Lecture Notes LP: Lyon 2026

Sophie Huiberts

2026-09-14

We will cover linear programming from a theoretical perspective, with a focus on the simplex method. Topics will include:

- What are linear programs, what are they good for, what are they bad for
- Fundamental concepts such as duality
- Convex polyhedra, their combinatorics and geometry
- The simplex method (theoretically)
- Why the simplex method is slow
- Why the simplex method is fast
- If we have time: the simplex method (practically)

Despite the linear programming and the simplex method predating modern computers, this is still an active area of research. There are many algorithms that work 'better' than theory can currently explain. The simplex method is one of those, although it is perhaps the best understood among the poorly understood algorithms.

0.1. Promises about notation

Sets of integers are denoted $[k] = \{1, .., k\}$.

We will often need to index into matrices or vectors. This is done using subscripts. For a vector $b \in \mathbb{R}^n$ and index $i \in [n]$, the i'th entry of b is denoted as $b_i \in \mathbb{R}$. Similarly, if $S \subset [n]$ then we denote the restricted vector by $b_S \in \mathbb{R}^S$.

When we index into a matrix, we only ever select rows (selecting columns would be ridiculous). That is, when $A \in \mathbb{R}^{n \times d}$ and $i \in [n]$ then $A_i \in \mathbb{R}^d$ is the i'th row of A. Similarly, if $S \subset [n]$ then we denote the submatrix by $A_S \in \mathbb{R}^{S \times [n]}$.

For those who have not seen this notation before. When we have sets K, L then K^L denotes the set of functions from L to K. Hence if k is a natrual number and $h \in \mathbb{R}^{[k]}$, then k is a function that assigns to each index $i \in [k]$ a real number in $\mathbb{R}$. In other words, h is a vector. When we write a real vector space as $\mathbb{R}^k$, then that is effecitvely a shorthand for $\mathbb{R}^{[k]}$. Similarly, $\mathbb{R}^{n \times d}$ we can think of as shorthand for $\mathbb{R}^{[n] \times [k]}$.

There may be times when you are tempted to believe that $n \geq d$ implies that $[d] \subset [n]$. Resist this temptation; the consequent is syntactically correct but semantically worse than useless.

1. Introduction

When we talk about a linear program, we are talking about any problem that can be written in the following form:

$$\begin{aligned} &\text{maximize } c^T x \\ &\text{subject to } Ax \leq b. \end{aligned} \tag{1}$$

The program is specificied by the *constraint matrix* $A \in \mathbb{R}^{n \times d}$, the *right-hand side vector* $b \in \mathbb{R}^n$ and the *objective vector* $c \in \mathbb{R}^d$. A *feasible solution* to the program is any vector $x \in \mathbb{R}^d$ that satisfies the system of coordinate-wise linear inequalities $Ax \leq b$. That is, if we denote the rows of A as row vectors $A_1, A_2, ..., A_n$, then x must satisfy $A_i x \leq b_i$ for all $i = 1, ..., n$. No feasible solution need exist.

An *optimal feasible solution* is any feasible solution $x \in \mathbb{R}^d$ such that for any other feasible solution $x' \in \mathbb{R}^d$ we have $c^T x \geq c^T x'$. That is, among all feasible solutions, x should have an *objective value* that is the highest possible. No optimal feasible solution need exist.

The input data of the LP is A, b, c, and the computational task described by this data is to find an optimal feasible solution x or prove that none exists.

1.1. Graphical example

We drew an LP in 2 variables on the board. The feasible set is the intersection of a number of (affine) half-spaces. The result is a convex polyhedron (in 2 variables, a polygon). An optimal feasible solution is a point that is the furthest possible in the direction of the objective vector. In 3 variables we drew a three-dimensional cube as a feasible region. In either case, we agree that (under all reasonable circumstances) if there is an optimal feasible solution then there is one that is also a vertex of the feasible polyhedron.

1.2. LP on the computer

Most LPs are there to be solved. A user writes (a program to construct) the input data A, b, c in the computer, and a dedicated software library is called to return an optimal feasible solution x. In this role, LP is an important piece of infrastructure in industrial societies. It is used to design and operate national power systems, in forest management, in petroleum refinery, and when cutting paper, metal or glass. In biology, you can find LP in flux balance analysis. In data science, LP underlies least absolute error regression. If you manufacture a product, you may use LP when moving products from factories to warehouses. And on your own computer and phone, LP is used when drawing windows and when stabilizing video. LP is also an essential part of the dominant paradigms for solving mixed integer (non-)linear mixed integer programming problems, a technology with even broader applications.

In all these applications, the simplex method is an essential part of all available software packages. Sophisticated software may include additional algorithms for LP on top of a simplex method, such as an interior-point method or a first-order method.

1.3. LP as mathematical objects

LPs can be solved, but they are also mathematical objects with rich structure that can be studied in their own right. In this course we will see some of this structure. This will include how to certify whether a feasible solution is optimal and how to certify whether or not any feasible solution exists. The structure of LP solutions (strong duality, complementary slackness) is a useful feature in numerous mathematical proofs.

The various algorithms for LP motivate mathematical questions as well.

- For the simplex method, we will see in this course a number of questions in convex geometry and in polyhedral combinatorics.
- In algorithm theory, it is known that LPs can be solved in weakly polynomial time. This was first proven for the ellipsoid method, a theoretical algorithm that works very poorly in practice. It is unknown whether LPs can be solved in strongly polynomial time.
- The other well-known class of weakly polynomial time algorithms are the interior-point methods. These can be competitive in practice with simplex methods. IPMs have a similarly rich theory as simplex methods do.

1.4. We Don't Talk About Standard Form

Other textbooks and resources may consider other forms of LP. This includes what is called "standard form"

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax = b \\ &\qquad\qquad x \geq 0, \end{aligned} \tag{2}$$

and the similarly non-descriptively named "canonical form"

$$\begin{aligned} &\text{maximize } c^T x \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad x \geq 0. \end{aligned} \tag{3}$$

Please convince yourself that any optimization problem written in either of these forms can be translated into one formulated in the inequality form (1) used in these notes, and hence that we can restrict our focus to only the form (1) without loss of generality.

1.5. Chapter End Notes

Linear programming has little to do with computer programming. Instead, the origin of the term 'program' lies with schedules and such. Think of an event programme, or think of television programming.

Although the LP concepts we discuss are universal, every textbook and lecture note uses different notation, different standards, and different words. This includes using the same words for slightly different concepts. This is an unfortunate situation, but there is little to be done about it at this point. We are carrying almost 80 year so historical baggage with us, and any attempt to fix it would be an instance of xkcd 927.

If you wish to refer to other sources besides these lecture notes, please **avoid Wikipedia on this topic**. Specifically avoid the pages on *linear programming*, *simplex method* and *revised simplex method*. They are poorly written and contain factual errors. I would hate for you to get confused by those.

2. The simplex method

The simplex method is a combinatorial algorithm which operates on the feasible set. We will develop the simplex method by considering first the relevant geometric concepts, and then talk through an example. The full algorithm is done afterwards. In the next lecture we prove that the simplex method terminates with an optimal solution.

2.1. Convex Polyhedra

The feasible set is a convex polyhedron. Convex: when you have two feasible points, then every point in the line segment connecting these two points is also feasible.

Polyhedron: you know that this is. Given a vector $a \in \mathbb{R}^d$ and a scalar $y \in \mathbb{R}$, the space $\{x \in \mathbb{R}^d : a^T x = y\}$ is called a hyperplane.[1] A set of the form $\{x \in \mathbb{R}^d : a^T x \leq y\}$ is called a halfspace. A set is a polyhedron if it can be written as the intersection of finitely many halfspaces.

Polyhedra have *faces*. If $P \subset \mathbb{R}^d$ is a polyhedron and $F \subset P$ then we say F is a face if, for every $x, x' \in P$ and every $\lambda \in (0,1)$, we have that $\lambda x + (1-\lambda)x' \in F$ implies $x, x' \in F$. Given an

[1] Based on your education before this, you may know this as an *affine* hyperplane and reserve the unqualified term for cases with $y = 0$. For our purposes that distinction makes little sense.

inequality description $P = \{x \in \mathbb{R}^d : Ax \leq b\}$, for every face $F \subset P$ there is an index set $S \subset [n]$ such that $F = \{x \in P : A_S x = b_S\}$. In the other direction, every subset of constraints will isolate a face.

Every face has a dimension. The polyhedron P is its own d-dimensional face. If you were imagining a P which is not d-dimensional, good job but in this course we will not bother with these edge cases. A $d-1$-dimensional face is called a facet. A 1-dimensional face is called an edge. A 0-dimensional face is called a vertex.

2.2. Development from Graphical Example

We assume we start with a *basis* of the LP. That is a set of indices $B \subset [n]$ of cardinality $|B| = d$ for which the square matrix A_B is invertible. From a basis we can extract the corresponding *basic solution* as $x^B = A_B^{-1} b_B$. We additionally assume that the basis is feasible. That is, we assume the basic solution satisfies $Ax_B \leq b$. These assumptions are not without loss of generality, but we will see later that they are also no big deal.

Finally, we assume that our current basic solution is not optimal. Otherwise we would already be finished. The simplex method works by changing its current basis into a better feasible basis, repeatedly until an optimal solution is reached.

Every vertex of the feasible set is a basic feasible solution: for any vertex $x' \in P$ there exists a basis $B \subset [n]$ with $x' = x^B$. Every facet of the feasible set corresponds with a linear constraint, a row from $Ax \leq b$.

By way of graphical example, let us take a cube. The starting basis consists of the indices for the constraints creating the bottom, front-left and front-right facets. The basic feasible solution is the vertex where these facets meet.

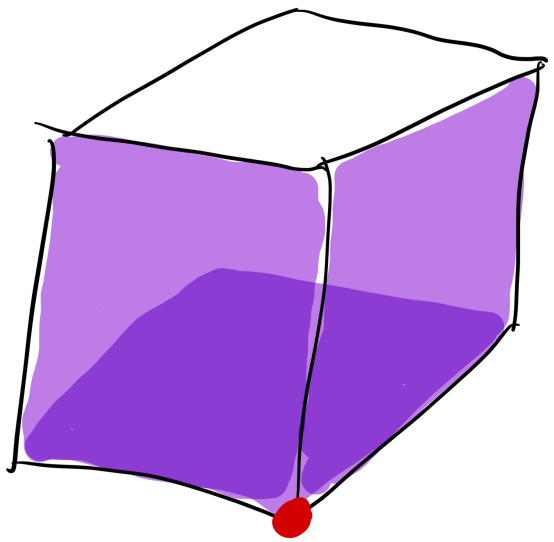

Figure 1: Books cover

We wish to improve our basis and obtain a new vertex. Geometrically, we do this by leaving one facet, traversing along the edges where all remaining facets meet, and ending in the vertex on the opposite side of the edge. We change our basis by removing the index for the facet we left and adding the index for the facet we now touch. For the graphical example, let us move away from the bottom face and along the vertical edge. To accomplish this, let us first describe the direction in which we move.

We wish to find a vector $v \in \mathbb{R}^d$ such that the edge is part of the half-line $x_B + \lambda v, \lambda \geq 0$. Let $p \in B$ denote the index of the constraint creating the bottom facet. Then the vector $v \in \mathbb{R}^d$ must satisfy the following properties:

- $A_i v = 0$ for every $i \in B$ with $i \neq p$. This will ensure that $A_i(x_B + \lambda v) = A_i x_B = b_i$, so that the line segment stays inside the facets that it should stay inside.
- $A_p v = -1$ (or your favorite negative number of choice) will ensure $A_p(X_B + \lambda v) < b_p$, so that when we move in the direction of v then we leave the bottom facet p.

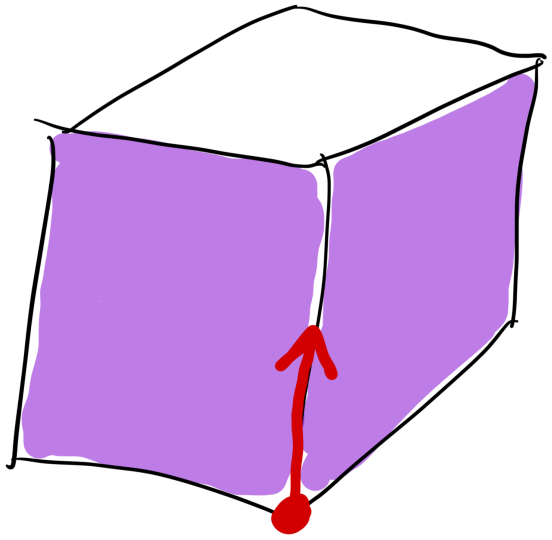

Figure 2: Books cover

We can write the requirements for v more compactly as $A_B v = -e_p$, where $e_p \in R^B$ is the unit vector with a 1 on index p and zeroes at all other indices. One can solve this linear system as $v = -A_B^{-1} e_p$, meaning that v is the p-index column of the basis inverse matrix A_B^{-1}.

Now that we have determined the direction v, the next step is to compute the value of $\lambda \geq 0$ for which $x_B + \lambda v$ is the other endpoint of our chosen edge. For this, we must determine the largest value possible value λ for which the moved point is feasible $A(x_B + \lambda v) \leq b$. That is, we need $A_i(x_B + \lambda v) \leq b_i$ for every $i \in [n]$. Among these n inequalities, we need only consider those for which $A_i v > 0$. For those, we can rewrite the previous inequality to state

$$\lambda \leq \frac{b_i - A_i x_B}{A_i v} \quad \text{for all} \quad i \in [n] \quad \text{with} \quad A_i v > 0. \tag{4}$$

For the i with $A_i v \leq 0$, the corresponding inequality does not yield an upper bound on λ. Let λ^* denote the largest value for which all inequalities (4) hold.

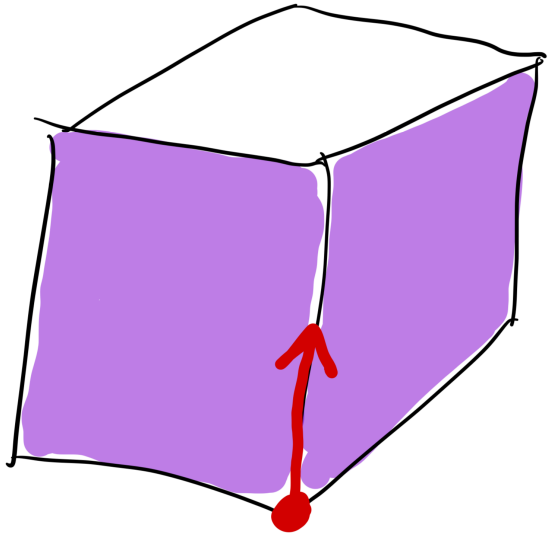

Figure 3: Books cover

Our next vertex of the feasible set is the point $x_B + \lambda v$ as calculated above. The constraint that will enter the basis, is the constraint $q \in [n] \setminus B$ for which (4) tight: $\lambda^* = \frac{b_q - A_q x_B}{A_q v}$.

To perform the simplex method, update $B \leftarrow B \setminus \{p\} \cup \{q\}$. Repeat until you are satisfied.

2.3. Full Algorithm

The simplex method keeps doing iterations[2] until no more improving moves are possible. That is, when there is no more possible choice of p for which $v = -A_B^{-1} e_p$ satisfies $c^T v > 0$, i.e., for which the next vertex found would have better objective value. That is, the method keeps going while there exists any i for which $\left(c^T A_B^{-1}\right)_i < 0$, and stops as soon as $c^T A_B^{-1} \geq 0$.

With that, we have all the pieces necessary to describe a simplex method.

SIMPLEX METHOD(A, b, c, B):

while $c^T A_B^{-1} \not\geq 0$:
  solve $A_B x^B = b_B$ for x^B
  choose $p \in B$ with $\left(c^T A_B^{-1}\right)_p < 0$ // This choice is called the 'pivot rule'
  solve $A_B v = -e_p$ for v
  set $\lambda^* = \min_{\substack{i \in [n] \setminus B \\ A_i v > 0}} \frac{b_i - A_i x^B}{A_i v}$ // This line and the next are the 'ratio test'
  choose $q \in [n] \setminus B$ with $\lambda^* = \frac{b_q - A_q x^B}{A_q v}$
  update $B \leftarrow B \setminus \{p\} \cup \{q\}$
return B.

The objective value at a basis B is $c^T x^B$, and the objective value at the next iteration will be $c^T(x_B + \lambda^* v)$. Hence, the increase in objective value is equal to $\lambda^* c^T v = -\lambda^* \cdot c^T A_B^{-1} e_p$. The term $-c^T A_B^{-1} e_p$ is for historical reasons called the *reduced cost*, and λ^* is called the *step size*. The objective increase is the product of the reduced cost and the step size.

[2] or linear algebraic reasons, these iterations are called "pivot steps"

2.3.1. But What If $\lambda^* = \infty$

This situation can happen when every $i \in [n] \setminus B$ satisfies $A_i v \leq 0$. In that case, the ray $x_B + \lambda v$ with $\lambda \in [0, \infty)$ certifies that the LP has no optimal solution. The algorithm may return 'unbounded'.

2.3.2. But What If $\lambda^* = 0$

If $\lambda^* = 0$ then that would mean we make a basis exchange $B' = B \setminus \{p\} \cup \{q\}$ without changing the current vertex $x^B = x^{B'}$. This is a situation that in theory is called *degeneracy*.

Degeneracy is a complex topic, whose definitions, causes, and implications are vastly different in theory and in practice. I am not aware of any comprehensive academic publications describing degeneracy in practice, and I do not respect the theoretical approaches to it. Thus, for this course, we will act as if **degeneracy does not exist**.

2.3.3. But What If q Is Not Unique

It theoretically could happen that there are multiple indices $q, q' \in [n] \setminus B$ wit h$\lambda^* = \frac{b_q - A_q x^B}{A_q v} = \frac{b_{q'} - A_{q'} x^B}{A_{q'} v}$. However, if that were to occur then in the next iteration we could have a situation with $\lambda^* = 0$. Thus, q not being unique qualifies as degeneracy. See the previous subsubsection why degeneracy does not exist.

In high-performance codes, choosing q correctly is important. For this purpose, one uses variations on *Harris' ratio test*. Experts broadly agree that Harris' ratio test is mandatory, but they disagree on *why* it is beneficial.

2.3.4. But How Do I Choose p

There is a freedom to choose any $p \in B$ which satisfies $\left(c^T A_B^{-1}\right)_p < 0$. The rule by which you choose p is called the *pivot rule*, and multiple pivot rules have been described over the decades.

- Dantzig, the originator of the simplex method, chose the $p \in B$ for which $\left(c^T A_B^{-1}\right)_p$ is minimal. Because he worked in the 'standard equality form' of LP, there were good reasons to call this the *most negative reduced cost pivot rule*.
- If alphabetical ordering is your favorite, you could consider choosing the smallest index $p \in B \subset [n]$ for which $\left(c^T A_B^{-1}\right)_p < 0$. This is called *Bland's least index rule* because Bland used this rule to study the simplex method under degeneracy (which, I remind you, does not exist).
- If you wish to maximize the amount of progress you make per pivot step, you could look ahead and choose the p for which the objective increase $-\lambda^* \cdot c^T A_B^{-1} e_p$ is maximized. This works to minimize the number of pivot steps but is very expensive computationally, since you must execute the ratio test once for every possible $p \in B$ in each iteration. This is called the *maximum increase pivot rule*.
- The geometrically cleanest choice is the *steepest edge pivot rule*. In this pivot rule we consider the normalized objective improvement per unit of movement $\frac{-c^T A_B^{-1} e_p}{\| -A_B^{-1} e_p \|}$.

In high-performance codes, you usually find variations on the steepest edge pivot rule. Assorted changes can be made to it in order to speed up its computation and make it more numerically stable. [Goldfarb's update rule, Harris' approximate steepest edge rule "devex", partial pricing, multiple pricing]

2.3.5. But Is This Termination Criterion Sufficient?

This is an excellent question: just because none of the current vertex' outgoing edges gives an improving move (local optimality), is that sufficient to conclude that we have found an optimal solution (global optimality)? We will prove this in the next chapter.

2.3.6. But What If I Do Not Have A Feasible Basis To Start From

We can resolve this situation by writing a *two-phase simplex algorithm*. Pick your favorite basis $B \subset [n]$, not necessarily feasible. Now write down the following *phase one* LP

$$\begin{aligned} \text{minimize } & s \\ \text{subject to } & Ax \leq b + s1_{[n]\setminus B} \\ & s \geq 0. \end{aligned} \tag{5}$$

where $1_{[n]\setminus B} \in \mathbb{R}^n$ indicates the vector all whose entries in B have value 0 and all entries in $[n] \setminus B$ have value 1. Compute $s^B \in \mathbb{R}$ as the minimum value of s^B for which $Ax^B \leq b + s^B 1_{[n]\setminus B}$. Generically, there is a single index $i \in [n] \setminus B$ for which $A_i x^B = b_i + s^B$.

If $s^B \leq 0$ then B was already feasible for (1), so the interesting case is to assume that $s^B > 0$. The index set $B \cup \{i\}$ is a feasible basis for (5). (Convince yourself that this is a basis, and that it is feasible). You can use the simplex method on this linear program (5) to find a feasible basis for (5) with minimum possible value of s. If that minimum value is 0, then among our basis of $d+1$ elements we can find a subset of d indices which create a feasible basis for (1). (Convince yourself of this). If the minimum value is strictly greater than 0, then we have proven that (1) admits no feasible solutions. The optimal basis of the phase one LP (5) will indicate a set of $d+1$ constraints which can not be satisfied simultaneously in the phase two LP (1).

2.3.7. But What Is The Running Time

In theory, each iteration can be completed in polynomial time. Hence when theorists talk about the running time of the simplex method, they are interested in the number of pivot steps required before the algorithm terminates. Decades of computational experience indicate that this number of pivot steps grows roughly linear with the problem size $d+n$. During this course, we will look at a number of theoretical investigations into this question.

In practice, the running time of the simplex method depends on your number of memory accesses. Unlike the vast majority of algorithms, the simplex method is *memory-bound*. Mostly your CPU idles while it waits for new data to arrive from RAM. This is partly why the ratio test is so expensive: it requires looking at the entire constraint matrix.

In real applications of LP, the constraint matrix is very sparse, with as little as 1% to 0.01% of entries being non-zero. Good implementations of the simplex method are aware of that sparsity and make good use of it.

3. LP Duality and Complementary Slackness

Consider an apple falling into a polyhedral fruit bowl. Under the influence of gravity, the apple finds its way to the lowest point it can reach. The apple has solved an optimization problem: it has maximized its position in the downwards direction, subject to being inside the bowl.

When the apple has reached the lowest point, it will stay there at rest. The apple is no longer moving, which means that its acceleration is 0. From Newton's principle of $F = ma$, this implies that the net force acting on the apple is 0.

Two types of forces are acting on the apple. Gravity is pulling it downwards, and the normal forces from the bowl's walls are pushing the apple inwards into the bowl. Since the net force is 0, that means these forces exactly cancel each other out.

We write the apple-in-the-bowl problem as a linear program of the form (1). The objective vector c points down towards the ground. For every flat face of the fruit bowl, there is a corresponding inequality of the form $A_i x \leq b_i$. The apple lands in a location x^* which, subject to these inequality constraints, is the furthest possible in the downward direction. In other words, it maximizes $c^T x$ subject to $Ax \leq b$.

When the apple is at location x^*, it will touch some of the walls of the bowl. Let $S \subset [n]$ denote the indices of the corresponding constraints. That is, we have $S = \{i \in [n] : A_i x^* = b_i\}$.

Note that the constraint row vector A_i is an outward facing normal vector to the bowl wall. Because of that, we can write any inward-facing normal vector in our physics problem as $-y_i A_i$ for some $y_i \in \mathbb{R}$ with $y_i \geq 0$.

Newton's balance of forces, then, tells us the existence of a vector $y^* \geq 0$ such that

$$c^T + \sum_{i \in [n]} -y_i^* A_i = 0. \tag{6}$$

In more compact notation, this vector y^* satisfies the relation $A^T y^* = c$.

With all this in place, it is hopefully not a stretch to define the dual linear program. Its feasible set consists of all vectors that satisfy the specification we currently have of y.

Definition 1 (*Dual LP*)**.** *The original linear program (1) is called the* primal *LP. The corresponding* dual *LP is*

$$\begin{aligned} &\text{minimize } b^T y \\ &\text{subject to } A^T y = c \\ &\qquad\qquad y \geq 0 \end{aligned} \tag{7}$$

The dual linear program is an object of remarkable importance. It allows us to prove bounds on the optimal value of the primal LP (*the weak duality theorem*). We can find the optimal value of the primal LP by instead solving the dual LP (*the strong duality theorem*). Moreover, an optimal solution to one can tell us the structure of the optimal solutions to the other (*complementary slackness*).

Theorem 2 (*Weak duality*)**.** *Let $x \in \mathbb{R}^d$ be feasible for (1) and let $y \in \mathbb{R}^n$ be feasible for (7). Then $c^T x \leq b^T y$.*

Proof. Since $c = A^T y$ we can substitute $c^T x = y^T A x = \sum_{i \in [n]} y_i A_i x$. For every summand here, we know that $y_i \geq 0$ and $A_i x \leq b_i$. Hence $y_i A_i x \leq y_i b_i$ for every $i \in [n]$, and it follows that $\sum_{i \in [n]} y_i A_i x \leq y^T b$. □

Theorem 3 (*Strong duality*)**.** *Let $x^* \in \mathbb{R}^d$ be an optimal feasible solution for (1) and let $y^* \in \mathbb{R}^n$ be an optimal feasible solution for (7). Then $c^T x^* = b^T y^*$.*

Proof. The proof of strong duality requires the use of the following lemma, which we will not prove (but which does have an excellent Wikipedia page).

Lemma 4 (*Farkas' lemma*)**.** *Given a matrix $M \in \mathbb{R}^{k \times d}$ and a vector $c \in \mathbb{R}^d$, exactly one of the following is true:*

1. *There exists a vector $y \in \mathbb{R}^k$ satisfying $y^T M = c$ and $y \geq 0$.*
2. *There exists a vector $v \in \mathbb{R}^d$ satisfying $Mv \leq 0$ and $c^T v > 0$.*

Let $S = \{i \in [n] : A_i x^* = b_i\}$. To start, we prove that A_S and c cannot satisfy the second alternative (and hence must satisfy Farkas' first alternative).

If there existed a direction $v \in \mathbb{R}^d$ for which $A_S v \leq 0$ and $c^T v > 0$, then consider the point $x^* + \varepsilon v$ for ε to be determined. By assumption on v, we know that $A_S v \leq 0$ and hence $A_S(x^* + \varepsilon v) \leq b_S$. By construction of S we know that $\underset{S}{A_{[n]}} x^* < \underset{S}{b_{[n]}}$. Hence there exists some $\varepsilon > 0$ small enough for which $\underset{S}{A_{[n]}}(x^* + \varepsilon v) < \underset{S}{b_{[n]}}$. The point $x^* + \varepsilon v$ is thus a feasible solution to (1), but it has an objective value of $c^T(x^* + \varepsilon v) > c^T x^*$. This violates the assumption that x^* is optimal, and hence no such direction v can exist.

By Farkas' lemma, there must exist a vector $y_S \in \mathbb{R}^S$ with $y_S^T A_S = c$ and $y_S \geq 0$. We can amend this into a vector $y \in \mathbb{R}^n$ by setting $\underset{S}{y_{[n]}} = 0$. This vector satisfies $b^T y = b_S^T y_S = (A_S x^*)^T y_S = y_S^T A_S x^* = c^T x^*$. Observe that the vector y is feasible for (7). This proves that the optimal value of (7) is at least $b^T y^* \geq b^T y$. Weak duality tells us that $c^T x^* \geq b^T y^*$, and so we can go around and find $c^T x^* = b^T y \leq b^T y^* \leq c^T x^*$. This proves the theorem. □

Although we will not prove Farkas' lemma, please observe that it does exactly what Newton told us. The apple had no further direction v in which it could travel downwards ($c^T v > 0$) that wouldn't have it intersect the walls it was currently touching ($A_S(x^* + v) \leq b_S$, where S is the set of $i \in [n]$ with $A_i x^* = b_i$). In the physics story this meant the apple is at rest, hence the forces balance to zero. In Farkas' lemma, when we have an optimal point then the set of tight constraints there $A_S x \leq b_S$ allows for no further downward movement, hence the second alternative of Farkas is eliminated and the first alternative must be true. The equations $y^T M = c$ give us the same force balance as in the physics story.

Theorem 5 (*Complementary slackness*)**.** *Let $x^* \in \mathbb{R}^d$ be an optimal feasible solution for (1) and let $y^* \in \mathbb{R}^n$ be an optimal feasible solution for (7). Then for any $i \in [n]$, at least one of the following must hold: $y_i^* = 0$ or $A_i x = b_i$.*

Proof. Consider once again the equality $c^T x^* = \sum_{i \in [n]} y_i^* A_i x^*$. By strong duality, we know that $c^T x^* = b^T y^*$, and hence $\sum_{i \in [n]} y_i^* A_i x^* = \sum_{i \in [n]} y_i^* b_i$. For every individual $i \in [n]$, we already know that $A_i x^* \leq b_i$.

We prove complementary slackness by contradiction. Assume there is some $j \in [n]$ for which both $y_j^* > 0$ and $A_j x < b_j$. In that case we would have $y_j^* A_j x^* < y_j^* b_j$, which would imply that $\sum_{i \in [n]} y_i^* A_i x^* < \sum_{i \in [n]} y_i^* b_i$. We have found a contradiction, and hence no such j can exist. Therefore, for any $i \in [n]$ we have at least one of $y_i^* = 0$ or $A_i x = b_i$. □

Note that complementary slackness is clearly present already in the story of the fruit bowl. Only those parts of the fruit bowl that the apple touches ($A_i x^* = b_i$) can impart a normal force ($y_i A_i \neq 0$). In other words, when the apple does not touch a given facet of the bowl ($A_i x^* < b_i$) then that wall does not contribute to the normal force ($y_i A_i = 0$, hence $y_i = 0$). In our proofs, complementary slackness was a consequence of strong duality. Intuitively, I personally think of complementary slackness as preceding strong duality.

The dual LP is important for computational purposes. Somehow, for most applications, the simplex method on the dual LP is faster than it is on the primal LP. Because dual LPs tend to be structurally different from primal LPs, every high-quality software package for LP includes two simplex method implementations: a *primal simplex method* and a *dual simplex method.*

Mathematically the dual LP is also a rich object. In Alantha's lectures you will see some of its uses in approximation algorithms.

3.1. Certifying the optimal solution of an LP

The simplex method from last chapter terminates when it finds a feasible basis $B \subset [n]$ for which $cA_B^{-1} \geq 0$. When we define a vector $y \in \mathbb{R}^n$ by taking $y_B^T = cA_B^{-1}$ and $y_{[n]\setminus B} = 0$ then we can observe that y is dual feasible for (1). It satisfies complementary slackness, and hence it is an optimal dual solution with value $b^T y = c^T x^*$.

In other words, the simplex method terminates when its primal feasible basis gives a matching dual optimal feasible solution. This certificate allows the simplex method to prove that its primal solution is optimal too.

4. The Simplex Method is Slow

Next week (September 16) we will prove that the simplex method runs in exponential time in the worst case. (Actually, we will start by proving that the simplex method terminates in finite time assuming non-degeneracy. We forgot to do that earlier) We will prove this for the steepest edge pivot rule, by way of the extended formulation for the regular 2^k-sided polygon of Ben-Tal and Nemirovski.

If we have time left over, we will additionally prove that Bland's least-index pivot rule and Dantzig's most-negative reduced cost rule require exponential time on a Klee-Minty cube.

5. The Simplex Method is Fast

My next lecture after that will be on September 29 or 30. Here, we will get started on modern theory and examine why the simplex method is fast. I have not yet decided for sure, but most likely we will start with the Dantzig / Ye / Kitahara and Mizuno upper bound on the running time for Dantzig's pivot rule.

Depending on the mood, afterwards we either talk about degeneracy and bound shifting, or we look at a probabilistic model.

6. Subsequent lectures

The remaining lectures have not been planned yet. My hope is that we can reach close to the state-of-the-art of current research: the by-the-book analysis framework of Bach, Black, Huiberts and Kafer. That is the current best explanation available for the running time of the simplex method. If not that, we can at least do the BBHK proof under Dantzig / Ye / Kitahara and Mizuno assumptions.