\ & \text{subject to} & \sum_{j=1}^n a_{ij} x_j \leq b_i \quad (i = 1, \ldots, m), \\ & & x_j \geq 0 \quad\quad\quad\quad (j = 1, \ldots, n), \end{array} \tag{1.2}$$

where the c_j, b_i and a_{ij} are real numbers. By defining

$$\begin{aligned} x &:= (x_1, \ldots, x_n)^T, \\ c &:= (c_1, \ldots, c_n)^T, \end{aligned}$$

$$
\begin{aligned}
b &:= (b_1, \ldots, b_m)^T, \\
A &:= \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix},
\end{aligned}
$$

(1.2) can be written in more compact form as

$$
\begin{array}{lll}
\text{(LP)} & \text{minimize} & c^T x \\
& \text{subject to} & Ax \le b, \\
& & x \ge 0,
\end{array}
\qquad (1.3)
$$

where the relations $\le$ and $\ge$ hold for vectors of the same length if and only if they hold componentwise.

The vector c is called the *cost vector* of the LP, and the linear function $z : x \mapsto c^T x$ is called the *objective function.* The vector b is referred to as the *right-hand side* of the LP. The inequalities $\sum_{j=1}^n a_{ij} x_j \le b_i$, for $i = 1, \ldots, m$ and $x_j \ge 0$, for $j = 1, \ldots, n$ are the *constraints* of the linear program. (Due to their special nature, the constraints $x_j \ge 0$ are sometimes called *nonnegativity constraints* or *restrictions*).

The LP is called *feasible* if there exists a nonnegative vector $\tilde{x}$ satisfying $A\tilde{x} \le b$ (such an $\tilde{x}$ is called a *feasible solution*); otherwise the program is called *infeasible.* If there are feasible solutions with arbitrarily small objective function value, the LP is called *unbounded*; otherwise it is *bounded.* A linear program which is both feasible and bounded has a unique minimum value $c^T \tilde{x}$ attained at a (not necessarily unique) optimal feasible solution $\tilde{x}$. Solving the LP means finding such an optimal solution $\tilde{x}$ (if it exists).

Geometric interpretation. For each constraint

$$
\begin{aligned}
\sum_{j=1}^n a_{ij} x_j &\le b_i \quad \text{or} \\
x_j &\ge 0,
\end{aligned}
$$

the points $\tilde{x} \in \mathbb{R}^n$ satisfying the constraint form a closed *halfspace* in $\mathbb{R}^n$. The points for which equality holds form the boundary of this halfspace, the *constraint hyperplane.*

The set of feasible solutions of the LP is therefore an intersection of halfspaces, which is by definition a (possibly empty) *polyhedron* P. The facets of P are induced by (not necessarily all) constraint hyperplanes. The nonnegativity constraints $x_j \ge 0$ restrict P to lie inside the positive orthant of $\mathbb{R}^n$.

Let us do an example and consider the problem

$$
\begin{array}{lrcrcl}
\text{minimize} & -x_1 - x_2 & & & & \\
\text{subject to} & -x_1 & + & x_2 & \le & 1, \\
& x_1 & & & \le & 3, \\
& & & x_2 & \le & 2, \\
& x_1, x_2 \ge 0. & & & &
\end{array}
\qquad (1.4)
$$

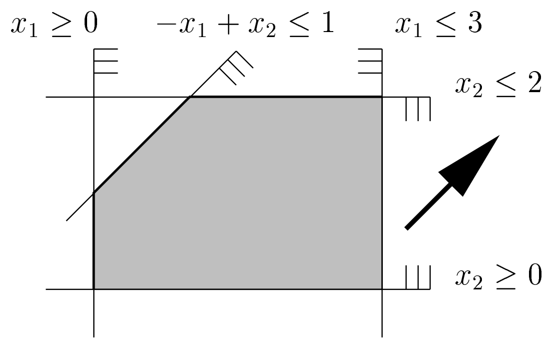


Figure 1.6: Geometric interpretation of LP

Figure 1.6 shows the feasible region of this LP. The fat arrow indicates the vector $-c^T = (1, 1)$, the so-called *optimization direction.* The objective is to find a feasible point that is extreme in the optimization direction. In the example, a unique such point exists, namely the upper right corner of the feasible region. It is easy to see that this corner has coordinates $(3, 2)$ and therefore objective function value -5. We will come back to this example in the next chapter where we show how the simplex method gets to this optimal corner.

1.3 Quadratic Programming

As the name suggests, the objective function in a quadratic program is no longer linear but quadratic, of the general form

$$\sum_{j=1}^{n} c_j x_j + \sum_{1 \leq i \leq j \leq n} d_{ij} x_i x_j .$$

The constraints and nonnegativity restrictions are as before. Defining a symmetric matrix

$$D := \begin{pmatrix} d_{11} & d_{12}/2 & \cdots & d_{1n}/2 \\ d_{12}/2 & d_{22} & \cdots & d_{2n}/2 \\ \vdots & & \vdots & \\ d_{1n}/2 & d_{2n}/2 & \cdots & d_{nn} \end{pmatrix},$$

we can write a quadratic programming problem in the form

$$\begin{array}{lll} \text{(QP)} & \text{minimize} & c^T x + x^T D x \\ & \text{subject to} & Ax \leq b, \\ & & x \geq 0. \end{array} \qquad (1.5)$$

In this manuscript, we restrict our attention to a special class of QP problems, defined by matrices D which are *positive semidefinite.* This means that

$$x^T D x \geq 0, \ \forall x \in I\!R^n.$$

As we will see later, this condition ensures that – like (LP) – problem (QP) has at most one local minimum which is at the same time a global minimum.

An example for an interesting quadratic programming problem is obtained when $c = 0, D = E$, E the $n \times n$-unit matrix. In this case, the function to minimize is $x^T x = \|x\|^2$. The problem is then to find the minimum-norm point in the polyhedron defined by the constraints of the problem.

Chapter 2

Linear Programming and the Simplex Method

In this chapter, we describe the standard and the revised simplex method for solving LP problems. We proceed by explaing the standard simplex method, by example. This presentation is very informal but covers all the basic aspects of the method. The revised simplex method is then just a view from a slightly different perspective.

2.1 The Standard Simplex Method

2.1.1 Tableaus

When confronted with an LP in standard form (1.2), the simplex method starts off by introducing *slack variables* $x_{n+1}, \ldots, x_{n+m}$ to transform the inequality system $Ax \leq b$ into an equivalent system of equalities and additional nonnegativity constraints on the slack variables. The slack variable x_{n+i} closes the gap between the left-hand side and right-hand side of the i-th constraint, i.e.

$$x_{n+i} := b_i - \sum_{j=1}^{n} a_{ij} x_j,$$

for all $i = 1, \ldots, m$. The i-th constraint is then equivalent to

$$x_{n+i} \geq 0,$$

and the linear program can be written as

$$\begin{array}{llll} \text{(LP)} & \text{minimize} & \sum_{j=1}^{n} c_j x_j & \\ & \text{subject to} & x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j & (i = 1, \ldots, m), \\ & & x_j \geq 0 & (j = 1, \ldots, n+m), \end{array} \tag{2.1}$$

or in a more compact form as

$$\begin{array}{lll} \text{(LP)} & \text{minimize} & \underline{c}^T x \\ & \text{subject to} & \underline{A}x + Ex' = b, \\ & & x, x' \geq 0, \end{array} \tag{2.2}$$

where x' is the vector of slack variables.

Together with the objective function, the m equations for the x_{n+i} in (2.1) contain all the information about the LP. Following tradition, we will represent this information in *tableau* form where the objective function – denoted by z – is written last and separated from the other equations by a solid line. (The restrictions $x_j \geq 0$ do not show up in the tableau but represent implicit knowledge.) In this way we obtain the *initial tableau* for the LP.

$$\begin{array}{lclcccc} x_{n+1} & = & b_1 & -a_{11}x_1 & - & \cdots & -a_{1n}x_n \\ & & & \vdots & & & \\ x_{n+m} & = & b_m & -a_{m1}x_1 & - & \cdots & -a_{mn}x_n \\ \hline z & = & & c_1x_1 & + & \cdots & +c_nx_n \end{array} \tag{2.3}$$

Let's illustrate the process of getting the initial tableau from an LP in standard form for our initial example (1.4).

After introducing slack variables x_3, x_4, x_5, the LP in equality form is

$$\begin{array}{lcccccccc} \text{minimize} & -x_1 - x_2 & & & & & & & \\ \text{subject to} & & x_3 & = & 1 & + & x_1 & - & x_2, \\ & & x_4 & = & 3 & - & x_1, & & \\ & & x_5 & = & 2 & & & - & x_2, \\ & x_1, \ldots, x_5 \geq 0. & & & & & & & \end{array} \tag{2.4}$$

From this we obtain the initial tableau

$$\begin{array}{lcccccc} x_3 & = & 1 & + & x_1 & - & x_2 \\ x_4 & = & 3 & - & x_1 & & \\ x_5 & = & 2 & & & - & x_2 \\ \hline z & = & & - & x_1 & - & x_2 \end{array} \tag{2.5}$$

Abstracting from the initial tableau (2.3), a general tableau for the LP is any system $\mathcal{T}$ of $m+1$ linear equations in the variables $x_1, \ldots, x_{n+m}$ and z, with the properties that

(i) $\mathcal{T}$ expresses m left-hand side variables x_B and z in terms of the remaining n right-hand side variables x_N, i.e. there is an m-vector β, a n-vector γ, an $m \times n$-matrix Λ and a real number z_0 such that $\mathcal{T}$ (written in compact form) is the system

$$\begin{array}{lcccc} x_B & = & \beta & - & \Lambda x_N \\ \hline z & = & z_0 & + & \gamma^T x_N \end{array} \tag{2.6}$$

(ii) Any solution of (2.6) is a solution of (2.3) and vice versa.

By property (ii), any tableau contains the *same* information about the LP but represented in a *different* way. All that the simplex algorithm is about is constructing a sequence of tableaus by gradually rewriting them, finally leading to a tableau in which the information is represented in such a way that the desired optimal solution can be read off directly. We will immediately show how this works in our example.

2.1.2 Pivoting

Here is the initial tableau (2.5) to (1.4) again.

$$\begin{array}{rcrrrrr} x_3 & = & 1 & + & x_1 & - & x_2 \\ x_4 & = & 3 & - & x_1 & & \\ x_5 & = & 2 & & & - & x_2 \\ \hline z & = & & - & x_1 & - & x_2 \end{array}$$

By setting the right-hand side variables x_1, x_2 to zero, we find that the left-hand side variables x_3, x_4, x_5 assume nonnegative values $x_3 = 1, x_4 = 3, x_5 = 2$. This means, the vector $x = (0, 0, 1, 3, 2)$ is a feasible solution of (2.4) (and the vector $x' = (0, 0)$ is a feasible solution of (1.4)). The objective function value $z = 0$ associated with this feasible solution is computed from the last row of the tableau. In general, any feasible solution that can be obtained by setting the right-hand side variables of a tableau to zero is called a *basic feasible solution* (BFS). In this case we also refer to the tableau as a feasible tableau. The left-hand side variables of a feasible tableau are called *basic* and are said to constitute a *basis*, the right-hand side ones are *nonbasic*. The goal of the simplex algorithm is now either to construct a new feasible tableau with a corresponding BFS of smaller z-value, or to prove that there exists no feasible solution at all with smaller z-value. In the latter case the BFS obtained from the tableau is reported as an optimal solution to the LP; in the former case, the process is repeated, starting from the new tableau.

In the above tableau we observe that increasing the value of x_1 (i.e. making x_1 positive) will decrease the z-value. The same is true for x_2, and this is due to the fact that both variables have negative coefficients in the z-row of the tableau. Let us arbitrarily choose x_2. By how much can we increase x_2? If we want to maintain feasibility, we have to be careful not to let any of the basic variables go below zero. This means, the equations determining the values of the basic variables may limit x_2's increment. Consider the first equation

$$x_3 = 1 + x_1 - x_2. \tag{2.7}$$

Together with the implicit constraint $x_3 \geq 0$, this equation lets us increase x_2 up to the value $x_2 = 1$ (the other nonbasic variable x_1 keeps its zero value). The second equation

$$x_4 = 3 - x_1$$

does not limit the increment of x_2 at all, and the third equation

$$x_5 = 2 - x_2$$

allows for an increase up to the value $x_2 = 2$ before x_5 gets negative. The most stringent restriction therefore is $x_3 \geq 0$, imposed by (2.7), and we will increase x_2 just as much as we can, so we get $x_2 = 1$ and x_3=0. From the remaining tableau equations, the values of the other variables are obtained as

$$\begin{aligned} x_4 &= 3 - x_1 = 3, \\ x_5 &= 2 - x_2 = 1. \end{aligned}$$

To establish this as a BFS, we would like to have a tableau with the new zero variable x_3 replacing x_2 as a nonbasic variable. This is easy – the equation (2.7) which determined the new value of x_2 relates both variables. This equation can be rewritten as

$$x_2 = 1 + x_1 - x_3,$$

and substituting the right-hand side for x_2 into the remaining equations gives the new tableau

$$\begin{array}{rcrcrcr} x_2 & = & 1 & + & x_1 & - & x_3 \\ x_4 & = & 3 & - & x_1 & & \\ x_5 & = & 1 & - & x_1 & + & x_3 \\ \hline z & = & -1 & - & 2x_1 & + & x_3 \end{array}$$

with corresponding BFS $x = (0, 1, 0, 3, 1)$ and objective function value $z = -1$. This process of rewriting a tableau into another one is called a *pivot step*, and it is clear by construction that both systems have the same set of solutions. The effect of a pivot step is that a nonbasic variable (in this case x_2) *enters* the basis, while a basic one (in this case x_3) *leaves* it. Let us call x_2 the *entering variable* and x_3 the *leaving variable.*

In the new tableau, we can still increase x_1 and obtain a smaller z-value. x_3 cannot be increased since this would lead to larger z-value. The first equation puts no restriction on the increment, from the second one we get $x_1 \leq 3$ and from the third one $x_1 \leq 1$. So the third one is the most stringent, will be rewritten and substituted into the remaining equations as above. This means, x_1 enters the basis, x_5 leaves it, and the tableau we obtain is

$$\begin{array}{rcrcrcr} x_2 & = & 2 & & & - & x_5 \\ x_4 & = & 2 & - & x_3 & + & x_5 \\ x_1 & = & 1 & + & x_3 & - & x_5 \\ \hline z & = & -3 & - & x_3 & + & 2x_5 \end{array}$$

with BFS $x = (1, 2, 0, 2, 0)$ and $z = -3$. Performing one more pivot step (this time with x_3 the entering and x_4 the leaving variable), we arrive at the tableau

$$\begin{array}{rcrcrcr} x_2 & = & 2 & & & - & x_5 \\ x_3 & = & 2 & - & x_4 & + & x_5 \\ x_1 & = & 3 & - & x_4 & & \\ \hline z & = & -5 & + & x_4 & + & x_5 \end{array} \tag{2.8}$$

with BFS $x = (3, 2, 2, 0, 0)$ and $z = 5$. In this tableau, no nonbasic variable can increase without making the objective function value larger, so we are stuck. Luckily, this means that we have already found an optimal solution. Why? Consider any *feasible* solution $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_5)$ for (2.4), with objective function value z_0. This is a solution to (2.5) and therefore a solution to (2.8). Thus,

$$z_0 = -5 + \tilde{x}_4 + \tilde{x}_5$$

must hold, and together with the implicit restrictions $x_4, x_5 \geq 0$ this implies $z_0 \geq -5$. The tableau even delivers a proof that the BFS we have computed is the unique optimal solution to the problem: $z = -5$ implies $x_4 = x_5 = 0$, and this determines the values of the other variables. Ambiguities occur only if some of the nonbasic variables have zero coefficients in the z-row of the final tableau. Unless a specific optimal solution is required, the simplex algorithm in this case just reports the optimal BFS it has at hand.

What do the algebraic manipulations we have just done correspond to in the geometric picture as given in Figure 1.6? The following fact helps us to understand this.

Fact 2.1 *Consider a standard form LP with feasible polyhedron P. The point $\tilde{x}' = (\tilde{x}_1, \ldots, \tilde{x}_n)$ is a vertex of P if and only if the vector $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_{n+m})$ with*

$$\tilde{x}_{n+i} := b_i - \sum_{j=1}^{n} a_{ij}\tilde{x}_j, \ i = 1, \ldots, m$$

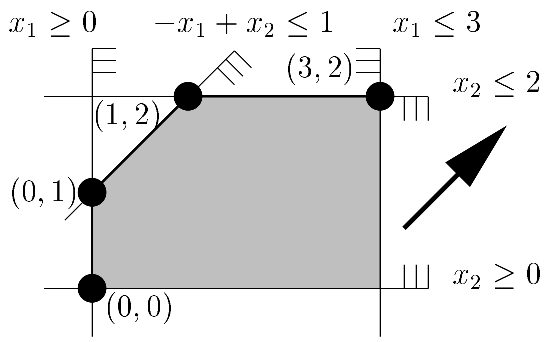
is a basic feasible solution of the LP.

This means that the simplex method only works on vertices ('corners') of the feasible region P, and it traverses a sequence of vertices until an optimal one is found.

Two consecutive tableaus constructed by the simplex method have $n - 1$ nonbasic variables in common. Their BFS thus share $n - 1$ zero variables. Equivalently, the corresponding vertices lie on $n - 1$ common constraint hyperplanes, and this means that they are adjacent in P. The feasible solutions obtained in the process of continuously increasing the value of a nonbasic variable until it becomes basic correspond to the points on the edge of P connecting the two vertices. Establishing these facts formally requires at least some basic polyhedra theory.

Here we are content with checking the correlations in case of Example 1.4. The LP consists of five constraints over two variables, therefore the feasible region is a polygon in $\mathbb{R}^2$. Every constraint hyperplane defines a facet, so we get a polygon with five edges and five vertices. In the previous subsection we were going through a sequence of four tableaus until we discovered an optimal BFS. The picture below shows how this corresponds to a sequence of adjacent vertices. Since the objective function value gets smaller in every iteration, the path of vertices is monotone in the optimization direction $-c$.

BFS	vertex	$z = x_1 + x_2$
$(0,0,1,3,4)$	$(0,0)$	0
$(0,1,0,3,1)$	$(0,1)$	1
$(1,2,0,2,0)$	$(1,2)$	3
$(3,2,2,0,0)$	$(3,2)$	5



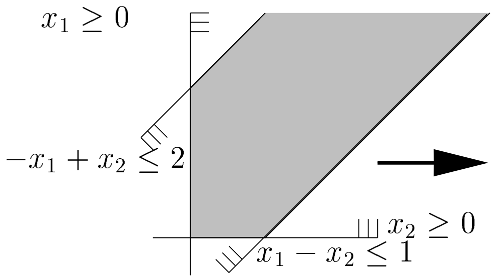
2.2 Exception Handling

So far our outline of the simplex method went pretty smooth. This is in part due to the fact that we have only seen one very small and trivial example of the way it works. On the other hand, the method *is* simple, and we will just incorporate some 'exception handling' and do a little fine tuning, again basically by example.

2.2.1 Unboundedness

During a pivot step, we make the value of a nonbasic variable just large enough to get the value of a basic variable down to zero. This, however, might never happen. Consider the example

$$\begin{array}{lrcrcl}
\text{minimize} & -x_1 & & & & \\
\text{subject to} & x_1 & - & x_2 & \leq & 1, \\
 & -x_1 & + & x_2 & \leq & 2, \\
 & \multicolumn{5}{l}{x_1, x_2 \geq 0.}
\end{array}$$



with initial tableau

$$\begin{array}{rclcrcr}
x_3 & = & 1 & - & x_1 & + & x_2 \\
x_4 & = & 2 & + & x_1 & - & x_2 \\
\hline
z & = & & - & x_1 & &
\end{array}$$

After one pivot step with x_1 entering the basis we get the tableau

$$\begin{array}{rcrcrcr}
x_1 & = & 1 & + & x_2 & - & x_3 \\
x_4 & = & 3 & & & - & x_3 \\
\hline
z & = & -1 & - & x_2 & + & x_3
\end{array}$$

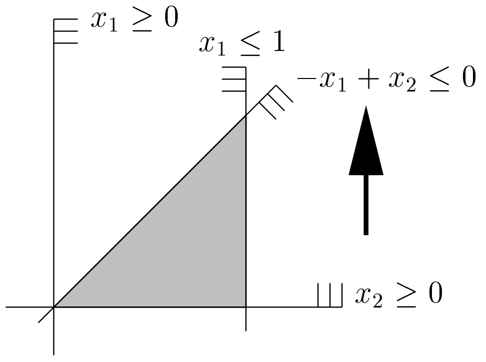
If we now try to bring x_2 into the basis by increasing its value, we notice that none of the tableau equations puts a limit on the increment. We can make x_2 arbitrarily large and z arbitrarily small – the problem is unbounded. By letting x_2 go to infinity we get a feasible halfline – starting from the current BFS – as a witness for the unboundedness. In our case this is the set of feasible solutions

$$\{(1,0,0,3) + x_2(1,1,0,0) \mid x_2 \geq 0\}.$$

Such a halfline will typically be the output of the algorithm in the unbounded case. Thus, unboundedness can quite naturally be handled with the existing machinery. In the geometric interpretation it just means that the feasible polyhedron P is unbounded in the optimization direction.

2.2.2 Degeneracy

While we can make some nonbasic variable arbitrarily large in the unbounded case, just the other extreme happens in the degenerate case: some tableau equation limits the increment to zero so that no progress in z is possible. Consider the LP



$$\begin{array}{lrcrcl} \text{minimize} & -x_2 & & & & \\ \text{subject to} & -x_1 & + & x_2 & \leq & 0, \\ & x_1 & & & \leq & 2, \\ & x_1, x_2 \geq 0, & & & & \end{array} \tag{2.9}$$

with initial feasible tableau

$$\begin{array}{rcrcrcr} x_3 & = & & & x_1 & - & x_2 \\ x_4 & = & 2 & - & x_1 & & \\ \hline z & = & & & & - & x_2 \end{array}$$

The only candidate for entering the basis is x_2, but the first tableau equation shows that its value cannot be increased without making x_3 negative. This may happen whenever in a BFS some basic variables assume zero value, and such a situation is called *degenerate*. Unfortunately, the impossibility of making progress in this case does not imply optimality, so we have to perform a 'zero progress' pivot step. In our example, bringing x_2 into the basis results in another degenerate tableau with the same BFS.

$$\begin{array}{rcrcrcr} x_2 & = & & & x_1 & - & x_3 \\ x_4 & = & 2 & - & x_1 & & \\ \hline z & = & & - & x_1 & + & x_3 \end{array}$$

Nevertheless, the situation has improved. The nonbasic variable x_1 can be increased now, and by entering it into the basis (replacing x_4) we already obtain the final tableau

$$\begin{array}{rcrcrcr} x_1 & = & 2 & & & - & x_4 \\ x_2 & = & 2 & - & x_3 & - & x_4 \\ \hline z & = & -2 & + & x_3 & + & x_4 \end{array}$$

with optimal BFS $x = (x_1, \ldots, x_4) = (2, 2, 0, 0)$.

In this example, after one degenerate pivot we were able to make progress again. In general, there might be longer runs of degenerate pivots. Even worse, it may happen that a tableau repeats itself during a sequence of degenerate pivots, so the algorithm can go through an infinite sequence of tableaus without ever making progress. This phenomenon is known as *cycling*. If the algorithm does not terminate, it must cycle. This follows from the fact that there are only finitely many different tableaus.

Fact 2.2 *The LP (1.2) has at most $\binom{m+n}{m}$ tableaus.*

To prove this, we show that any tableau $\mathcal{T}$ is already determined by its basis variables. Write $\mathcal{T}$ as

$$\begin{array}{ccccc} x_B & = & \beta & - & \Lambda x_N \\ \hline z & = & z_0 & + & \gamma^T x_N, \end{array}$$

and assume there is another tableau $\mathcal{T}'$ with the same basic and nonbasic variables, i.e. $\mathcal{T}'$ is the system

$$\begin{array}{ccccc} x_B & = & \beta' & - & \Lambda' x_N \\ \hline z & = & z_0' & + & \gamma'^T x_N, \end{array}$$

By the tableau properties, both systems have the same set of solutions. Therefore

$$\begin{array}{rcl} (\beta - \beta') - (\Lambda - \Lambda')x_N & = & 0 \text{ and} \\ (z_0 - z_0') + (\gamma^T - \gamma'^T)x_N & = & 0 \end{array}$$

must hold for all n-vectors x_N, and this implies $\beta = \beta', \Lambda = \Lambda', \gamma = \gamma'$ and $z_0 = z_0'$. Hence $\mathcal{T} = \mathcal{T}'$.

There are two standard ways to avoid cycling:

- Bland's *smallest subscript rule*: If there is more than one candidate x_k for entering the basis (or more than one candidate for leaving the basis, which is another manifestation of degeneracy), choose the one with smallest subscript k.
- Avoid degeneracies altogether by symbolic perturbation.

By Bland's rule, there *is* always a way of escaping from a sequence of degenerate pivots. For this, however, one has to give up the freedom of choosing the entering variable. This is unfavourable in many instances, and one resorts to the method of symbolic perturbation instead, although this requires more computational effort. The method – also known as the *lexicographic* method – perturbs the right-hand side vector b of the LP by adding powers of a symbolic constant ε (assumed to be infinitesimally small). The LP then becomes

$$\begin{array}{lll} \text{minimize} & \sum_{j=1}^n c_j x_j & \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j \leq b_i + \varepsilon^i & (i = 1, \ldots, m), \\ & x_j \geq 0 & (j = 1, \ldots, n), \end{array} \tag{2.10}$$

and if the original LP (1.2) is feasible, so is (2.10). A solution to (1.2) can be obtained from a solution to (2.10) by ignoring the contribution of ε, i.e. by setting ε to zero. Moreover, any valid tableau for (2.10) reduces to a valid tableau for (1.2) when the terms involving powers of ε are disregarded.

In case of (2.9), the initial tableau of the perturbed problem is

$$\begin{array}{ccccccc} x_3 & = & \varepsilon & + & x_1 & - & x_2 \\ x_4 & = & 2 + \varepsilon^2 & - & x_1 & & \\ \hline z & = & & & & - & x_2 \end{array}$$

Pivoting with x_2 entering the basis gives the tableau

$$\begin{array}{rcrcrcr} x_2 & = & \varepsilon & + & x_1 & - & x_3 \\ x_4 & = & 2+\varepsilon^2 & - & x_1 & & \\ \hline z & = & -\varepsilon & - & x_1 & + & x_3 \end{array} \tag{2.11}$$

This is no longer a degenerate pivot, since x_2 (and z) increased by ε. Finally, bringing x_1 into the basis gives the tableau

$$\begin{array}{rcrcrcr} x_1 & = & 2+\varepsilon^2 & & & - & x_4 \\ x_2 & = & 2+\varepsilon+\varepsilon^2 & - & x_3 & - & x_4 \\ \hline z & = & -2-\varepsilon-\varepsilon^2 & + & x_3 & + & x_4 \end{array} \tag{2.12}$$

with optimal BFS $x = (2+\varepsilon^2, 2+\varepsilon+\varepsilon^2, 0, 0)$. The optimal BFS for (2.9) is recovered from this by ignoring the additive terms in ε. In general, the following holds, which proves nondegeneracy of the perturbed problem.

Fact 2.3 *In any BFS of 2.10, the values of the basic variables are nonzero polynomials in ε, of degree at most m. The tableau coefficients at the nonbasic variables are unaffected by the perturbation.*

To find the leaving variable, polynomials in ε have to be compared. This is done lexicographically, i.e.

$$\sum_{k=1}^{m} \lambda_k \varepsilon^k < \sum_{k=1}^{m} \lambda'_k \varepsilon^k$$

if and only if $(\lambda_1, \ldots, \lambda_m)$ is lexicographically smaller than $(\lambda'_1, \ldots, \lambda'_m)$. The justification for this is that one could actually assign a very small numerical value to ε (depending on the input numbers of the LP), such that comparing lexicographically is equivalent to comparing numerically, for all polynomials that turn up in the algorithm.

In the perturbed problem, progress is made in every pivot step. Cycling cannot occur and the algorithm terminates after at most $\binom{m+n}{m}$ pivot steps.

In the feasible polyhedron, degeneracies correspond to 'overcrowded vertices', which are vertices where more than n of the constraint hyperplanes meet. There are several ways to represent the same vertex as an intersection of exactly n hyperplanes, and a degenerate pivot switches between two such representations. The perturbation slightly moves the hyperplanes relative to each other in such a way that any degenerate vertex is split into a collection of nondegenerate ones very close together.

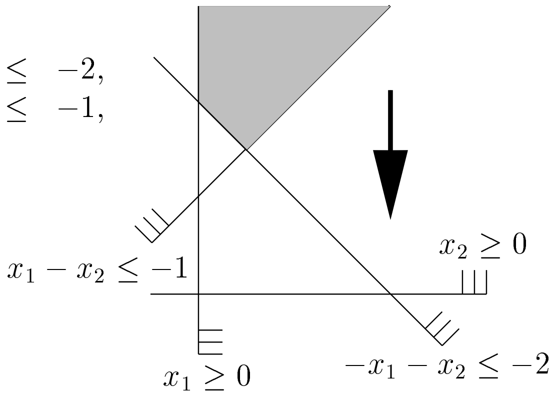
2.2.3 Infeasibility

To start off, the simplex method needs some feasible tableau. In all examples considered so far such a tableau was readily available since the initial tableau was feasible. We say that the problem has a *feasible origin*. This is equivalently expressed by the fact that the right-hand side vector b of the LP is nonnegative. If this is not the case, we first solve an auxiliary problem that either constructs a BFS to the original problem or proves that the original problem is infeasible. The auxiliary problem has an additional variable x_0 and is defined as

$$\begin{array}{lll} \text{minimize} & x_0 & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij}x_j - x_0 \leq b_i & (i = 1, \ldots, m), \\ & x_j \geq 0 & (j = 0, \ldots, n). \end{array}$$

This problem is feasible (choose x_0 big enough), and it is clear that the original problem is feasible if and only if the optimum value of the auxiliary LP is zero. Let us do an example and consider the problem

$$\begin{array}{lrcrcr} \text{minimize} & & & x_2 & & \\ \text{subject to} & -x_1 & - & x_2 & \leq & -2, \\ & x_1 & - & x_2 & \leq & -1, \\ & & x_1, x_2 \geq 0. & & & \end{array}$$



with initial tableau

$$\begin{array}{rcrcrcr} x_3 & = & -2 & + & x_1 & + & x_2 \\ x_4 & = & -1 & - & x_1 & + & x_2 \\ \hline z & = & & & & & x_2 \end{array}$$

This problem has an infeasible origin, because setting the right-hand side variables to zero gives $x_3 = -2, x_4 = -1$. The auxiliary problem (with the objective function called w in the tableau to avoid confusion) is

$$\begin{array}{lrcrcrcr} \text{minimize} & & & & & x_0 & & \\ \text{subject to} & -x_1 & - & x_2 & - & x_0 & \leq & -2, \\ & x_1 & - & x_2 & - & x_0 & \leq & -1, \\ & & x_0, x_1, x_2 \geq 0. & & & & & \end{array}$$

with initial tableau

$$\begin{array}{rcrcrcrcr} x_3 & = & -2 & + & x_1 & + & x_2 & + & x_0 \\ x_4 & = & -1 & - & x_1 & + & x_2 & + & x_0 \\ \hline w & = & & & & & & + & x_0 \end{array}$$

The auxiliary problem has an infeasible initial tableau, too, but we can easily construct a feasible tableau by performing one pivot step. We start increasing the value of x_0, this time not with the goal of *maintaining* feasibility but with the goal of *reaching* feasibility. To get $x_3 \geq 0$, x_0 has to increase by at least 2, and this also makes x_4 positive. By setting $x_0 := 2$ we get $x_3 = 0$ and $x_4 = 1$. Solving the first tableau equation for x_0 and substituting from this into the remaining equations as usual gives a new *feasible* tableau with x_0 basic and x_3 nonbasic.

$$\begin{array}{rcrcrcrcr} x_0 & = & 2 & - & x_1 & - & x_2 & + & x_3 \\ x_4 & = & 1 & - & 2x_1 & & & + & x_3 \\ \hline w & = & 2 & - & x_1 & - & x_2 & + & x_3 \end{array}$$

The simplex method can now be used to solve the auxiliary problem. In our case, by choosing x_2 as the entering variable, we accomplish this in one step. The resulting tableau is

$$\begin{array}{rclllllll} x_2 & = & 2 & - & x_1 & + & x_3 & - & x_0 \\ x_4 & = & 1 & - & 2x_1 & + & x_3 & & \\ \hline w & = & & & & & & & x_0 \end{array}$$

Since all coefficients of nonbasic variables in the w-row are nonnegative, this is an optimal tableau with BFS $x = (x_0, \ldots, x_4) = (0, 0, 2, 0, 1)$. The associated zero w-value asserts that the LP we originally wanted to solve is actually feasible, and we can even construct a feasible tableau for it from the final tableau of the auxiliary problem by ignoring x_0 and expressing the original objective function z in terms of the nonbasic variables; from the first tableau equation we get in our case $z = x_2 = 2 - x_1 + x_3$, and this gives a valid feasible tableau

$$\begin{array}{rclllll} x_2 & = & 2 & - & x_1 & + & x_3 \\ x_4 & = & 1 & - & 2x_1 & + & x_3 \\ \hline z & = & 2 & - & x_1 & + & x_3 \end{array}$$

with corresponding BFS $x = (x_1, \ldots, x_4) = (0, 2, 0, 1)$ for the original LP. For this to work, x_0 should be nonbasic in the final tableau of the auxiliary problem which is automatically the case if the problem is nondegenerate. (To guarantee this in the general situation, choose x_0 as the leaving variable whenever this is a possible choice.)

If the optimum value of the auxiliary problem is nonzero, we can conclude that the original LP is infeasible and simply report this fact.

2.3 The Revised Simplex Method

In the preceding sections we have quite informally covered all the basics one would need to come up with an actual implementation of the simplex method. Let us do a brief analysis of the runtime in terms of arithmetic operations. To find the entering variable, at most n entries of the tableau's z-row need to be examined, contributing time $O(n)$. Searching for the leaving variable takes constant time per basic variable, for a total of $O(m)$ time. Finally, the tableau update can be done at constant cost per entry, requiring time $O(mn)$. This means, the tableau update is the dominationg operation.

Considering this, you might wonder whether you really need to store and update the whole tableau in each iteration. Namely, if you search for an entering variable, you only need the z-row of the tableau, and for finding the leaving variable, all tableau columns except the one corresponding to the entering variable are irrelevant, so why do you need to keep them?

You would prefer to have a method that generates tableau entries 'on the fly', just when you need them. There is hope for such a method, because the argument explaining Fact

2.2 shows that the tableau is uniquely determined by the basic variables, so the tableau *has* a compact representation in form of the subscript set B. We just need to make this relationship between B and the tableau explicit in the sense that we get formulas to obtain the tableau entries from B.

The revised simplex method is centered around such formulas. It works on an implicit representation of the tableau and completely saves the tableau update. At the same time, the relationship between consecutive tableaus and their connection to the original data become clearer.

To describe the revised simplex method, let us now consider an LP in equality form, given as

$$\begin{array}{lll} \text{(LP)} & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \\ & & x \geq 0, \end{array} \tag{2.13}$$

with n variables and m equality constraints. In view of the previous sections, this LP may come from an LP in inequality form (1.3) after adding slack variables, or it might have been given to us in this format right from the start. Assume now, we are given a tableau

$$\begin{array}{ccccc} x_B & = & \beta & - & \Lambda x_N \\ \hline z & = & z_0 & + & \gamma^T x_N \end{array} \tag{2.14}$$

to (2.13), its entries β, γ, Λ and z_0 being determined by the choice of the basic variables x_B.

For subscript set $G \subseteq [n] := \{1, \ldots, n\}$ let A_G collect the columns corresponding to the variables with subscripts in G. Then the equations of (2.13) read as

$$A_B x_B + A_N x_N = b. \tag{2.15}$$

Since (2.14) has by definition of a tableau the same set of solutions as (2.15), the former is obtained by simply solving (2.15) for x_B, which gives

$$x_B = A_B^{-1} b - A_B^{-1} A_N x_N, \tag{2.16}$$

and therefore

$$\begin{array}{lll} \beta & = & A_B^{-1} b, \\ \Lambda & = & A_B^{-1} A_N. \end{array} \tag{2.17}$$

By similar reasoning we compute γ and z_0. As before, let c_G collect the entries of c corresponding to variables with subscripts in G. Then the equation for z in (2.13) reads as

$$z = c_B^T x_B + c_N^T x_N. \tag{2.18}$$

Again by the tableau property, the last row of (2.14) is equivalent to (2.18), and the former is obtained by simply substituting from (2.16) into (2.18), which gives

$$z = c_B^T A_B^{-1} b + (c_N^T - c_B^T A_B^{-1} A_N) x_N, \tag{2.19}$$

and therefore

$$\begin{array}{rcl} z_0 & = & c_B^T A_B^{-1} b, \\ \gamma^T & = & c_N^T - c_B^T A_B^{-1} A_N. \end{array} \tag{2.20}$$

Rather than maintaining the tableau (2.14) explicitly, the revised simplex method maintains only the current BFS, the basic subscript set B as well as the *inverse* of the matrix A_B. Note that if (2.13) has been obtained by adding slack variables to (1.3), then before the first iteration, $A_B = A_B^{-1} = E$, E the unit matrix. How to obtain a first basis B in case the LP is directly given in the format (2.13) is an exercise.

Let us now recall the details of the pivoting step and see how they are realized in the revised simplex method. It is important to note that we are *not* going to see anything new, we are just taking a look from a different angle.

The substeps of the revised simplex method have certain historical names. The process of finding the entering variable is known as *pricing*, to find the leaving variable, we perform a *ratio test*, and the process of going from B to B' and A_B^{-1} to $A_{B'}^{-1}$, B' the new basis, is known as the *update*.

2.3.1 Pricing

To find the entering variable, we evaluate γ^T according to (2.20), i.e. set

$$y^T := c_B^T A_B^{-1} \tag{2.21}$$

and

$$\gamma^T = c_N^T - y^T A_N. \tag{2.22}$$

This takes $O(m^2)$ time to compute y^T, given that A_B^{-1} is available, and $O(m)$ time for every entry of γ^T. A variable x_j may enter the basis if its corresponding γ-entry is negative. Thus, to find an entering variable, we do not necessarily have to evaluate *all* the entries of γ^T, we may stop as soon as we have found a negative one. This observation is a potential source of major savings in comparison with the standard simplex method, although in the worst case all entries of γ may have to be evaluated, leading to $O(mn)$ arithmetic operations again.

2.3.2 Ratio Test

From (2.14) we immediately see that if we increase the value of the entering variable x_j to some value $t \geq 0$, the values of the basic variables change according to the formula

$$x_B(t) = \beta - \Lambda_j t = \beta - A_B^{-1} A_j t. \tag{2.23}$$

Recall that the leaving variable was a variable $x_i, i \in B$ which becomes zero first when increasing t. For each $i \in B$, the value t_i such that $x_i(t_i) = 0$ is the solution of the linear equation

$$\beta^i = \Lambda_j^i t_i,$$

equivalently

$$t_i = \frac{\beta^i}{\Lambda_j^i}, \tag{2.24}$$

β^i and Λ_j^i the components of β resp. Λ_j corresponding to variable x_i. Equation (2.24) explains the term *ratio test.*

While the BFS β is assumed to be available, te column Λ_j of the tableau matrix Λ corresponding to the entering variable x_j is computed according to (2.17), i.e. by setting

$$\Lambda_j := A_B^{-1} A_j. \tag{2.25}$$

Again, since A_B^{-1} is available, this can be done in time $O(m^2)$.

The leaving variable and the new BFS are easily computed from this (and the old BFS) in time $O(m)$, just like the standard simplex method does it from the explicit tableau representation. One remark is in order. If we apply the method of symbolic perturbation to cope with degeneracies, this step gets more expensive. In order to obtain the new BFS, in the worst case n polynomials of degree up to n in ε have to be compared and updated (consider the step from (2.11) to (2.12) in Subsection 2.2.2). This takes time $O(m^2)$ rather than $O(m)$ when only actual numbers are involved. Luckily, this does not introduce asymptotic overhead, since the previous steps already take $O(m^2)$ time.

2.3.3 Update

The tableau update is replaced by an update of the set B and the matrix A_B^{-1}. Considering B as an ordered set, B' arises from B by replacing the leaving index i at some position k with the entering index j. This means, $A_{B'}$ arises from A_B by replacing the k-th column, and the new column is exactly the column A_j. Invoking (2.25), we get

$$A_{B'} = A_B \begin{pmatrix} 1 & \cdots & \Lambda_j^1 & \cdots & 0 \\ & \ddots & \vdots & & \\ & & & & \\ & & \vdots & \ddots & \\ 0 & \cdots & \Lambda_j^n & \cdots & 1 \end{pmatrix}, \tag{2.26}$$

so A_B is updated by multiplying with a unit matrix whose k-th column is replaced by the column Λ_j already computed in (2.25). Such a matrix is known as an *eta matrix.* (2.26) now implies

$$A_{B'}^{-1} = \begin{pmatrix} 1 & \cdots & -\Lambda_j^1/\Lambda_j^k & \cdots & 0 \\ & \ddots & \vdots & & \\ & & 1/\Lambda_j^k & & \\ & & \vdots & \ddots & \\ 0 & \cdots & -\Lambda_j^n/\Lambda_j^k & \cdots & 1 \end{pmatrix} A_B^{-1},$$

and this multiplication can be performed in time $O(m^2)$.

As a consequence, a pivot step in the revised simplex method can be done in time $O(mn)$, as before. However, the dominating step is now the pricing, where savings are possible. Moreover, the revised simplex method has less space reqirements because only an $m \times m$-matrix rather than the whole tableau of size $O(mn)$ needs to be stored. If n is quite large, this makes a tremendous difference.

2.4 Complexity

It is clear from the previous discussions that the worst-case complexity of the simplex method is $O(mnI)$, where I is the number of iterations. This number I can be almost anything, and it crucially depends on the problem and on the way the entering variable is chosen in case there is more than one choice. The rule according to which the selection is then made is called a *pivot rule*.

The pivot rule originally proposed by Dantzig for his method is the following: choose the variable j with the smallest value of γ_j. The intuition is that by bringing this variable into the basis, the objective function decreases most rapidly. Namely, if variable x_j is set to $t \geq 0$, the objective function value changes from z_0 to $z_0 + \gamma_j t$.

Under this rule, also known as *Dantzig's rule*, the typical number of iterations of the simplex method only depends on m. This means, if m is a constant, the simplex algorithm will in many cases be a linear time algorithm. But beware: there are examples where the number of iterations is much larger, even exponential in m. However, for the applicability of the simplex method in practice, such examples are not too relevant, and the linear behaviour is usually observed in most cases.

Whenever we apply the simplex method (and its quadratic-programming counterpart in Chapter 4) in computational geometry, we have the situation that $n \gg m$, where m is very small or constant. From what we have just said, we then expect fast linear-time algorithms for the problems we consider.

If you really want provable results, there exist (randomized) pivot rules which guarantee an expected runtime of $O(n)$. Their drawback is that they are not that efficient in practice. What is still missing is a practically fast implementation which is provably good.

Chapter 3

Convex Programming and the Kuhn-Tucker Conditions

In the previous chapter we have considered *linear* programs, i.e. problems of minimizing a linear function c^Tx in n variables x, subject to constraints

$$\begin{aligned} &Ax \leq b, \\ &x \geq 0 \end{aligned}$$

(inequality form) or

$$\begin{aligned} &Ax = b, \\ &x \geq 0 \end{aligned}$$

(equality form). The simplex method actually works with problems in equality form, but the inequality form is a convenient starting point, because after introducing slack variables, an initial tableau (although not necessarily a feasible one) is readily available. As one form may be more convenient than the other in some considerations, we will keep using both constraint formats.

In this chapter, the objective function will no longer be restricted to be a linear function. As indicated in the introduction, our ultimate goal is to minimize quadratic functions $c^Tx + x^TDx$, where D is a symmetric, positive semidefinite matrix, but for this chapter, we focus on the even more general class of continuosly differentiable *convex* functions (we omit the condition 'continuosly differentiable' in all subsequent statements).

This means, we consider problems of minimizing a convex function subject to linear (in)equality and nonnegativity constraints. Such problems are referred to as *Convex Programming* (CP). For any n-vector c and any symmetric, positive semidefinite $n \times n$-matrix D, the quadratic function

$$c^Tx + x^TDx$$

is such a convex function.

3.1 Basics about Convex Functions

First of all, what is a convex function? Here is the formal definition.

Definition 3.1 *$f : \mathbb{R}^n \mapsto \mathbb{R}$ is* convex *if and only if for all $x, x' \in \mathbb{R}^n$ and all $t \in [0,1]$,*

$$f((1-t)x + tx') \leq (1-t)f(x) + tf(x') \tag{3.1}$$

holds.

This means, a convex function looks like a 'bowl': whenever you connect two points $f(x), f(x')$ by a line segment, this line segment lies above the graph of the function. Note that we do not require strict convexity, so a linear function is convex, too, although it does not quite look like a bowl. An alternative definition of a convex function is obtained by requiring the region

$$f^+ := \{(x, x_{n+1}) \in \mathbb{R}^{n+1} \mid x_{n+1} \geq f(x)\}$$

above the graph of f to be a convex set, meaning that with any two points $x, x' \in f^+$, their convex combination $(1-t)x + tx', t \in [0,1]$ is also in f^+.

The following lemma explains why convex functions are 'nice': over a convex domain, they do not have any proper local minima in which one might get stuck during the optimization process.

Lemma 3.2 *Let $C \subseteq \mathbb{R}^n$ be convex. $x \in C$ is a (global) minimum of f over C if and only if*

$$\nabla f(x)(x' - x) \geq 0$$

for all $x' \in C$, where

$$\nabla f(x) := \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)$$

is the gradient of f at x.

Proof. Consider the convex combination $x(t)$ of x and x', given as

$$x(t) := (1-t)x + tx', \; t \in [0,1].$$

Because C is a convex set, $x(t)$ is again in C, and

$$\frac{\partial}{\partial t} f(x(t))|_{t=0} \geq 0$$

must hold, otherwise we had

$$\frac{\partial}{\partial t} f(x(t))|_{t=0} = \lim_{t \to 0} \frac{f(x(t)) - f(x)}{t} < 0,$$

and $f(x(t)) < f(x)$ would hold for some small t. On the other hand,

$$\frac{\partial}{\partial t} f(x(t))|_{t=0} = \nabla f(x)(x' - x),$$

by the chain rule.

If x is not a minimum, let x' be any better solution. By convexity we can then argue that

$$\frac{f(x(t)) - f(x)}{t} \leq f(x') - f(x) = \delta < 0,$$

for all $t \in [0, 1]$. This implies

$$\nabla f(x)(x' - x) = \lim_{t \to 0} \frac{f(x(t)) - f(x)}{t} \leq \delta < 0.$$

$\square$

3.2 Kuhn-Tucker Theorems

Now we are ready to prove the so-called *Kuhn-Tucker conditions* which provide optimality criteria for exactly the class of problems we are interested in. Let us start with problems in inequality form.

Theorem 3.3 *(Kuhn-Tucker conditions for inequality constraints.) Consider the problem*

$$\begin{array}{lll} (CP) & \textit{minimize} & f(x) \\ & \textit{subject to} & Ax \leq b, \\ & & x \geq 0, \end{array} \tag{3.2}$$

x an n-vector, A an $m \times n$-matrix, b an m-vector, $f : \mathbb{R}^n \mapsto \mathbb{R}$ a convex function. A feasible solution $\tilde{x} \in \mathbb{R}^n$ is optimal for CP if and only if there is an m-vector $\lambda \geq 0$ and an n-vector $\mu \geq 0$ such that

(i) $\nabla f(\tilde{x}) = -\lambda^T A + \mu$,

(ii) $\lambda^T (A\tilde{x} - b) = 0$,

(iii) $\mu^T \tilde{x} = 0$.

Before we give a proof of this result, let us discuss its geometric interpretation. Formulated in a complicated way, condition (i) means that the gradient vector of f at the point $\tilde{x}$ is a negative linear combination of the rows of A and the rows of the negative $n \times n$-unit matrix $-E$. These rows are exactly the normal vectors of the hyperplanes that delimit the constraint halfspaces

$$\begin{array}{rcl} Ax & \leq & b, \\ -x & \leq & 0, \end{array}$$

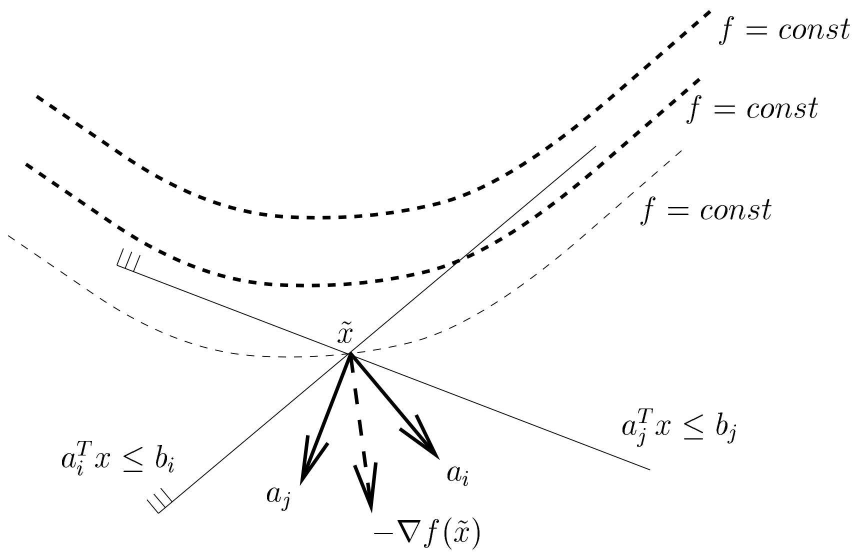


Figure 3.1: The Kuhn-Tucker optimality condition

where each normal vector 'points away' from its halfspace. Conditions (ii) and (iii) say that only those constraints contribute to the linear combination that are *binding* at $\tilde{x}$, i.e. that hold with equality at $\tilde{x}$. Thus, $\tilde{x}$ is optimal if and only if $-\nabla f(\tilde{x})$ lies in the cone spanned by the normal vectors of the binding constraints (Figure 3.1).

The Theorem will be a corollary of the following statement about equality-constrained CP.

Theorem 3.4 *(Kuhn-Tucker conditions for equality constraints.) Consider the problem*

$$\begin{array}{lll} \textit{(CP)} & \textit{minimize} & f(x) \\ & \textit{subject to} & Ax = b, \\ & & x \geq 0, \end{array} \tag{3.3}$$

x an n-vector, A an $m \times n$-matrix, b an m-vector, $f : \mathbb{R}^n \mapsto \mathbb{R}$ a convex function. A feasible solution $\tilde{x} \in \mathbb{R}^n$ is optimal for CP if and only if there is an m-vector λ and an n-vector $\mu \geq 0$ such that

(i) $\nabla f(\tilde{x}) = -\lambda^T A + \mu$,

(ii) $\mu^T \tilde{x} = 0$.

Unlike in Theorem 3.3, we no longer require nonnegativity of λ. Moreover, the condition $\lambda^T(A\tilde{x} - b) = 0$ does not appear here, because it is trivially satisfied by all feasible solutions $\tilde{x}$.

Proof. (Kuhn-Tucker conditions for equality constraints.) The proof is based on Lemma 3.2 and the simplex method. First, let us assume that $\tilde{x}$ is optimal. From the Lemma we get

$$\nabla f(\tilde{x})\tilde{x} \leq \nabla f(\tilde{x})x,$$

for all feasible x. This means, $\tilde{x}$ is an optimal solution to the linear programming problem

$$\begin{array}{lll} \text{(CP)} & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \\ & & x \geq 0, \end{array} \tag{3.4}$$

with $c^T = \nabla f(\tilde{x})$.

Let $\tilde{X} = (\tilde{X}_1, \ldots, \tilde{X}_n)$ be an optimal basic feasible solution to (3.4), associated with a basis $B \subseteq [n]$. We have $c^T\tilde{x} = c^T\tilde{X}$, but not necessarily $\tilde{x} = \tilde{X}$. The tableau associated with B is

$$\begin{array}{cccccc} x_B & = & \beta & - & \Lambda x_N \\ \hline z & = & z_0 & + & \gamma^T x_N, \end{array}$$

where

$$\gamma^T = c_N^T - y^T A_N, \quad y^T = c_B^T A_B^{-1}, \tag{3.5}$$

with $\gamma \geq 0$ because B was an optimal basis, see (2.21), (2.22).

Now define

$$\begin{array}{rcl} \lambda & := & -y, \\ \mu_i & := & \left\{ \begin{array}{ll} 0 & \text{if } i \in B \\ \gamma_i & \text{if } i \in N \end{array} \right., i = 1, \ldots, n. \end{array}$$

We claim that the vectors λ and μ satisfy the requirements of the Theorem. We already know that $\mu \geq 0$. Moreover, by (3.5),

$$\begin{array}{rcl} c_B^T & = & y^T A_B = y^T A_B + \mu_B^T, \\ c_N^T & = & y^T A_N + \gamma^T = y^T A_N + \mu_N^T, \end{array}$$

so

$$\nabla f(\tilde{x}) = c^T = y^T A + \mu = -\lambda^T A + \mu.$$

Finally, we need to show that $\mu^T\tilde{x} = 0$. For this, we recall that the z-row of the tableau expresses the objective function value $z = c^T x$ in terms of the nonbasic variables. Thus we get

$$\begin{array}{rcl} c^T\tilde{x} & = & z_0 + \gamma^T\tilde{x}_N = z_0 + \mu^T\tilde{x}, \\ c^T\tilde{X} & = & z_0, \end{array}$$

because $\tilde{X}_N = 0$ by definition of a basic feasible solution. From $c^T\tilde{x} = c^T\tilde{X}$, $\mu^T\tilde{x} = 0$ follows.

For the other direction let us assume that vectors λ, μ exist that satisfy the requirements of the Theorem. We claim that in this case $\tilde{x}$ is an optimal solution of the linear problem (3.4). Via Lemma 3.2, this also implies that $\tilde{x}$ is an optimal solution to the original problem (3.3).

To show that $\tilde{x}$ optimally solves (3.4), we prove that the inequality $c^T\tilde{x} \le c^T x$ follows from $Ax = b, x \ge 0$. First note that the latter conditions (using $\mu \ge 0$) imply

$$\begin{aligned} -\lambda^T Ax &= -\lambda^T b, \\ \mu^T x &\ge 0, \end{aligned}$$

so

$$(-\lambda^T A + \mu)x \ge -\lambda^T b. \tag{3.6}$$

This further gives

$$c^T x = \nabla f(\tilde{x})x = (-\lambda^T A + \mu)x \overset{(3.6)}{\ge} -\lambda^T b = -\lambda^T A\tilde{x} + \mu^T\tilde{x} = \nabla f(\tilde{x})\tilde{x} = c^T\tilde{x}.$$

$\sqcup\!\!\!\!\sqcap$

To prove Theorem 3.3, convert the inequalities $Ax \le b$ into equalities by introducing slack variables. Then apply Theorem 3.4.

Finally, here is another Kuhn-Tucker theorem, this time for problems with general equality and inequality constraints.

Theorem 3.5 *(Kuhn-Tucker conditions with general constraints.) Consider the problem*

$$\begin{array}{lll} (CP) & \textit{minimize} & f(x) \\ & \textit{subject to} & Ax = b, \\ & & Cx \le d, \end{array} \tag{3.7}$$

x an n-vector, A an $m_1 \times n$-matrix, b an m_1-vector, C an $m_2 \times n$-matrix, d an m_2-vector, $f : \mathbb{R}^n \mapsto \mathbb{R}$ a convex function. A feasible solution $\tilde{x} \in \mathbb{R}^n$ is optimal for CP if and only if there is an m_1-vector λ and an m_2-vector $\mu \ge 0$ such that

(i) $\nabla f(\tilde{x}) = -\lambda^T A + \mu^T C$,

(ii) $\mu^T(C\tilde{x} - d) = 0$.

Proof. Substitute $x = y - z, y, z \ge 0$ and introduce slack variables to turn the inequalities $Cx \le d$ into equalities. Then apply Theorem 3.4. $\sqcup\!\!\!\!\sqcap$

Chapter 4

Quadratic Programming

In this chapter, we consider *quadratic programming problems*, i.e. problems of the type

$$\begin{array}{lll} \text{(QP)} & \text{minimize} & c^T x + x^T D x \\ & \text{subject to} & Ax = b, \\ & & x \geq 0, \end{array} \tag{4.1}$$

where c is an n-vector, A an $m \times n$-matrix, b an m-vector and D a symmetric, positive semidefinite $n \times n$-matrix, i.e. $x^T D x \geq 0$ holds for all x.

Of course, we would like to apply the machinery of the previous chapter, so we should make sure that such problems are in the class CP.

Lemma 4.1 *$f(x) = c^T x + x^T D x$ is a convex function.*

To solve problem QP, we will try to stick to the simplex method as close as we can. There is, however, one major obstacle. In case of LP, there is always an optimal solution which is a vertex of the feasible region. In QP, this is not necessarily the case. As a trivial example, consider the problem of minimizing the function x^2 subject to the constraints $-1 \leq x \leq 1$. The feasible region is a polytope in 1-space with vertices -1 and 1, but the unique optimal solution occurs in the interior, at the point 0. Therefore, while for LP, the simplex method can restrict its attention to basic feasible solutions, this is not possible for QP; in fact, every feasible solution can appear as the optimal solution for a suitable objective function.

However, with an appropriate generalization of the concept of a basis, we will be back in a finite space of solutions we need to consider. To motivate this concept, let us derive an alternative definition of an LP basis.

4.1 LP Bases Reviewed

Consider the LP in equality form

$$\begin{array}{lll} \text{(LP)} & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \\ & & x \geq 0 \end{array} \tag{4.2}$$

$B \subseteq [n]$ is a basis if and only if there exists a feasible tableau

$$\begin{array}{ccccc} x_B & = & \beta & - & \Lambda x_N \\ \hline z & = & z_0 & + & \gamma^T x_N. \end{array}$$

Via the tableau, the values of the nonbasic variables x_N uniquely determine the values of the basic variables x_B. In particular, the current BFS $\tilde{x}$ is determined by $\tilde{x}_N = 0$. In other words, $\tilde{x}$ is the unique feasible (and therefore optimal) solution to the problem

$$\begin{array}{lll} (\mathrm{LP}(B)) & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \\ & & x_B \geq 0, \\ & & x_N = 0. \end{array} \tag{4.3}$$

Moreover, because $\tilde{x}$ is unique anyway, the constraints $x_B \geq 0$ are redundant, and $\tilde{x}$ is even an optimal solution to the 'unconstrained' problem

$$\begin{array}{lll} (\mathrm{ULP}(B)) & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \\ & & x_N = 0. \end{array} \tag{4.4}$$

In order for the simplex algorithm really to work, we have required the LP to be nondegenerate, and this meant that we do not allow any BFS $\beta = A_B^{-1} b$ to have zero entries. In Subsection 2.2.2, we have achieved this by symbolically perturbing the right-hand side b, with the effect that the following stronger property holds, which we would like to assume for the rest of the manuscript.

Assumption 4.2 *The system $Ax = b, x_N = 0$ has solutions only for sets N with $N \leq n - m$. Equivalently, the system $A_B x_B = b$ has solutions only for sets B with $|B| \geq m$.*

Under this assumption, an LP basis can be defined in yet another way: let $z(G)$ be the optimum value of LP(G), $G \subseteq [n]$ (we set $z(G) := -\infty$ if the problem is unbounded). Let $z_0 \neq -\infty$ be some value. Then we can say that a basis is an inclusion-minimal set B such that $z(B) = z_0$. Considering all possible values $z_0 = z(G), G \subseteq [n]$, we obtain all possible bases B. Why? First of all, nondegeneracy implies $|B| \geq m$, while minimality implies $|B| \leq m$, so B has the right size by this definition. Moreover, A_B must be regular, otherwise the system $A_B x_B = b$ would have at least a one-dimensional solution space, which in particular means that a solution exists with $x_i = 0$ for some $i \in B$. But this contradicts the nondegeneracy.

4.2 Pivoting in Quadratic Programming

To generalize the simplex method to quadratic programming problems (4.1), we start off with a definition of a basis which coincides with the basis concept of LP in case the matrix D vanishes. Throughout this section, it is always useful to see what happens if $D = 0$ and check whether in this case, familiar statements about LP arise (needless to say, they should, otherwise something is wrong).

4.2.1 Bases

Consider the problem

$$\begin{array}{lll} (\mathrm{QP}(B)) & \text{minimize} & c^T x + x^T D x \\ & \text{subject to} & Ax = b, \\ & & x_B \geq 0, \\ & & x_N = 0, \end{array} \tag{4.5}$$

and let $z(B)$ denote the objective function value of an optimal solution of QP(B). (This problem may be unbounded in which case we have $z(B) = -\infty$.)

Definition 4.3 *A* basis *of QP is an inclusion-minimal set B such that $z(B) = z_0$, for some fixed value $z_0 \neq -\infty$.*

A few facts follow from this definition. As before, nondegeneracy implies $|B| \geq m$, but we do not necessarily have $|B| = m$. In fact, $|B|$ can be as large as n. Beyond that, the following statements hold.

Lemma 4.4 *Let B be a basis of QP.*

(i) Any optimal solution $\tilde{x}$ to QP(B) satisfies $\tilde{x}_B > 0$.

(ii) QP(B) has a unique optimal solution $\tilde{x}$.

(iii) $\tilde{x}$ is at the same time an optimal solution to the unconstrained problem

$$\begin{array}{lll} (\mathit{UQP}(B)) & \textit{minimize} & c^T x + x^T D x \\ & \textit{subject to} & Ax = b, \\ & & x_N = 0. \end{array} \tag{4.6}$$

Proof. (i) follows from the minimality of B. To see (iii), apply the Kuhn-Tucker Theorem 3.5 to QP(B). Because of $\tilde{x}_B > 0$, the vector μ in this theorem vanishes. Then, however, $\tilde{x}$ is an optimal solution to UQP(B) by the same Theorem. To prove (ii), let us write out the conditions of Theorem 3.5 explicitly, for problem UQP(B). We get that x is optimal for UQP(B) if and only if an m-vector λ and n-vector μ with $\mu_B = 0$ exists such that

$$\begin{array}{rcl} Ax & = & b, \\ x_N & = & 0, \\ c^T + x^T D & = & -\lambda^T A + \mu^T. \end{array} \tag{4.7}$$

This means, the (x, λ, μ) defining optimal solutions form an affine space L. We claim that this affine space is a single point, proving (ii) (because L in particular contains the optimal solutions to QP(B)) and uniqueness of the vectors λ, μ. First assume, we had optimal solutions $(\tilde{x}, \lambda, \mu)$, $(\tilde{x}', \lambda', \mu')$ with $\tilde{x} \neq \tilde{x}'$, where $\tilde{x}$ is assumed to be any optimal solution to QP(B). Then $\{x \mid (x, \lambda, \mu) \in L\}$ contains a line. Moving from $\tilde{x}$ along this line, we can reach an optimal solution $\tilde{x}''$ to QP(B) with $\tilde{x}''_i = 0$ for some $i \in B$, contradicting the minimality of B. This means, we must have $x = \tilde{x}$ for all $(x, \lambda, \mu) \in L$. Then, however, λ must be unique as well, because the nondegeneracy assumption implies that the rows of A_B are linearly independent (exercise), so that λ is already uniquely computed from the subsystem of equations

$$c_B^T + \tilde{x}_B^T D(B) = -\lambda^T A_B,$$

$D(B)$ being the quadratic submatrix of D correspodning to rows and columns with indices in B. Finally, it also follos that μ is unique. $\square$

This means, just like in LP, a basis B determines a unique optimal solution $\tilde{x}$ to a 'subproblem' QP(B). While for LP, this uniqueness already follows from the uniqueness of a feasible solution, in case of QP we need the additional equations imposed by the Kuhn-Tucker optimality conditions.

4.2.2 Optimality condition

Given an optimal solution $\tilde{x}$ to some basic problem QP(B) which is then also a solution to UQP(B), we would – like in the LP case – like to find a variable $x_j, j \notin B$ whose increase would lead to a smaller objective function value. Equivalently, we would like to find an index j such that $z(B \cup \{j\}) < z(B)$ or certify that no such index exists. The following Lemma characterizes the set of improving indices.

Lemma 4.5 *Let B be a basis of QP with associated optimal solution $\tilde{x}$ and vectors λ, μ, $\mu_B = 0$ such that*

$$c^T + \tilde{x}^T D = -\lambda^T A + \mu^T. \tag{4.8}$$

(By the previous lemma, $\tilde{x}, \lambda$ and μ are unique.) Then $z(B \cup \{j\}) < z(B)$ holds if and only if $\mu_j < 0$.

Proof. QP($B \cup \{j\}$) differs from QP(B) only by the fact that the equality $x_j = 0$ has been replaced by the inequality $x_j \geq 0$. In the corresponding Kuhn-Tucker conditions according to Theorem 3.5, this replaces the previously unrestricted variable μ_j by a restricted variable $\mu_j \geq 0$. It follows that $\tilde{x}$ is optimal for QP($B \cup \{j\}$) if and only if $\mu_j \geq 0$ really holds. The Lemma follows. $\square$

Note that if $\mu_j \geq 0$ for all $j \notin B$, then $\tilde{x}$ is an optimal solution to the whole problem.

4.2.3 The Pivot Step

How do we actually find the improved solution $z(B \cup \{j\})$ and a corresponding basis B' ? In LP, this was easy, we just needed to increase x_j until some other variable x_i got down to zero. At that point, $B' = B \cup \{j\} - \{i\}$ is the desired new basis with $z(B') = z(B \cup \{j\})$. In case of QP, it's not that simple, because of three reasons.

(i) What does 'increasing x_j' mean? In case of LP, a unique feasible solution to the problem LP($B \cup \{j\}$) is associated with each possible value of x_j (and these solutions taken together form an edge of the feasible region which is traversed as x_j increases). In case of QP(B), many feasible solutions may correspond to a fixed value of x_j.

(ii) Although it is *feasible* to increase x_j until some $x_i, i \in B$ goes down to zero, it may not be *profitable.* In other words, the objective function value might reach a minimum before any other constraints become binding. In case of a linear objective function, of course, this can not happen.

(iii) Even if x_j profitably increases until some basic variable reaches zero, the feasible solution we have at this point is not necessarily the desired optimal solution to QP($B \cup \{j\}$). In LP we are only allowed to argue in that way because x_j cannot be increased further without violating feasibility. In QP, it may well be possible to 'change' the direction in which we move our solution (because this direction is in general not unique, see item (i)) and still keep improving the objective function further without leaving the feasible region.

Item (i) can easily be dealt with if we consider for fixed value t of x_j the unique *optimal* solution of the problem QP($B \cup \{j\}$) subject to the additional constraint $x_j = t$. Why is this solution unique and how do we get it?

Let $\tilde{x}$ be the optimal solution to QP(B) and UQP(B) and rewrite the optimality conditions (4.7) in matrix form. We get that $\tilde{x}$ is obtained from the uniquely solvable system of equations

$$\begin{pmatrix} D & A^T & -E(N)^T \\ A & 0 & 0 \\ -E(N) & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} -c \\ b \\ 0 \end{pmatrix}, \tag{4.9}$$

$E(N)$ the $n \times n$-matrix whose j-th column E_j satisfies

$$E_j := \begin{cases} \mathbf{e}_j, & \text{if } j \in N, \\ 0, & \text{if } j \notin N \end{cases}$$

Now define $\hat{B} := B \cup \{j\}$ and consider the problem

$$\begin{array}{lll} \mathrm{UQP}_{\{x_j=t\}}(\hat{B}) & \text{minimize} & c^T x + x^T D x \\ & \text{subject to} & Ax = b, \\ & & x_{N-\{j\}} = 0, \\ & & x_j = t, \end{array} \tag{4.10}$$

which arises from UQP(B) by replacing equality $x_j = 0$ with $x_j = t$. The Kuhn-Tucker conditions do not change at all, and we get that the optimal solution $\tilde{x}$ to (4.10) is obtained by solving the system

$$\begin{pmatrix} D & A^T & -E(N)^T \\ A & 0 & 0 \\ -E(N) & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} -c \\ b \\ -t\mathbf{e}_j \end{pmatrix} \tag{4.11}$$

for x, λ, μ. If (4.9) is uniquely solvable, so is (4.11) because both systems only differ in their right-hand side. As long as the solution $\tilde{x}$ of (4.11) is feasible for QP($\hat{B}$), it is also an optimal solution to

$$\begin{array}{lll} \mathrm{QP}_{\{x_j=t\}}(\hat{B}) & \text{minimize} & c^T x + x^T D x \\ & \text{subject to} & Ax = b, \\ & & x_B \geq 0, \\ & & x_{N-\{j\}} = 0, \\ & & x_j = t, \end{array} \tag{4.12}$$

Now we have – just like in LP – the situation that a unique point $\tilde{x}(t)$ is associated to each value $x_j = t$.

Let's consider item (ii) above. How large should we let x_j get? In LP, our only worry was that at some point, the solution $\tilde{x}(t)$ becomes infeasible, and that's where we had to stop. Of course, then we also have to stop here, and computing for each variable $x_i, i \in B$ the value t_i at which it becomes zero, is just a classical case of the *ratio test* as we had it before in LP (see Subsection 2.3.2).

But we may have to stop earlier, namely when the objective function reaches a minimum before any basic variable has become zero. How do we notice this? That's easy: as soon as we get $\mu_j = 0$ from (4.11), we know that the current solution is optimal not only for $\mathrm{QP}_{\{x_j=t\}}(\hat{B})$ but also for QP($\hat{B}$). Namely, in this case the Kuhn-Tucker conditions imply that the current solution is optimal for the problem

$$\begin{array}{ll} \text{minimize} & c^T x + x^T D x \\ \text{subject to} & Ax = b, \\ & x_B \geq 0, \\ & x_{N-\{j\}} = 0, \end{array}$$

and because we know that $x_j \geq 0$ also holds, we have the desired optimum for QP($\hat{B}$). Moreover, $\hat{B}$ is a basis (exercise). Finding the time t_j at which μ_j becomes zero is just as easy as finding the times t_i before.

Concerning item (iii), we still need to discuss what happens at the point where the current solution $\tilde{x}(t)$ is about to become infeasible for QP($\hat{B}$), satisfying $\tilde{x}_i(t) = 0$ for some $i \in B$. Then $\tilde{x}$ is not only the unique optimal solution to the problem $\mathrm{QP}_{\{x_j=t\}}(\hat{B})$ that we currently handle, but also the unique optimal solution for $\mathrm{QP}_{\{x_j=t\}}(\hat{B} \setminus \{i\})$. If we just update $\hat{B}$ to $\hat{B} \setminus \{i\}$, we are back in the situation where we have problem (4.10) and a unique optimal solution to it. Then we just repeat the preceding steps. Because $\hat{B}$ gets smaller in every such 'subiteration', we must sooner or later end up in the situation that μ_j becomes zero, in which case we have found our new basis, and the pivot step is finished.

4.3 Size of a Basis

In an implementation, we really need to solve the linear system (4.11), but in the form we have written it down, it is a system in $2n+m$ equations in $2n+m$ unknowns, which is not feasible to solve for large n. In contrast, for LP, the linear systems we needed to solve were of order $m \times m$. In particular, their sizes are independent of n. Luckily, we do not need to solve the large system (4.11), there is an equivalent smaller one. To see this, we rewrite problem $\mathrm{UQP}_{\{x_j=t\}}(\hat{B})$ (whose optimal solution is characterized by (4.11)) as a problem over only $|\hat{B}|$ variables, namely as

$$\begin{array}{lll} \mathrm{UQP}_{\{x_j=t\}}(\hat{B}) & \text{minimize} & c_{\hat{B}}^T x_{\hat{B}} + x_{\hat{B}}^T D(B) x_{\hat{B}} \\ & \text{subject to} & A_{\hat{B}} x_{\hat{B}} = b, \\ & & x_j = t. \end{array} \qquad (4.13)$$

Considering the Kuhn-Tucker conditions for this problem, we get a system in only $|\hat{B}| + m + 1$ variables. This means, if $\hat{B}$ is sufficiently small, we deal with small systems again. Since $|\hat{B}|$ is always at most $|B|+1$, B the basis we have started with, we would like to have a bound on the maximum basis size. As the next Lemma shows, this bound depends on the rank of D.

Lemma 4.6 *Let $\tilde{x}$ be an optimal solution to (QP). If r denotes the rank of D, then an optimal solution $\tilde{x}'$ to (CP) exists such that*

$$|\{j : \tilde{x}'_j > 0\}| \leq m + r.$$

In case of LP, i.e. if D is the zero matrix of rank 0, we recover the statement that the optimal solution is a BFS, w.l.o.g. For QP it follows that any basis has size at most $m+r$.
Proof. By Theorem 3.5, there are vectors λ, μ such that

$$c^T + \tilde{x}^T D = -\lambda^T A + \mu.$$

It follows that any y in the affine space

$$L = \{y \mid y^T D = x^T D\} \cap \{y \mid Ay = b\}$$

is also optimal. We have

$$\begin{array}{rcl} \dim(L) & \geq & \dim(\{y \mid y^T D = x^T D\}) + \dim(\{y \mid Ay = b\}) - n \\ & \geq & (n-r) + (n-m) - n \\ & = & n - m - r. \end{array}$$

This means, the set of optimal solutions contains the $n-m-r$-dimensional polytope $L \cap \{x \mid x \geq 0\}$. A vertex $\tilde{x}'$ of this polytope has the required property. $\square$

Chapter 5

Some Applications

In this final chapter we come back to the examples of geometric optimization problems introduced back in the first section of the first chapter, and see how the previously developed algorithms can efficiently be applied to them.

Look at the *Largest ball in polyhedron* problem (LBIP) as defined in (1.1). In this formulation, the problem is LP with n inequality constraints but only $d+1$ variables. Recall that our scenario was that d is small but n quite large. If we would just apply the simplex method to this problem, we would introduce n slack variables to turn the inequalities into equalities; subsequently we would then work with bases B of size n, involving the computations of inverses A_B^{-1} of $n \times n$-matrices. Computing the inverse takes $O(n^3)$ time if done with Gaussian elimination, and this is just too much.

We would rather have the problem the other way round, featuring many variables but only few constraints. Then the simplex method would work with small basis matrices and can be expected to be efficient.

Luckily, there is something called *duality* in linear programming. Consider the problem

$$\begin{array}{lll} \text{(LP)} & \text{minimize} & c^T x \\ & \text{subject to} & Ax \leq b. \end{array} \tag{5.1}$$

(This is the situation in (LBIP)). One can prove that if (LP) has an optimal solution, then also the problem

$$\begin{array}{lll} \text{(LP')} & \text{minimize} & b^T y \\ & \text{subject to} & A^T y = -c, \\ & & y \geq 0 \end{array} \tag{5.2}$$

has an optimal solution, and the optimal solutions coincide. Moreover, from an optimal BFS to (5.2), an optimal BFS to (5.1) can easily be reconstructed. In case of (LBIP), (5.2) has only $d+1$ equality constraints (and n variables); it can therefore effciently be handled using the simplex method, i.e. usually in time $O(n)$ if d is assumed to be a constant, see Section 2.4.

The LP arising from the *Smallest enclosing annulus* problem (area version) is also transformed to a problem with few constraints, using the same kind of duality, again leading to an algorithm which we expect to be an $O(n)$ algorithm.

There are several versions of the duality theorem, depending on whether the problem has equality or inequality constraints, nonnegativity restrictions or not etc., but all these versions are essentially equivalent.

The version which is easiest to prove using the material of this manuscript is probably the following.

Consider the problems

$$\begin{array}{lll} \text{(LP)} & \text{minimize} & c^T x \\ & \text{subject to} & Ax \leq b, \\ & & x \geq 0 \end{array} \tag{5.3}$$

and

$$\begin{array}{lll} \text{(LP')} & \text{maximize} & b^T y \\ & \text{subject to} & A^T y \leq c, \\ & & y \leq 0. \end{array} \tag{5.4}$$

If (5.3) has an optimal solution, so has (5.4), and both optimal values coincide.

The *Distance of polytopes* problem (DOP) as defined in (1.1) is directly solvable within our framework in an efficient manner, because it only has $2d+2$ equality constraints. Moreover, the objective function was

$$(p-q)^T(p-q) = (p-q)^T \begin{pmatrix} E & -E \\ -E & E \end{pmatrix} (p-q),$$

which is a positive definite matrix; consequently, the matrix D in the objective function has only rank $2d$ (recall that this was important to have only small systems of equations to solve). As for the simplex method itself, the variant for quadratic programming is also expected to be fast in practice, if at each stage, the variable x_j is entered which has the smallest value of μ_j, see Subsection 4.2.2. As for the problems before, we expect an $O(n)$ algorithm for (DOP) in case the dimension is constant.

Finally, we have the *Smallest enclosing ball* problem. Why is this a quadratic programming problem? Let $p_1, \ldots, p_n$ be the input points. Now we claim that the problem can be formulated in the form

$$\begin{array}{lll} \text{(SEB)} & \text{minimize} & p^T p - p_i^T p_i x_i, \\ & \text{subject to} & p = \sum_{i=1}^n x_i p_i, \\ & & \sum_{i=1}^n x_i = 1, \\ & & x \geq 0, \end{array} \tag{5.5}$$

and the optimal point p is the center of the smallest enclosing ball of the points. To prove this is the 'master exercise'.

Problem (SEB) has $d+1$ equality constraints, and the rank of the matrix D in the objective function – when formulated as QP – is d. Thus, the problem can efficiently be solved using the adaptation of the simplex method for quadratic programming.