Chapter 1

Linear Programming

1.1. Objective of linear programming

The purpose of linear programming [MUR 83, MEL 04, VAN 08] is to optimize a linear function $J(\mathbf{x}) = \mathbf{f}^T\mathbf{x}$ of a set of variables grouped in vector $\mathbf{x} \in \mathbb{R}^n$ in the presence of linear constraints. This is one of the rare cases where an iterative algorithm converges into a finite number of iterations, by only using elementary manipulations.

1.2. Stating the problem

Consider a polyhedron in $\mathbb{R}^n$ (with $n \geq 2$), defined by a system of linear inequalities $\mathbf{Ax} \leq \mathbf{b}$. To each point of polyhedron, a value defined by linear function $J(\mathbf{x}) = \mathbf{f}^T$ is assigned. Here, $\mathbf{f} \in \mathbb{R}^n$ is a constant vector, initially known. By *linear programming* we understand a procedure, which enables us to solve the problem of finding a point $x \in \mathbb{R}^n$ of the polyhedron that minimizes or maximizes *J* function. Since the maximization problem is similar to the minimization one. This problem reads as follows:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^n} \mathbf{f}^T\mathbf{x} \\ \text{with} : \begin{cases} \mathbf{Ax} \leq \mathbf{b} \\ \mathbf{x} \geq \mathbf{0}, \end{cases} \end{cases} \qquad [1.1]$$

where "min" means *minimize* and: $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^{m}$, with $m < n$. Usually, J is referred to as *economic function* or *objective function* or simply *criterion* (of linear optimization). The inequalities $\mathbf{Ax} \leq \mathbf{b}$ define the constraints of the problem, while "s.t." stands for "subject to". Matrix $\mathbf{A}$ is by nature of maximum rank (i.e. *epic*), in order to make the constraints independent of each other.

To illustrate the corresponding geometric problem [1.1], consider the case of a polygon (in the Euclidean plane), as shown in Figure 1.1.

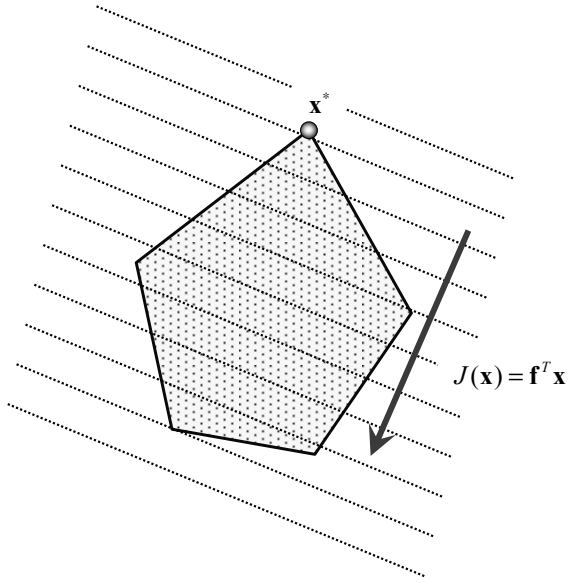


Figure 1.1. *Geometrical representation of the linear optimization problem*

The set of parallel lines is generated by considering $\mathbf{f}^T\mathbf{x}$ equal to various constants, hence the name *linear programming problem*. In this context, a result of mathematics states that the minimum can only be obtained at one of the polyhedron vertices (e.g. $\mathbf{x}^*$ in the figure). If the lines are also parallel to a side of the polyhedron, then all the points of this side correspond to an extreme of the objective function. More generally, a non-vertex polyhedron point can correspond to an optimal solution, only if there is an optimum side of the polyhedron that includes it.

Usually, the problem [1.1] is stated in *the canonical form* below:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^n} \mathbf{f}^T\mathbf{x} \\ \text{with: } \begin{cases} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0}, \end{cases} \end{cases} \quad [1.2]$$

where $\mathbf{f} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, and $\mathbf{b} \in \mathbb{R}^m$ (with $2 \leq m < n$) are the preset parameters.

Note, however, that problems [1.1] and [1.2] are not directly equivalent, but [1.1] can be expressed in the the canonical form, by introducing new variables that measure the difference between $\mathbf{b}$ and $\mathbf{A}\mathbf{x}$:

$$\Delta\mathbf{x} = \mathbf{b} - \mathbf{A}\mathbf{x}. \quad [1.3]$$

For this reason, $\Delta\mathbf{x} \in \mathbb{R}^m$ is referred to as the *offset vector*. With the notations:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{x} \\ \Delta\mathbf{x} \end{bmatrix} \in \mathbb{R}^{n+m}, \quad \mathbf{A}' = \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{m\times(n+m)}, \quad \mathbf{f}' = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n+m}, \quad [1.4]$$

where $\mathbf{I} \in \mathbb{R}^{m\times m}$ is the unit matrix, the problem [1.1] is expressed in canonical form [1.2]:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^n} \mathbf{f}'^T\mathbf{x}' \\ \text{with: } \begin{cases} \mathbf{A}'\mathbf{x}' = \mathbf{b} \\ \mathbf{x}' \geq 0. \end{cases} \end{cases} \quad [1.5]$$

When variables x_i are not subject to being positive, the canonical case is retrieved by introducing positive variables x_i^+ and x_i^-, so that:

$$x_i = x_i^+ - x_i^-, \quad \forall i \in \overline{1,n}. \quad [1.6]$$

Considering the canonical form of the problem, the objective is to find the point corresponding to the minimum of $\mathbf{f}^T\mathbf{x}$ in the polyhedron defined by the constraints. The minimization function being linear, this goal can easily be reached by testing the first-order optimality constraints, given that the solution necessarily corresponds to one of the polyhedron vertices (or to one of its edges). We can note that each vertex of the polyhedron has at least $n-m$ null coordinates.

1.3. Lagrange method

The function $\mathbf{f}^T\mathbf{x}$ being linear and the set of admissible solutions being convex, satisfying the first-order constraints is a necessary and sufficient condition of optimality.

To solve problem [1.2], it suffices to find the extrema of the *Lagrange function* below:

$$\mathcal{L}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda},\boldsymbol{\mu}) = \mathbf{f}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x}-\mathbf{b}) - \boldsymbol{\mu}^T\left(\mathbf{x}-\mathbf{y}^2\right), \qquad [1.7]$$

where $\boldsymbol{\lambda}\in\mathbb{R}^m$ and $\boldsymbol{\mu}\in\mathbb{R}^n$ are *Lagrange multipliers*, whereas, by definition, $\mathbf{y}^2 = \left[\, y_1^2 \;\; y_2^2 \;\cdots\; y_n^2 \,\right]^T$. The vector $\mathbf{y}^2$ has been introduced to replace inequality $\mathbf{x}\geq 0$ by equality $\mathbf{x}-\mathbf{y}^2 = 0$.

Cancelling overall gradient $\mathcal{L}$ means canceling partial gradients $\nabla_{\mathbf{x}}\mathcal{L}\equiv\mathcal{L}_{\mathbf{x}}$, $\nabla_{\mathbf{y}}\mathcal{L}\equiv\mathcal{L}_{\mathbf{y}}$, $\nabla_{\boldsymbol{\lambda}}\mathcal{L}\equiv\mathcal{L}_{\boldsymbol{\lambda}}$, and $\nabla_{\boldsymbol{\mu}}\mathcal{L}\equiv\mathcal{L}_{\boldsymbol{\mu}}$, These correspond in effect to the constraints of the first order. More precisely:

$$\begin{cases} \mathcal{L}_{\mathbf{x}} \equiv \mathbf{0} \\ \mathcal{L}_{\mathbf{y}} \equiv \mathbf{0} \\ \mathcal{L}_{\boldsymbol{\lambda}} \equiv \mathbf{0} \\ \mathcal{L}_{\boldsymbol{\mu}} \equiv \mathbf{0} \end{cases} \Leftrightarrow \begin{cases} \mathbf{f} + \mathbf{A}^T\boldsymbol{\lambda} - \boldsymbol{\mu} = \mathbf{0} \\ 2\mu_i y_i = 0, \;\; \forall i \in \overline{1,n} \\ \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \\ \mathbf{x} - \mathbf{y}^2 = \mathbf{0}. \end{cases} \qquad [1.8]$$

The second group of equations in the linear system [1.8] is particularly interesting. It implies the impossibility of having two non-zero elements μ_i and x_i at the same time. In fact, $\mu_i \neq 0$ and $x_i = 0$, $x_i \neq 0$ and $\mu_i = 0$, or $\mu_i = x_i = 0$. In short, $\boldsymbol{\mu}^T\mathbf{x} = 0$, with $\boldsymbol{\mu},\mathbf{x}\in\mathbb{R}_+^n$.

Therefore, from [1.8], the following equations are derived:

$$\begin{cases} \boldsymbol{\mu} = \mathbf{f} + \mathbf{A}^T\boldsymbol{\lambda} \geq \mathbf{0} \\ \boldsymbol{\mu}^T\mathbf{x} = \mathbf{x}^T\boldsymbol{\mu} = \mathbf{x}^T\left(\mathbf{f} + \mathbf{A}^T\boldsymbol{\lambda}\right) = \mathbf{0} \\ \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \\ \mathbf{x} - \mathbf{y}^2 = \mathbf{0}. \end{cases} \qquad [1.9]$$

The second equation of system [1.9] shows that the solution of the linear optimization problem is an orthogonal vector to $\mathbf{f}+\mathbf{A}^T\boldsymbol{\lambda}$, where $\boldsymbol{\lambda}$ is a vector varying in such a way that $\mathbf{f}+\mathbf{A}^T\boldsymbol{\lambda}\geq\mathbf{0}$. Of course, the last two equations identify themselves with the problem constraints.

1.4. Simplex algorithm

1.4.1. *Principle*

A well-known procedure to solve problem [1.2] by the system [1.9] comes from the *simplicial method*, referred to as the *simplex algorithm* [BLA 77]. This algorithm allows finding the minimum in a finite number of iterations and, moreover, by only using elementary computations. The approach is based on the idea search for the minimum among the vertices of the polyhedron defined by the constraints. Thus, starting from one of the vertices, the search is directed to the first vertex where the objective function decreases. If no such vertex exists, the minimum is given by the current vertex. Otherwise, the current vertex becomes the starting point for a new search.

The simplex algorithm is summarized in algorithm 1.1.

1. Determine a first vertex by stating the problem in simplicial form.
2. Check if the objective function has no solution.
3. If so, exit with the message: *no solution*.
4. Check if the selected vertex corresponds to the optimum.
5. If so, return the solution and exit with the message: *solution found*.
6. Otherwise, go to next vertex allowing the objective function to decrease.
7. Return to Step 4.

Algorithm 1.1. *Steps of the simplex algorithm principle*

1.4.2. *Simplicial form formulation*

If the problem can be stated that:

$$\mathbf{A}=\begin{bmatrix}\mathbf{I} & \mathbf{B}\end{bmatrix} \quad \text{and} \quad \mathbf{b}>\mathbf{0}, \qquad [1.10]$$

an evident vertex is:

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \qquad [1.11]$$

This formulation, known as the *simplicial form*, can be derived from the initial formulation in the following way.

The matrix $\mathbf{A}$ being epic, by permuting the components of vector $\mathbf{x}$, therefore of columns of $\mathbf{A}$, the constraints are expressed as follows:

$$\mathbf{A}'\mathbf{x}' = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix} \mathbf{x}' = \mathbf{b}, \qquad [1.12]$$

where $\mathbf{x}'$ represents the vector derived from $\mathbf{x}$ after performing the permutation that isolates the non-singular square matrix $\mathbf{A}_1$.

The pre-multiplication of $\mathbf{A}'$ and $\mathbf{b}$ by $\mathbf{A}_1^{-1}$ then leads to:

$$\mathbf{B} = \mathbf{A}_1^{-1}\mathbf{A}_2, \quad \mathbf{b}' = \mathbf{A}_1^{-1}\mathbf{b}. \qquad [1.13]$$

The simplicial form being defined, it has to be tested whether the vertex $\mathbf{x}'$ corresponds to the optimum or not, by checking if the first-order constraints are verified.

By segmenting $\mathbf{f}$ as:

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}, \quad \mathbf{f}_1 \in \mathbb{R}^m, \quad \mathbf{f}_2 \in \mathbb{R}^{n-m}, \qquad [1.14]$$

the first-order constraints are written as:

$$\begin{cases} \mathbf{f}_1 + \boldsymbol{\lambda} = \mathbf{0} \\ \mathbf{f}_2 + \mathbf{B}^T\boldsymbol{\lambda} \geq \mathbf{0}, \end{cases} \qquad [1.15]$$

given that $\boldsymbol{\mu}$ has to cancel itself where $\mathbf{x}$ is non-null (see equation [1.11]). After eliminating $\boldsymbol{\lambda}$ in [1.15], we obtain:

$$\mathbf{r} = \mathbf{f}_2 - \mathbf{B}^T\mathbf{f}_1 \geq \mathbf{0}. \qquad [1.16]$$

If the constraints [1.15] are not verified, a nearby vertex should be tested.

1.4.3. *Transition from one simplicial form to another*

Consider that the simplicial form [1.10] is not verifying the inequality [1.16]. Then all points $\mathbf{x}^T = \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T \end{bmatrix}$ for which:

$$\mathbf{x}_1 = \mathbf{b} - \mathbf{B}\mathbf{x}_2 \geq \mathbf{0}, \quad \mathbf{x}_2 \geq \mathbf{0}, \tag{1.17}$$

verify the constraints.

Denote $e \in \overline{1, n-m}$ the index of the smallest negative component of **r**, referred to as *input index*. Then, $\mathbf{b}_e$ is the corresponding column of **B** and $f_{2,e}$ is the *e*-th coordinate of $\mathbf{f}_2$, that is the (*m* + *e*)-th coordinate of **f**. Obviously:

$$f_{2,e} - \mathbf{b}_e^T \mathbf{f}_1 < 0. \tag{1.18}$$

Choose $\mathbf{x}_2$ null excepting for its *e*-th coordinate, equal to $\alpha \geq 0$. The corresponding vector $\mathbf{x}$ is admissible if:

$$\mathbf{x}_1 = \mathbf{b} - \mathbf{B}\mathbf{x}_2 = \mathbf{b} - \alpha \mathbf{b}_e \geq \mathbf{0} \quad \Leftrightarrow \quad b_j \geq \alpha b_{e,j}, \ \ \forall j \in \overline{1, m}. \tag{1.19}$$

The cost function to minimize is therefore written as:

$$\mathbf{f}^T \mathbf{x} = \mathbf{f}_1^T \mathbf{x}_1 + \mathbf{f}_2^T \mathbf{x}_2 = \mathbf{f}_1^T \left(\mathbf{b} - \alpha \mathbf{b}_e \right) + \alpha f_{2,e} = \mathbf{f}_1^T \mathbf{b} - \alpha \left(\mathbf{f}_1^T \mathbf{b}_e - f_{2,e} \right). \tag{1.20}$$

Due to property [1.18], the minimum of $\mathbf{f}^T \mathbf{x}$ corresponds to the greatest value of α verifying the inequality [1.19]. Two cases are to be considered:

1) If $\mathbf{b}_e \leq \mathbf{0}$, the problem has no finite solution.

2) If at least one element of $\mathbf{b}_e$ is non negative, since $\alpha \geq 0$, the inequality [1.19] involves that $b_{e,j}$ cannot be positive when b_j is negative. Consequently, from the index set $\overline{1, m}$, two subsets can be extracted: $\mathcal{E}_e^+$, the subsets of indices j for which $b_{e,j} > 0$, and $\mathcal{E}_e^-$, the index subsets corresponding to $b_{e,j} < 0$. Therefore, the inequality [1.19] implies:

$$\max_{j \in \mathcal{E}_e^-} \frac{b_j}{b_{e,j}} \leq \min_{j \in \mathcal{E}_e^+} \frac{b_j}{b_{e,j}}. \tag{1.21}$$

Since the inequality [1.19] is automatically verified for $b_{e,j} = 0$, the other inequality, [1.21], leads to the natural choice of $\alpha \geq 0$:

$$\alpha^* = \min_{j \in \mathcal{B}_e^+} \frac{b_j}{b_{e,j}}. \qquad [1.22]$$

In the second case, let $s \in \mathcal{B}_e^+$ be the corresponding index to the minimum, referred to as the *output index*. The point corresponding to α^* defines a new vertex of the constraints polyhedron, as it has $n - m$ null coordinates (when replacing [1.22] in [1.19], the s-th component of $\mathbf{x}_1$ is cancelled).

Once a non-null component of $\mathbf{x}_2$ is found, it can be saved in $\mathbf{x}_1$, on the incoming position (where, now, $\mathbf{x}_1$ is zero). After having permuted the coordinates of indices $m+e$ (input) and s (output) from vector $\mathbf{x}$, the linear constraint can be written as below:

$$\left[\mathbf{A}_s \quad \mathbf{A}_e\right]\mathbf{x} = \mathbf{b}, \qquad [1.23]$$

with $\mathbf{A}_s$ the matrix derived from the unit matrix by replacing its s-th column by $\mathbf{b}_e$ and $\mathbf{A}_e$ the matrix derived from $\mathbf{B}$ by replacing its e-th column by the s-th column of the unit matrix. To reach again the form [1.10], equation [1.24] has to be pre-multiplied by the inverse of $\mathbf{A}_s$. Fortunately, the particular form of this matrix facilitates the inversion. According to the Gauss procedure, the inverse $\mathbf{A}_s^{-1}$ corresponds to the unit matrix in which the s-th column has been replaced by the vector:

$$\left[-\frac{b_{e,1}}{b_{e,s}} \quad \cdots \quad -\frac{b_{e,s-1}}{b_{e,s}} \quad \frac{1}{b_{e,s}} \quad -\frac{b_{e,s+1}}{b_{e,s}} \quad \cdots \quad -\frac{b_{e,m}}{b_{e,s}}\right]^T. \qquad [1.24]$$

After the pre-multiplication of [1.23] by $\mathbf{A}_s^{-1}$, equation [1.10] is obtained again. Nevertheless, by difference from the initial equation, now, the e-th component of $\mathbf{x}_2$ is null (being, in fact, the s-th component of the former $\mathbf{x}_1$). When the previous operations are repeated, we note that the smallest negative component of the new vector $\mathbf{r}$ cannot be located at the previous e-th position, but at another one, which therefore gives the new input index e. At the end of this stage, the new $\mathbf{x}_2$ will surely have two null components.

The procedure is resumed when the first-order constraint [1.16] is verified. Note that the vector $\mathbf{r}$ defined by [1.16] has to be reconstructed at the end of each iteration. The number of iterations is at most $n-m$ (i.e. the number of $\mathbf{x}$ components to be cancelled).

1.4.4. *Summary of the simplex algorithm*

The simplex procedure starts from the following data:

– the objective vector: $\mathbf{f} \in \mathbb{R}^n$;

– the epic matrix: $\mathbf{A} \in \mathbb{R}^{m \times n}$ (with $m < n$);

– the free vector: $\mathbf{b} \in \mathbb{R}^m$.

The algorithm 1.2. of the simplex, shown below, is designed by assuming that the optimization problem is already formulated in canonical form [1.2], in order to use algorithm 1.2, of the simplex, shown below.

1. Initialization.

a. Test if matrix $\mathbf{A}$ is epic. If the test fails, remove the linearly dependent rows, such that the matrix becomes epic. In this case, update m by the number of remaining linearly dependent rows. In subsidiary, remove the elements of free vector $\mathbf{b}$ that correspond to removed rows from matrix $\mathbf{A}$.

b. Construct the *simplex table* as follows:

index rows and their permutations	$\boxed{1}$	$\boxed{2}$	$\cdots$	$\boxed{n-1}$	$\boxed{n}$	
solution of the problem	x_1	x_2	$\cdots$	x_{n-1}	x_n	
objective vector	f_1	f_2	$\cdots$	f_{n-1}	f_n	
epic matrix	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,n-1}$	$a_{1,n}$	b_1
	$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,n-1}$	$a_{2,n}$	b_2
	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
	$a_{m,1}$	$a_{m,2}$	$\cdots$	$a_{m,n-1}$	$a_{m,n}$	b_m
						free vector

• Notes: – Each permutation operated between the columns of matrix $\mathbf{A}$ is also operated between the columns of the table. In the end, the solution must be able to return to the initial succession of the elements, indicated by the first row of the table.

– Each pre-multiplication applied to matrix **A** also applies to free vector **b** of the table. Nevertheless, the first 3 rows of the table are not affected by this operation.

c. Complete the second row of the table with zero values. Therefore, the procedure starts from an initial point located in the origin of $\mathbb{R}^n$.

2. Make the main block of the epic matrix invertible.

If the main block of **A** is invertible, go to the next step. Otherwise, keep the $(m-1)$ first columns of the table and successively replace index column m by one of the index columns $m+1$, $m+2$, ..., until the main block becomes invertible. Once an appropriate column is found, it must swap its place with column m.

3. Pre-multiplication of the epic matrix and the free vector by the inverse of the main block.

After this operation, the main block of the matrix becomes unitary, whereas the conjoined block defines matrix **B** in the [1.10] form.

4. Iterative construction of the solution.

4.1. Construct the vector **r** according to definition [1.16], where vector $\mathbf{f}_1$ includes the first m elements of the third row in the simplex table, whereas vector $\mathbf{f}_2$ includes the remaining elements.

4.2. If inequality [1.16] is verified, stop the computational process and go to step 5.

4.3. Otherwise, determine the input index e, that corresponds to the (negative) minimum element of vector **r**. This indicates the column $m+e$ of the simplex table that includes vector $\mathbf{b}_e$ (the e-th column of matrix **B**).

4.4. Construct sets $\mathcal{E}_e^-$ and $\mathcal{E}_e^+$ for vector $\mathbf{b}_e$.

4.5. Calculate $x_e^- = \max_{j \in \mathcal{E}_e^-} \frac{b_j}{b_{e,j}}$ and $x_e^+ = \min_{j \in \mathcal{E}_e^+} \frac{b_j}{b_{e,j}}$ (by using vectors $\mathbf{b}_e$ and **b** of the simplex table). Keep the output index s for which $x_e^+ = b_s / b_{e,s}$.

4.6. If $x_e^- > x_e^+$, stop the algorithm with the message: *solution does not exist (conflicting constraints)*.

4.7. Otherwise, put $x_{m+e} = x_e^+$ on the simplex table.

4.8. Swap the index columns s and $m+e$ of the simplex table.

4.9. Pre-multiply the epic matrix and the free vector of the table by the quasi-unitary matrix, where the s -th column is defined by vector [1.24].

4.10. Resume the iterative process at step 4.1.

5. Complete the first m elements of the second row of the table by the values of the current free vector.
Normally, only the null elements should be replaced. The non null elements have already taken corresponding values of the free vector.

6. Return the solution with the elements ordered according to the index values of the first row of the simplex table.

Algorithm 1.2. *Main stages of the simplex algorithm*

1.5. Implementation example

The following problem will be solved by the simplex algorithm:

$$\left[\begin{array}{l}\min\limits_{\mathbf{x}\in\mathbb{R}^n}\left(x_1-2x_2+x_3\right)\\ \text{with: }\begin{cases}3x_1-2x_2+2x_3\le 1\\ 4x_1+12x_2+5x_3\le 2\\ x_1,x_2,x_3\ge 0.\end{cases}\end{array}\right. \qquad [1.25]$$

Define the offset variables $x_4 \ge 0$ and $x_5 \ge 0,$ in order to formulate problem [1.25] in the canonical form:

$$\left[\begin{array}{l}\min\limits_{\mathbf{x}\in\mathbb{R}^n} x_1-2x_2+x_3\\ \text{with: }\begin{cases}3x_1-2x_2+2x_3+x_4=1\\ 4x_1+12x_2+5x_3+x_5=2\\ x_1,x_2,x_3,x_4,x_5\ge 0.\end{cases}\end{array}\right. \qquad [1.26]$$

Construct the initial simplex table and perform a permutation that brings the unitary matrix in the main block in the epic matrix:

[1]	[2]	[3]	[4]	[5]	
0	0	0	0	0	
1	–2	1	0	0	
3	–2	2	1	0	1
4	12	5	0	1	2

;

[4]	[5]	[3]	[1]	[2]	
0	0	0	0	0	
0	0	1	1	–2	
1	0	2	3	–2	1
0	1	5	4	12	2

. [1.27]

It can be easily noted that:

$$\mathbf{B}=\begin{bmatrix}2 & 3 & -2\\ 5 & 4 & 12\end{bmatrix},\quad \mathbf{f}_1=\begin{bmatrix}0\\ 0\end{bmatrix},\quad \mathbf{f}_2=\begin{bmatrix}1\\ 1\\ -2\end{bmatrix} \qquad [1.28]$$

and therefore: $\mathbf{r}=\mathbf{f}_2-\mathbf{B}^T\mathbf{f}_1=\mathbf{f}_2$. The third component $\mathbf{r}$ is negative, whereby: $e=3,\ \mathbf{b}_e=\mathbf{b}_3=\begin{bmatrix}-2 & 12\end{bmatrix}^T,\ x_e^-=x_3^-=-1/2,\ x_e^+=x_3^+=1/6=x_5,$ and $s=2.$

Update the simplex table:

[4]	[5]	[3]	[1]	[2]	
0	0	0	0	1/6	
0	0	1	1	–2	
1	0	2	3	–2	1
0	1	5	4	12	2

;

[4]	[2]	[3]	[1]	[5]	
0	1/6	0	0	0	
0	–2	1	1	0	
1	–2	2	3	0	1
0	12	5	4	1	2

. [1.29]

Invert the main block of the epic matrix:

$$\begin{bmatrix}1 & -2\\ 0 & 12\end{bmatrix}^{-1}=\begin{bmatrix}1 & 1/6\\ 0 & 1/12\end{bmatrix}. \qquad [1.30]$$

Pre-multiply table [1.29] by the inverse [1.30]:

[4]	[2]	[3]	[1]	[5]	
0	1/6	0	0	0	
0	–2	1	1	0	
1	0	17/6	11/3	1/6	4/3
0	1	5/12	1/3	1/12	1/6

. [1.31]

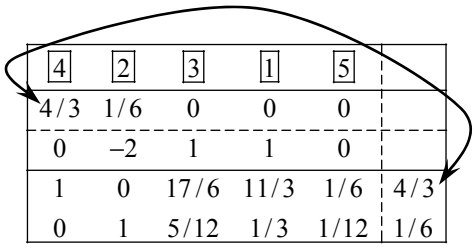
Now:

$$\mathbf{B}=\begin{bmatrix}17/6 & 11/3 & 1/6\\ 5/12 & 1/3 & 1/12\end{bmatrix},\quad \mathbf{f}_1=\begin{bmatrix}0\\ -2\end{bmatrix},\quad \mathbf{f}_2=\begin{bmatrix}1\\ 1\\ 0\end{bmatrix},\quad \mathbf{b}=\begin{bmatrix}4/3\\ 1/6\end{bmatrix}, \qquad [1.32]$$

which implies:

$$\mathbf{r} = \mathbf{f}_2 - \mathbf{B}^T\mathbf{f}_1 = \begin{bmatrix} 11/6 \\ 5/3 \\ 1/6 \end{bmatrix} \geq 0. \qquad [1.33]$$

The stop test being passed, the solution is directly given by the current free vector:



$$\Rightarrow \quad \mathbf{x} = \begin{bmatrix} 0 \\ 1/6 \\ 0 \\ 4/3 \\ 0 \end{bmatrix}. \qquad [1.34]$$

1.6. Linear programming applied to the optimization of resource allocation

1.6.1. *Areas of application*

Linear programming is particularly well adapted to optimizing the allocation of resources, in particular for achieving the objectives of a company subjected to management restrictions and environmental constraints. For this type of problem, the major difficulty is to reformulate it as a linear programming problem.

Since the resolution technique is completely defined by fully developed and easy to implement algorithms, it is on the formulation that the presentation of the various examples proposed in this chapter will focus. Regardless of the nature of the problem, the optimization is reduced to the minimization of an objective function.

1.6.2. *Resource allocation for advertising*

1.6.2.1. *Stating the problem*

In view of a major publicity campaign, a supermarket chain has to decide on the type of media to be used among radio, television and the press. The data of the problem are as follows:

– A flash radio advertisement can reach 20,000 potential buyers and costs 7,000 currency units (CU). The audience breakdown is shown in Table 1.1.

	Men	Women
Senior	2,500	3,500
Youth	6,000	8,000

Table 1.1. *Breakdown of potential buyers (radio listeners)*

– An advertising spot on television costs 4,000 CU and can reach 30,000 potential buyers, with the breakdown in Table 1.2.

	Men	Women
Senior	6,000	7,000
Youth	6,000	11,000

Table 1.2. *Breakdown of potential buyers (viewers)*

– An advertisement in the press costing 4,500 CU can reach 12,000 potential buyers, with the breakdown in Table 1.3.

	Men	Women
Senior	5,000	4,000
Youth	2,000	1,000

Table 1.3. *Breakdown of potential buyers (readers of the press)*

The proposed strategy is the following:

a) At least 220,000 potential buyers have to be advertised.

b) The number of young people must be at least twice the number of advertised seniors.

c) At least 40% of potential buyers have to be women.

d) The number of flash advertisements must be at least twice the number of advertisements in the press.

e) The number of advertisements in the press is limited to 7.

The problem is to find the number of flashes, the number of spots, and the number of advertisements, in order to have a minimum cost of the whole publicity campaign.

1.6.2.2. *Formulation as a linear programming problem*

Let:

– $x_1 \geq 0$ be number of flash radio advertisements;

– $x_2 \geq 0$ be number of commercials on television;

– $x_3 \geq 0$ the number of advertisements in the press.

The objective function to minimize is:

$$J(\mathbf{x}) = 7000x_1 + 4000x_2 + 4500x_3, \qquad [1.35]$$

where the constraints are given by the inequalities below:

$$\begin{cases} 20000x_1 + 30000x_2 + 12000x_3 \geq 220000 \\ 14000x_1 + 17000x_2 + 3000x_3 \geq 2\left(6000x_1 + 13000x_2 + 9000x_3\right) \\ 11500x_1 + 18000x_2 + 5000x_3 \geq 88000 \\ x_1 - 2x_3 \geq 0 \\ x_1 \leq 7, \end{cases} \qquad [1.36]$$

i.e.:

$$\begin{cases} -10x_1 - 15x_2 - 6x_3 \leq -110 \\ -2x_1 + 9x_2 + 15x_3 \leq 0 \\ -23x_1 - 36x_2 - 10x_3 \leq -176 \\ -x_1 + 2x_3 \leq 0 \\ x_1 \leq 7. \end{cases} \qquad [1.37]$$

It is important to note that the solution to problem [1.35] with constraints [1.37] must have integer values. This is an important constraint, which could not be taken into account in formulating the problem in terms of the linear programming. If the simplex algorithm does not lead to such solutions (which, in fact, is quite likely), we must round the non-integer values. Any number $x \in \mathbb{R}$ being framed by two consecutive integers:

$$\lfloor x \rfloor \leq x \leq \lceil x \rceil, \qquad [1.38]$$

the criterion [1.35] must now be minimized on a finite set of integer vectors (with $2^3 = 8$ maximum elements), from the non-integer solution given by the simplex algorithm.

1.6.3. *Optimization of a cut of paper rolls*

1.6.3.1. *Stating the problem*

A paper manufacturer receives the following orders:

– 120 rolls 60 cm wide;

– 200 rolls 75 cm wide;

– 190 rolls 90 cm wide;

– 180 rolls 110 cm wide.

Knowing that one can only have 50 rolls of 210 cm wide and that the number of rolls of 160 cm wide is limited, propose a cut satisfying the orders, while minimizing losses.

1.6.3.2. *Formulating the problem*

Let $x_i \geq 0$ be the number of 210 cm wide rolls with cut d_i, for $i \in \overline{1, N_{210}}$, where $N_{210} = 9$, as shown in Table 1.4. For example, if a roll of 90 cm is followed by a roll of 110 cm, the total cut is 200 cm, which would produce a waste roll of 10 cm wide.

No.	60 cm Roll	75 cm Roll	90 cm Roll	110 cm Roll	Cut d_i (cm)	Width of roll wasted (cm)
1	0	0	1	1	200	10
2	0	1	0	1	185	25
3	1	0	0	1	170	40
4	0	1	1	0	165	45
5	1	0	1	0	150	60
6	2	0	1	0	210	0
7	1	2	0	0	210	0
8	2	1	0	0	195	15
9	3	0	0	0	180	30

Table 1.4. *Possible cuts on 210 cm wide rolls*

Similarly, $x_{9+j} \geq 0$ is the number of 160 cm wide rolls with cut d_{9+j}, for $j \in \overline{1, N_{160}}$, where $N_{160} = 5$, as shown in Table 1.5.

No.	60 cm Roll	75 cm Roll	90 cm Roll	110 cm Roll	Cut d_{9+j} (cm)	Width of roll wasted (cm)
1	0	0	0	1	110	50
2	1	0	1	0	150	10
3	1	1	0	0	135	25
4	0	2	0	0	150	10
5	2	0	0	0	120	40

Table 1.5. *Possible cuts on 160 cm wide rolls*

The objective function to minimize is:

$$J(\mathbf{x}) = 10x_1 + 25x_2 + 40x_3 + 45x_4 + 60x_5 + 15x_8 + 30x_9 + 50x_{10} + 10x_{11} + 25x_{12} + 10x_{13} + 40x_{14}. \quad [1.39]$$

The constraints of the problem are expressed by the following system (according to Tables 1.4 and 1.5):

$$\begin{cases} x_3 + x_5 + 2x_6 + x_7 + 2x_8 + 3x_9 + x_{11} + x_{12} + 2x_{14} \geq 120 & \text{(rolls 60 cm)} \\ x_2 + x_4 + 2x_7 + x_7 + x_8 + x_{12} + 2x_{13} \geq 200 & \text{(rolls 75 cm)} \\ x_1 + x_4 + x_5 + x_6 + x_{11} \geq 190 & \text{(rolls 90 cm)} \\ x_1 + x_2 + x_3 + x_{10} \geq 180 & \text{(rolls 110 cm).} \end{cases} \quad [1.40]$$

The observation of the previous example (concerning integer values of the solution) is also valid in the case of problem [1.39] with constraints [1.40]. Nevertheless, the number of possibilities to be tested is much greater ($2^{14} = 16384$, at most).

1.6.4. *Structure of linear program of an optimal control problem*

1.6.4.1. *Stating the problem*

Given the linear process specification $\mathbf{x} \in \mathbb{R}^{nx}$, whose evolution is described by the discrete state equation:

$$\begin{cases} \mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n] \\ \mathbf{y}[n] = \mathbf{C}\mathbf{x}[n], \end{cases} \quad \forall n \in \mathbb{N}, \quad [1.41]$$

with $\mathbf{u} \in \mathbb{R}^{nu}$ control vector and $\mathbf{y} \in \mathbb{R}^{ny}$ output vector.

Let us note $\mathbf{y}_c$ the reference variable to be followed by the output system and $\boldsymbol{\varepsilon}$ the output error:

$$\boldsymbol{\varepsilon}[n] = \mathbf{y}_c[n] - \mathbf{y}[n], \quad \forall n \in \mathbb{N}. \tag{1.42}$$

State $\mathbf{x}$ is assumed to be known at each instant and the output at the final instant N is imposed:

$$\mathbf{y}[N] = \mathbf{y}_c[N]. \tag{1.43}$$

The purpose is to minimize criterion:

$$J(\mathbf{v}) = \sum_{n=1}^{N} \left[\sum_{i=1}^{nz} |z_i[n]| + k \sum_{j=1}^{ny} |\varepsilon_i[n]| \right], \tag{1.44}$$

where vector $\mathbf{v}$ will be defined later, whereas:

$$\mathbf{z}[n] = \mathbf{F}\mathbf{x}[n] + \mathbf{G}\mathbf{u}[n] \in \mathbb{R}^{nz}, \quad \forall n \in \overline{1, N}, \tag{1.45}$$

with matrices $\mathbf{F} \in \mathbb{R}^{nz \times nx}$ and $\mathbf{G} \in \mathbb{R}^{nz \times nu}$ initially known. Constant k is also known.

Constraints to be taken into account aim to limit the variation of the output and express themselves as follows:

$$|y_j[n]| \leq M_j, \quad \forall n \in \overline{1, N}, \ \forall j \in \overline{1, ny}, \tag{1.46}$$

i.e.

$$|y_j[n]| \leq M = \min_{j \in \overline{1, ny}} \{M_j\}, \quad \forall n \in \overline{1, N}, \ \forall j \in \overline{1, ny}. \tag{1.47}$$

1.6.4.2. *Structure of a linear program*

Firstly, the difference equation of system [1.41] has to be solved. Thus:

$$\mathbf{x}[n] = \mathbf{A}^n \mathbf{x}_0 + \sum_{k=0}^{n-1} \mathbf{A}^{n-1-k} \mathbf{B}\mathbf{u}[k], \quad \forall n \in \overline{1, N}. \tag{1.48}$$

To have positive variables, define:

$$u_i[n]=\begin{cases} u_i^+[n], & \text{if } u_i[n]\geq 0 \\ -u_i^-[n], & \text{if } u_i[n]<0 \end{cases}, \quad \forall n\in\overline{1,N}, \tag{1.49}$$

where unknown vectors $\mathbf{u}^+$ and $\mathbf{u}^-$ have non-negative elements.

Therefore:

$$\mathbf{u}[n]=\mathbf{u}^+[n]-\mathbf{u}^-[n], \quad \forall n\in\overline{1,N}, \tag{1.50}$$

Similarly, define:

$$\mathbf{z}[n]=\mathbf{z}^+[n]-\mathbf{z}^-[n] \quad \text{and} \quad \boldsymbol{\varepsilon}[n]=\boldsymbol{\varepsilon}^+[n]-\boldsymbol{\varepsilon}^-[n], \quad \forall n\in\overline{1,N}. \tag{1.51}$$

Constraint [1.47] can then be expressed in the form:

$$\begin{cases} y_j[n]\leq M \\ -y_j[n]\leq M, \end{cases} \quad \forall n\in\overline{1,N}, \ \forall j\in\overline{1,ny}. \tag{1.52}$$

That is, by introducing offset variables with non-negative components $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$, we can write:

$$\begin{cases} \boldsymbol{\alpha}^+[n]+\mathbf{y}[n]=M \\ \boldsymbol{\alpha}^-[n]-\mathbf{y}[n]=M, \end{cases} \quad \forall n\in\overline{1,N}. \tag{1.53}$$

Criterion [1.44] expresses as:

$$J(\mathbf{v})=\sum_{n=1}^{N}\left[\sum_{i=1}^{nz}\left(z_i^+[n]-z_i^-[n]\right)+k\sum_{j=1}^{ny}\left(\varepsilon_i^+[n]-\varepsilon_i^-[n]\right)\right]. \tag{1.54}$$

With previous notations, we obtain:

$$\begin{cases} \mathbf{Fx}[n]+\mathbf{Gu}[n]=\mathbf{z}^+[n]-\mathbf{z}^-[n] \\ \mathbf{y}_c[n]-\mathbf{y}[n]=\mathbf{y}_c[n]-\mathbf{Cx}[n]=\boldsymbol{\varepsilon}^+[n]-\boldsymbol{\varepsilon}^-[n], \end{cases} \quad \forall n\in\overline{1,N}. \tag{1.55}$$

Now, from [1.53], it results:

$$\begin{cases} \boldsymbol{\alpha}^+[n]+\mathbf{Cx}[n]=M \\ \boldsymbol{\alpha}^-[n]-\mathbf{Cx}[n]=M, \end{cases} \quad \forall n\in\overline{1,N}, \tag{1.56}$$

whereas [1.43] leads to:

$$\mathbf{C}\mathbf{x}[N] = \mathbf{y}_c[N]. \qquad [1.57]$$

When solution [1.48] and definition [1.50] are inserted into equations [1.55–1.57], the constraints of the problem can be expressed in the canonical form below:

$$\begin{cases} \mathbf{F}\mathbf{A}^n\mathbf{x}_0 + \mathbf{F}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right)+\mathbf{G}\mathbf{u}[n] = \mathbf{z}^+[n]-\mathbf{z}^-[n] \\ \mathbf{y}_c[n]-\mathbf{C}\mathbf{A}^n\mathbf{x}_0 - \mathbf{C}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right) = \boldsymbol{\varepsilon}^+[n]-\boldsymbol{\varepsilon}^-[n] \\ \boldsymbol{\alpha}^+[n]+\mathbf{C}\mathbf{A}^n\mathbf{x}_0 + \mathbf{C}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right) = M \qquad \forall n \in \overline{1,N}. \\ \boldsymbol{\alpha}^-[n]-\mathbf{C}\mathbf{A}^n\mathbf{x}_0 - \mathbf{C}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right) = M \\ \boldsymbol{\varepsilon}^+[N]-\boldsymbol{\varepsilon}^-[N] = 0, \end{cases} \qquad [1.58]$$

The problem defined by criterion [1.54] and constraints [1.58] is clearly structured in a linear program with non-negative unknowns: $\mathbf{u}^+[n]$, $\mathbf{u}^-[n]$, $\mathbf{z}^+[n]$, $\mathbf{z}^-[n]$, $\boldsymbol{\varepsilon}^+[n]$, $\boldsymbol{\varepsilon}^-[n]$, $\boldsymbol{\alpha}^+[n]$ and $\boldsymbol{\alpha}^-[n]$, at each instant $n \in \overline{1,N}$. In fact, unknown variable vector $\mathbf{v}$ comprises all vectors for the horizon control $\overline{1,N}$. The computational effort of the simplex algorithm may become high, because of the high number of unknown variables and, especially, the number of constraints, as these sizes depend on the length of the horizon control, N.

An alternative problem, with a significantly reduced computational effort can be formulated for each instant $n \in \overline{1,N}$ of the horizon control, from criterion:

$$J(\mathbf{v}_n) = \sum_{i=1}^{nz}\left(z_i^+[n]-z_i^-[n]\right)+k\sum_{j=1}^{ny}\left(\varepsilon_i^+[n]-\varepsilon_i^-[n]\right), \qquad [1.59]$$

where, this time, the unknown vector variable $\mathbf{v}_n$ includes only current values of the vectors mentioned above. The constraints are therefore:

$$\begin{cases} \mathbf{F}\mathbf{A}^n\mathbf{x}_0 + \mathbf{F}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right)+\mathbf{G}\mathbf{u}[n]=\mathbf{z}^+[n]-\mathbf{z}^-[n] \\ \mathbf{y}_c[n]-\mathbf{C}\mathbf{A}^n\mathbf{x}_0-\mathbf{C}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right)=\boldsymbol{\varepsilon}^+[n]-\boldsymbol{\varepsilon}^-[n] \\ \boldsymbol{\alpha}^+[n]+\mathbf{C}\mathbf{A}^n\mathbf{x}_0+\mathbf{C}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right)=M \\ \boldsymbol{\alpha}^-[n]-\mathbf{C}\mathbf{A}^n\mathbf{x}_0-\mathbf{C}\sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\left(\mathbf{u}^+[k]-\mathbf{u}^-[k]\right)=M, \end{cases} \quad [1.60]$$

the final condition [1.43] being necessarily disqualified.

Generally, the instantaneous problem is used for systems where parameters **A**, **B**, **C**, **F** and **G** vary in time. Weight k set in definition [1.59] may also vary from one instant to another. In this case, the simplex algorithm provides a solution that allows the output to track the variable reference and the input to adapt to parameter changes. In the previous case, the output must equal the variable reference in a prescribed number of instants, which could be unrealistic in practice, especially for reduced periods of horizon control.