# Project 6 — Linear Programming and the Simplex Method
# Due date: Wednesday, 21 December 2022, 11:59 PM

The purpose of this mini-project is to implement the Simplex Method to find the solution of linear programs, involving both the minimisation and the maximisation of the objective function.

## Linear Programming

Linear programming is an optimisation technique which aims to maximise or minimise a linear objective function subject to linear equality and inequality constraints. A typical linear program[1] is:

$$\begin{aligned} \max \ & f(x) = \sum_{i=1}^{n} c_i x_i, \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{1,j} x_j \leq h_1 \\ & \vdots \\ & \sum_{j=1}^{n} a_{m,j} x_j \leq h_m \\ & x_i \geq 0, \ \forall i \in \{1, \ldots, n\}. \end{aligned}$$

As we can observe above, we have a linear function $f(x)$, consisting of a sum of variables $x_i, \ i \in \{1, \ldots, n\}$, each one associated with a coefficient $c_i$. While finding the optimal value of the function without constraints would be a trivial task and would employ techniques you have already seen in previous courses, in this scenario the situation is complicated by the presence of linear constraints. In other words, the values of the vector $x_i$ cannot assume any arbitrary value - thus leading to an unbounded optimisation problem - but they need to satisfy a set of relationships, as well as the non-negativity condition (i.e., only positive values can be accepted and belong to the feasible region). To ease the notation, we can start by rewriting the above problem using matrix notation, as follows:

$$\begin{aligned} \max \quad & z = c^T x \\ \text{s.t.} \quad & Ax \leq h \\ & x \geq 0, \end{aligned} \tag{1}$$

where $z = f(x)$ represents the value of objective function - consisting in an inner product between the vector of coefficients (or weights) $c \in \mathbb{R}^n$ and the vector of unknowns $x \in \mathbb{R}^n$ - while $A \in \mathbb{R}^{m \times n}$ and $h \in \mathbb{R}^m$ are, respectively, the coefficients matrix and the right hand side of the constraints. The non-negativity condition, i.e., $x \geq 0$, stems from the fact that, since we are usually dealing with real quantities and practical problems, a solution with negative values would be meaningless. The formulation presented in Equation 1 is usually called *standard form* of a linear program, and it can be easily reformulated as follows in case we are dealing with a minimisation:

$$\begin{aligned} \min \quad & z = c^T x \\ \text{s.t.} \quad & Ax \geq h \\ & x \geq 0. \end{aligned} \tag{2}$$

[1] In this introductory chapter we choose to arbitrarily refer to a maximisation problem, but the same conclusions and observations can be equally extended to a minimisation problem.

Please notice that, in the case of minimisation, the inequalities signs are flipped. Passing from a generic formulation of the linear program to the standard from requires employing a set of rules, which help us dealing with the different scenarios we could face. For example, let us suppose that the $i$-th constraint is bounded between two values:

$$-h_{i_1} \leq \sum_{j=1}^{n} a_{i,j} x_j \leq h_{i_2}.$$

Since the standard form implies only the presence of inequalities, we can reformulate it as follows:

$$\sum_{j=1}^{n} a_{i,j} x_j \leq h_{i_2},$$
$$\sum_{j=1}^{n} a_{i,j} x_j \geq -h_{i_1},$$

i.e., we can consider the two inequalities separately. While the first of the two newly-written constraints can be immediately accepted in case of a maximization because it is already in standard form, the second one should undergo an additional transformation to be included in our problem:

$$\sum_{j=1}^{n} -a_{i,j} x_j \leq h_{i_1},$$

i.e., multiplying by $-1$ on both sides allows us to flip the inequality and to write the constraint in standard form. There are many different techniques which can be employed to reduce the constraints into standard form and a complete description of all of them goes beyond the scope of this assignment[2]. The methods relevant to the solution of this assignment will be discussed in class.
In the case $x \in \mathbb{R}^2$ we can give a graphical representation of our problem and of the constraints and, eventually, achieve a graphical solution, as the one proposed in the following example. Suppose we have to solve the following linear programming problem[3]:

$$\begin{aligned} \max \quad & z = 3x + 2y \\ \text{s.t.} \quad & x + 2y \leq 4 \\ & x - y \leq 1 \\ & x, y \geq 0 \end{aligned} \tag{3}$$

In Figure 1 we can find a graphical representation of our optimisation problem, where the feasible region defined by the constraints given has been highlighted. By construction, every point lying inside the feasible region satisfies all the constraints and could be a potential maximiser of the objective function $z$. How can we then proceed to find the optimal value? Thankfully, we can make use of the so-called **Fundamental Theorem of Linear Programming**, which can be formulated as follows:

**Theorem 1.** *If a linear program admits a solution (i.e., if it is neither unfeasible nor unbounded), this solution will lie on a vertex of the polytope defined by the feasible region. In the case in which two vertices are both maximisers (or minimisers) of the objective function, then also all the points lying on the line segment between them will represent optimal solutions of the linear programming problem.*

By bearing in mind this result, we can easily compute the value of $z$ at the vertices of the polytope, and find that the optimal value is $z = 8$ at the point $P = (2, 1)$. However, in general, computing the value of $z$ at all the vertices of the

[2] Those of you interested can check, as a reference, e.g., [1] and [2].
[3] The following example and figures are taken from [3].

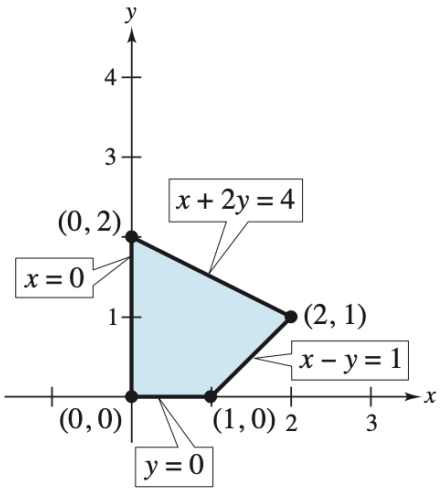


Figure 1: Representation of the feasible set for the example in equation 3.

feasible region can be a really expensive task, especially if we have several constraints and our optimisation problem is not confined in a two-dimensional space as it is in the simple example outlined in Equation 3. For this reason, we have to come up with a better solution strategy, capable of dealing also with higher dimensional problems. In this assignment we choose the simplex method, which will be explained in the next section.

## The Simplex Method

The simplex method is an algorithm for solving linear programming problems introduced by Dantzig in 1947. Even if other algorithms are available for the same purpose, it is still widely used in a lot of applications (e.g. in economics). Unfortunately this method has an important downside: in the worst case scenario its complexity is exponential and the number of candidate solutions that has to be examined to find the optimal value is too big to make its utilisation feasible.
In order to solve a linear program with the simplex method, we need to write our problem in the standard form presented in Equations 1 and 2, referring, respectively, to a maximisation and minimisation. We can then proceed by transforming all the inequalities into equality constraints, by adding *slack* variables (in case of a maximisation problem) or *surplus* variables (in case of a minimisation problem). These additional variables are also introduced in the objective function, with a coefficient equal to 0, and they have a specific meaning: they represent the difference between the value found in the optimal solution and the quantity on the right hand side (with the name slack or surplus referring only to the different sign). If we consider the example in equation 3, we have:

$$\begin{aligned} \max \quad & z = 3x + 2y + 0s_1 + 0s_2 \\ \text{s.t.} \quad & x + 2y + s_1 = 4 \\ & x - y + s_2 = 1 \\ & x, y \geq 0,\ s_1, s_2 \geq 0 \end{aligned} \tag{4}$$

where $s_1$ and $s_2$, along with their newly added non-negativity constraint, represent our slack variables. To further clarify the situation, it is helpful to consider the dimensionality of the quantities we are dealing with. Our coefficient vector will now be $c \in \mathbb{R}^{n+m}$, with $m$ representing the number of constraints, since - for every one of them - we need to add a slack variable to our original problem formulation. Of course, the unknowns vector will also change to $x \in \mathbb{R}^{n+m}$ and will have, as components, the $n$ unknowns of the original problem $(x_1, \ldots, x_n)$ and the $m$ slack variables $(s_1, \ldots, s_m)$. After these modifications, the extended coefficients matrix will be $A \in \mathbb{R}^{m \times (n+m)}$.

After redefining our problem with the introduction of the new variables, we can move to a general overview of the algorithm used to solve it. The simplex method starts from the identification of a basis of the extended coefficients matrix $A$, i.e., a square matrix with full rank containing a number of variables (both belonging to the original problem or to the newly added slack/surplus variables) equal to the number $m$ of constraints. The variables which will not be included in the basis are named *nonbasic variables* and, by setting their values all equal to 0, we can obtain what is called a *basic solution*. It is important to notice that not all basic solutions are feasible, since some of them could violate the non-negativity condition and, thus, cannot be accepted: it is always important to check that all constraints are satisfied, otherwise the algorithm would not give the expected results.
As a consequence of Theorem 1, it can be shown (see, e.g., [1]) that the existence of a feasible solution implies also the existence of a feasible basic solution; moreover the existence of an optimal solution implies the existence of an optimal basic solution, which will be the one searched for by the algorithm. The simplex algorithm seeks the solution of the linear programming problem by starting from a feasible basic solution and iterating through all the other basic solutions, whose number is:

$$N = \frac{(m+n)!}{m!n!}. \tag{5}$$

In the worst case scenario, the algorithm would achieve the optimal solution in the maximum possible number of iterations, which is exactly equal to the number of basic solutions. So, on one hand, we are sure that - if all the conditions are satisfied - the algorithm will reach the optimum in a finite number of iterations, but, on the other hand, this number grows exponentially with the number of variables and constraints (and, thus, the simplex method becomes unpractical for very large problems).
Let us now consider the simplex algorithm, which can be summarised as follows:

**Algorithm 1** Simplex Method

1: Write the problem in standard form and add slack/surplus variables
2: Consider a feasible starting basic solution (through solution of an *auxiliary problem*)
3: **while** the *optimality criterion* is not satisfied **do**
4: Apply the *iterative rule* by exchanging one basic and nonbasic variable.
5: **end while**
6: **return** the basic solution satisfying the optimality criterion

While the first point has been already tackled in the previous section, some additional clarifications about the other steps are needed to guide you through the implementation. You can find all the necessary details in the next paragraphs.

**The auxiliary problem.** In order to find a feasible starting solution, we decided to adopt the two-phase simplex method approach, which starts by solving an auxiliary minimisation problem, which – by introducing the artificial variables $u_1, \dots, u_m$ – can be defined as follows:

$$\begin{aligned}
\min\ & z = \sum_{i=1}^{n} u_i, \\
\text{s.t.}\ & \sum_{j=1}^{n} a_{1,j}x_j + s_1 + u_1 = h_1 \\
& \vdots \\
& \sum_{j=1}^{n} a_{m,j}x_j + s_m + u_m = h_m \\
& x_i, \dots, x_n \geq 0,\ s_1, \dots, s_m \geq 0,\ u_1, \dots, u_m \geq 0.
\end{aligned}$$

As we can notice, the auxiliary problem aims always at minimising the sum of the artificial variables, which are subject - as all the others - to the non-negativity constraint. The optimal solution of the auxiliary problem would then be achieved when all the artificial variables $u_1, \ldots, u_m$ have a value of 0. In case we cannot achieve a value of $z$ equal to 0, we say that the initial problem does not admit a feasible starting solution. A question that remains open is what would be a starting basic solution for the auxiliary problem. It can easily be shown that the latter is obtained by setting to 0 the original variables and the slack variables, and by setting equal to the right hand side $(h_1, \ldots, h_m)$ the artificial variables $u_1, \ldots, u_m$ just introduced. The optimal solution found for the auxiliary problem will then represent the starting feasible basic solution of the original problem and will then be plugged directly into it before applying the simplex method.

**The optimality criterion and the iterative rule.** At every iteration, we need to check whether or not the solution found satisfies the stopping criterion, defined - in our case - on the basis of the *reduced cost coefficients*:

$$r_D = c_D - c_B B^{-1} D. \tag{6}$$

Here $c_D$ represents the nonbasic coefficients of the objective function, $c_B$ the basic coefficients of the objective function, $B^{-1}$ the inverse of the basic matrix and $D$ the nonbasic matrix. The optimality condition for a maximisation problem is given by $r_D \leq 0$, while for a minimisation is $r_D \geq 0$. Please notice that the latter condition is satisfied when all components of the reduced cost coefficients vector are, respectively, less or bigger than 0.
If the optimality condition is not met, we have to decide which variable to take into the basis, by selecting the one with the highest reduced cost coefficient in case of a maximisation (or the one with the lowest for a minimisation). In case multiple variables have the same value, we could end up swapping the same two variables over and over again: this problem is known as *cycling* and it could potentially hinder the capacity of the simplex method to achieve convergence to the optimal value.
The **iterative rule** of the simplex algorithm consists instead in identifying the variable that has to be taken out of the basis, by computing the following ratio:

$$\frac{B^{-1}h}{B^{-1}D} \tag{7}$$

Among all the ratios obtained we then need to select the one with the smallest *positive* value. When the optimality condition is met, the algorithm stops at the optimal solution of the linear program.

## The assignment

Start by reading carefully the introduction to this mini-project and the additional material on linear programming. Then, solve the following exercises.

## 1. Graphical Solution of Linear Programming Problems [20 points]

Please consider the following two problems:

(1)

$$\begin{aligned} \min \quad & z = 4x + y \\ \text{s.t.} \quad & x + 2y \leq 40 \\ & x + y \geq 30 \\ & 2x + 3y \geq 72 \\ & x,\, y \geq 0 \end{aligned}$$

(2) A tailor plans to sell two types of trousers, with production costs of 25 CHF and 40 CHF, respectively. The former type can be sold for 85 CHF, while the latter for 110 CHF. The tailor estimates a total monthly demand of 265 trousers. Find the number of units of each type of trousers that should be produced in order to maximise the net profit of the tailor, if we assume that the he cannot spend more than 7000 CHF in raw materials.

Start by writing problem (2) as a linear programming problem. Then complete the following tasks:

- Solve the system of inequalities.
- Plot the feasible region identified by the constraints.
- Find the optimal solution and the value of the objective function in that point.

## 2. Implementation of the Simplex Method [30 points]

In this first part of the assignment, you are required to complete 2 functions which are part of a dummy implementation of the simplex method. Specifically you have to complete the TODOs in:

- *standardize.jl*, which writes a maximisation or minimisation input problem in standard form;
- *simplexSolve.jl*, which solves a maximisation or minimisation problem using the simplex method.

You are given also some already-implemented functions to help you in your task: *simplex.jl* is a wrapper which calls all the functions necessary to find a solution to the linear program; *auxiliary.jl* solves the auxiliary problem to find a feasible starting basic solution of the linear program; *printSol.jl* is a function which prints the optimal solution found by the simplex algorithm. Finally, *testSimplex.jl* presents a series of 6 problems to check if your implementation is correct, before moving to the next part of the assignment. Additional details to aid you in your implementation can be found in the comments inside the code.

## 3. Applications to Real-Life Example: Cargo Aircraft [25 points]

In this second part of the assignment, you are required to use the simplex method implementation to solve a real-life problem taken from economics (constrained profit maximisation).
A cargo aircraft has 4 compartments (indicated simply as $S_1, \ldots, S_4$) used to store the goods to be transported. Details about the weight capacity and storage capacity of the different compartments can be inferred from the data reported in the following table:

| Compartment | Weight Capacity ($t$) | Storage Capacity ($m^3$) |
|---|---|---|
| $S_1$ | 18 | 11930 |
| $S_2$ | 32 | 22552 |
| $S_3$ | 25 | 11209 |
| $S_4$ | 17 | 5870 |

The following four cargos are available for shipment during the next flight:

| Cargo | Weight ($t$) | Volume ($m^3/t$) | Profit (CHF/$t$) |
|---|---|---|---|
| $C_1$ | 16 | 320 | 135 |
| $C_2$ | 32 | 510 | 200 |
| $C_3$ | 40 | 630 | 410 |
| $C_4$ | 28 | 125 | 520 |

Any proportion of the four cargos can be accepted, and the profit obtained for each cargo is increased by $10\%$ if it is put in $S_2$, by $20\%$ if it is put in $S_3$ and by $30\%$ if it is put in $S_4$, due to the better storage conditions. The objective of this problem is to determine which amount of the different cargos will be transported and how to allocate it among the different compartments, while maximising the profit of the owner of the cargo plane. Specifically you have to:

1. Formulate the problem above as a linear program: what is the objective function? What are the constraints? Write down all equations, with comments explaining what you are doing.

2. Create a script *exercise2.jl* which uses the simplex method implemented in the previous exercise to solve the problem. What is the optimal solution? Visualise it graphically and briefly comment the results obtained (are you surprised of this outcome on the basis of your data?).

## 4. Cycling and Degeneracy [10 points]

Consider now the following simple problem:

$$\begin{aligned} \max \ & z = 3x_1 + 4x_2, \\ \text{s.t.} \ & 4x_1 + 3x_2 \leq 12 \\ & 4x_1 + x_2 \leq 8 \\ & 4x_1 + 2x_2 \leq 8 \\ & x_1, x_2 \geq 0. \end{aligned}$$

1. Create a script *exercise3.jl* which uses the simplex method implemented above to solve this problem. Do you achieve convergence within the maximum number of iterations (given by the maximum number of possible basic solutions)? Do you notice any strange behaviour? (*hint:* check, e.g., the indices of the entering and departing variables)

2. Look at the number of constraints and at the number of unknowns: what can you notice about the underlying system of equations? Represent them graphically and try to use this information to explain the behaviour of your solver in the previous point.

## Quality of the code and of the report [15 Points]

The highest possible score of each project is 85 points and up to 15 additional points can be awarded based on the quality of your report and code (maximum possible grade: 100 points). Your report should be a coherent document, structured according to the template provided on iCorsi. If there are theoretical questions, provide a complete and detailed answer. All figures must have a caption and must be of sufficient quality (include them either as .eps or .pdf). If you made a particular choice in your implementation that might be out of the ordinary, clearly explain it in the report. The code you submit must be self-contained and executable, and must include the set-up for all the results that you obtained and listed in your report. It has to be readable and, if particularly complicated, well-commented.

## Additional notes and submission details

Summarise your results and experiments for all exercises by writing an extended LaTeX report, by using the template provided on iCorsi (`https://www.icorsi.ch/course/view.php?id=14666`). Submit your gzipped archive file (tar, zip, etc.) – named `project_number_lastname_firstname.zip/tgz` – **on iCorsi** (see *[NC 2022] Project 6 - Submission Linear Programming and the Simplex Method*) **before the deadline**. Submission by email or through any other channel will not be considered. Late submissions will not be graded and will result in a

score of 0 points for that project. You are allowed to discuss all questions with anyone you like, but: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently. Please remember that plagiarism will result in a harsh penalisation for all involved parties.

## In-class assistance

If you experience difficulties in solving the problems above, please join us in class either on Tuesdays 16:30-18:15 or on Wednesdays 14:30-16:15 in room C1.03.

## References

[1] Dantzig, George. *Linear Programming and Extensions*. Princeton University Press, New Jersey, 1963.

[2] Maros, Istvaán. *Computational Techniques of the Simplex Method*. Springer Science & Business Media, New York, 2003.

[3] Larson, Ron. *Elementary linear algebra*. Nelson Education, 2016.