Optimisation Methods: Assignment 7

Deborah Sulem
deborah.sulem@usi.ch

Spring 2025

The purpose of this assignment is to familiarise yourself with concepts in linear programming and implement the Simplex algorithm.

It is due for **Wednesday 28th May at 10 am**. Only the second part (Python implementation) will be graded but you can also submit your answers for the first part and the latter will also be corrected (though not graded). For the second part, you need to submit a Python notebook on iCorsi and follow these instructions:

- Put the answers for each part of the question into separate cells.
- Before each cell, put a markdown header that says which part of the question comes in the following cell.
- Good coding style is part of the grade: add clear comments throughout the code when it is necessary.
- Before you submit your notebook, make sure it runs.

Part 0: reading

Read the Chapter 13 of the book "Numerical Optimisation" (Nocedal & Wright).

Part 1: exam-type exercises

Exercise 1
Consider the problem

$$\min_{x\in\mathbb{R}^2} x_1 + x_2 \quad \textit{subject to} \quad 0 \leq x_2 \leq 1$$

1. *Draw the feasible set and the cost vector.*
2. *Write this problem in standard form.*
3. *Identify all basic solutions and basic feasible solutions.*

Exercise 2
Consider the polyhedron

$$P = \{x \in \mathbb{R}^2 : x_1 + x_2 \leq 1, -x_1 + 2x_2 \leq 2, x_1 \geq 0, x_2 \geq 0\}$$

1. *Draw P.*

2. *Transform it into a polyhedron in standard form.*

3. *Identify all basic solutions and basic feasible solutions. For each of them, say if it is degenerate or not.*

Exercise 3
Consider the problem

$$\begin{aligned} \min_{x \in \mathbb{R}^4} x_1 \quad \textit{subject to} \quad & x_1 + x_2 + x_3 = 1 \\ & -x_1 + 2x_2 + x_4 = 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

Consider the basis with basic variables x_1 and x_2. Calculate the vector of reduced costs for that basis.

Exercise 4
Consider the problem:

$$\begin{aligned} \min -2x_1 + 2x_2 \qquad \textit{subject to} \ & x_1 + x_2 \leq 5 \\ & -x_1 + 2x_2 \leq 3 \\ & x_1 \geq 0 \\ & x_2 \geq 0. \end{aligned}$$

1. *Draw the feasible region and identify the minimiser graphically.*

2. *Write the above problem in standard form and solve it using the Simplex algorithm, starting from the point $(0, 0)$ in the original problem.*

Part 2: programming problems

Problem 1 (Basic solutions and reduced costs)
In this problem we consider the following polyhedron of $\mathbb{R}^2$:

$$P = \{x \in \mathbb{R}^2 : x_1 + x_2 \geq 1, x_1 + x_2 \leq 2, x_1 \geq 0, x_2 \geq 0\}$$

1. *Plot or draw P.*

2. *Transform P in standard form and define the corresponding constraint matrix A and vector b. We call m the number of rows of A.*

3. *Define a function `extract_basis_matrix` which constructs the basis matrix B from A and a list of basic indices. Test this function on the set of basic indices $[3, 4]$ and we call B_1 the result.*

4. *Define a function `basic_variables` which computes the basic variables x_B corresponding to a basis matrix B. If the input basis matrix B is singular this function should return None. Test this function on the previous basis matrix B_1 and we call x_{B1} the result.*

5. *Define two function* **`is_feasible`** *and* **`is_degenerate`** *which checks if the basic solution is respectively feasible and degenerate, given the basic variables x_B as input. Test these functions on the previous basis variables x_{B1}.*

6. *Define a function* **`full_basic_solution`** *which constructs the basic solution x given the number of variables n, the basic variables x_B and the list of basic indices. Test this function on the previous list of basic indices and basis variables x_{B1}.*

7. *Define a function* **`construct_basic_solution`** *which constructs the basic solution x given the constraint matrix A, the vector b, and the list of basic indices. This function should return the basic solution x (or None if it does not exist), and 2 booleans indicating if the basic solution is respectively feasible and degenerate. This function should call previously defined functions. Test this function on the previous list of basic indices and basis variables x_{B1}.*

8. *Construct the list of all possible sets of m indices in $\{1, \dots, n\}$ (without repetition) and apply the function* **`construct_basic_solution`** *to each set of m indices. For each of them print the result. How many basic feasible solutions does this polyhedron have and how many of them are degenerate?* Hint: you can use the function `combinations` from the library itertools to enumerate all sets of m indices.

9. *We now consider a LP problem over P:*

$$\min_{x \in P} c^T x$$

Define a function **`reduced_costs`** *which computes the (full) vector of reduced costs $\bar{c}$ with $\bar{c}_j = c_j - c_B^T B^{-1} A_j$, given the constraint matrix A, the cost vector c, and a set of basic indices. This function should return $\bar{c}$ as well as a boolean which is true if $\bar{c} \geq 0$. Test this function on $c = (1, 1, 0, 0)$ and the basic indices $[0, 3]$.*

10. *For each set of basic indices, compute the reduced costs and identify the optimal basic feasible solutions and the optimal cost. Observe that here we identify the minimiser by enumerating all possible basic feasible solutions and finding the one with the smallest cost.*

Problem 2 (Simplex algorithm)
In this problem we consider the following LP problem:

$$\min_{x \in \mathbb{R}^2} 3x_2 - 4x_1 \quad \textit{subject to} \quad \begin{aligned} 2x_1 + x_2 &\leq 6 \\ x_1 - x_2 &\leq 2 \\ x_1, x_2 &\geq 0. \end{aligned}$$

1. *Draw or plot the polyhedron corresponding to the feasible set in $\mathbb{R}^2$.*

2. *Write the above problem in standard form and define the corresponding constraint matrix A, cost vector c and vector b.*

3. *Initialisation: we want to initialise the Simplex algorithm starting from the point $x^0 = (0, 0)$ in the original problem. Compute the corresponding basic solution and check that it is a basic feasible solution. This will be our initial basic feasible solution* Hint: you may want to re-use functions of Problem 1...

4. *Define a function* **`index_entering_basis`** *which finds the non-basic index entering the basis using the smallest subscript rule, given as input the constraint matrix A, the cost vector c and a set of basic indices. This function should return None and print a message, if no entering index can be found, i.e., if the current basis is optimal. Test this function on the set of basic indices corresponding to the initial basic feasible solution and call j the output. Check that the corresponding reduced cost is negative.*

5. *Define a function* **`basic_direction`** *which computes the basic direction, given the constraint matrix A, a set of basic indices, and a non-basic index j leaving the basis. This function should return None if the basis matrix is singular. Test this function on the set of basic indices corresponding to the initial basic feasible solution and the entering basis index j, and call d the output.*

6. *Define a function* **`longest_step`** *which, given a current point x and a feasible direction d, computes the maximum θ such that $x + \theta d$ is feasible. Recall that θ is defined as*

$$\theta = \min_{i \in \{1,\dots,n : d_i < 0\}} -\frac{x_i}{d_i}.$$

This function should return both θ and ℓ, the smallest index for which $\theta = -\frac{x_\ell}{d_\ell}$, i.e., the index leaving the basis according to the smallest subscript rule. If $\theta = +\infty$, this function should return $(+\infty, None)$ and print a message like "this problem is unbounded". Test this function on the initial basic feasible solution and the basic direction d.

7. *Define a function* **`simplex_iteration`** *which performs one iteration of the Simplex method, given the constraint matrix A, the cost vector c, the vector b and a set of basic indices. This function should return the new set of basic indices, unless:*

    - *the current basis is optimal, in which case the function returns the current set of basis indices;*
    - *the problem is unbounded, in which case the function returns None.*

8. *Perform 3 iterations of Simplex method, starting from our initial basic feasible solution. Check that the last iteration returns an optimal basis and compute the corresponding minimiser and optimal cost.*