Non linear optimization, gradient methods

Recherche opérationnelle
Dimitri Watel - ENSIIE

2024

In this chapter, we focus on two gradient descent methods; these methods extend linear programming techniques to the optimization of arbitrary (non-linear) functions. For simplicity, we will assume that the constraints are linear, but these techniques can also be adapted to more general cases.

Let $f, g_i, h_j : \mathbb{R}^n \to \mathbb{R}, \forall\, i \in [\![1; m]\!], \forall\, j \in [\![1; p]\!]$ of class C^1.

We want to solve

$$\begin{array}{lll} \min & f(x) & \\ s.c. & g_i(x) \leq 0 & \forall\, i \in [\![1; m]\!] \\ & h_j(x) = 0 & \forall\, j \in [\![1; p]\!] \end{array}$$

We write S the set of feasible solutions.

1 Qualification and optimality conditions

We will make some reminders about these two concepts that will be used in the course.

Definition 1 (Saturated constraints). Let $x \in S$. We write the saturated constraints

$$I(x) = \{i \in [\![1; m]\!] | g_i(x) = 0\}$$

Definition 2 (Karush-Kuhn-Tucker conditions). $x \in S$ satisfies the Karush-Kuhn-Tucker conditions (KKT) if:

$$\exists \lambda_i \geq 0, \mu_j \in \mathbb{R} \qquad \forall\, i \in [\![1; m]\!] \quad j \in [\![1; p]\!]$$

$$\nabla f(x) + \sum_{i \in I(x)} \lambda_i \nabla g_i(x) + \sum_{j=1}^{p} \mu_j \nabla h_j(x) = 0$$

Definition 3 (Qualification of linear independance). x satisfies the linear independance qualification if the vectors $(\nabla g_i(x), i \in I(x^*) \cup (\nabla h_j(x), j \in [\![1; p]\!])$ are linearly independant.

Remark: there exists a more general definition of qualification, but we will not use it.

Theorem 1.1 ((KKT) necessary conditions). *If $x^* \in S$ is a local minimum of f and if x^* is qualified then x^* satisfies (KKT).*

Theorem 1.2 ((KKT) sufficient conditions). *If x^* satisfies (KKT), if the functions f et g_i are convex and if the functions h_j are linear, then x^* is a global minimum.*

We propose two algorithms in the following sections that allow us to reach a point satisfying the Karush-Kuhn-Tucker conditions in order to use the sufficient optimality conditions.

2 Projected gradient algorithm

Hypothesis: we assume that all points in S are qualified under linear independence.

The projected gradient algorithm is a gradient descent that does not go out of S. To do so, when we touch the boundary of S, we do not follow the gradient but a projection of the gradient on the boundary.

It stops when the (KKT) conditions are satisfied.

We use the following notations:

$$g_i(x) = \sum_{k=1}^{n} (a_{ik} x_k) - b_i \leq 0 \ \forall i \in [\![1; m]\!]$$

$$h_j(x) = \sum_{k=1}^{n} (a'_{jk} x_k) - b'_j = 0 \ \forall j \in [\![1; p]\!]$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \\ a'_{11} & a'_{12} & \cdots & a'_{1n} \\ a'_{21} & a'_{22} & \cdots & a'_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a'_{p1} & a'_{p2} & \cdots & a'_{pn} \end{pmatrix} \qquad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \\ b'_1 \\ b'_2 \\ \vdots \\ b'_p \end{pmatrix}$$

Notice that

$$\begin{array}{ll} L_i = {}^t\nabla(g_i(x)) & \forall i \in [\![1; m]\!] \\ L_{j+m} = {}^t\nabla(h_j(x)) & \forall j \in [\![1; p]\!] \end{array}$$

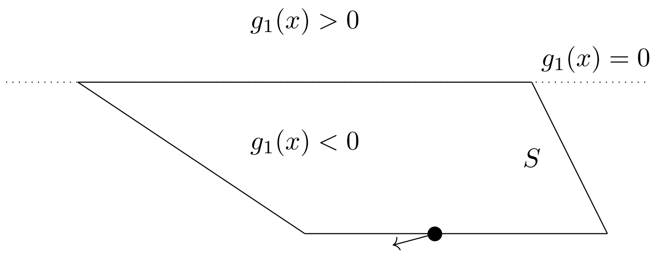
We start at a point $x \in S$ and move along the gradient. This point x is not necessarily easy to find. One possibility is to use the so-called two-phase method, but that is not the subject of this chapter, so we assume that x is given to us. Each iteration of the algorithm consists of 3 steps: 1. detect the edges touched by x; 2. project the negative gradient onto the edges and decide on a direction to follow; 3. follow this direction as long as it decreases the value of f.

2.1 Detect the boundaries

The set S is represented by a series of inequalities g_i and equalities h_j. Since these functions are linear, we have a polytope here (a generalization of a polygon in dimensioni n). This polytope has a boundary for each inequality $g_i(x) \leq 0$. We are on the boundary if $g_i(x) = 0$ and we are outside the boundary if $g_i(x) < 0$ (and we are on the *wrong side* of the boundary if $g_i(x) > 0$). We can also consider an equality as a somewhat special boundary (since it wraps around the entire space S). Thus, to summarize, the set of boundaries is defined with the following two sets:

$$I(x) = \{i | g_i(x) = 0\}$$
$$J = \{j | h_j(x) = 0\} = [\![1; p]\!]$$

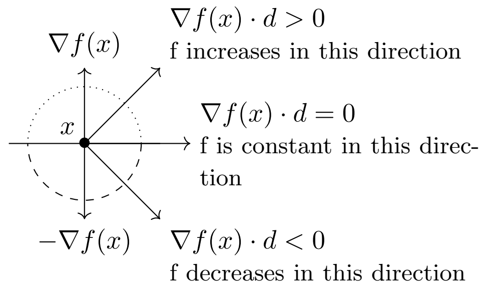
We recognize the set of saturated constraints $I(x)$.



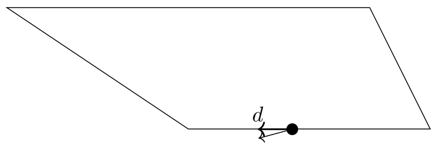
2.2 Project on the boundaries

We want to project the opposite of the gradient onto the boundary. First and foremost, we want to follow the opposite of the gradient because the gradient indicates the direction to take to increase the function the most. By following the opposite, we are moving in the direction that decreases the function the most. However, it is not always possible to follow this direction, especially if we are on a boundary. Therefore, we need to project onto the boundary to follow the closest valid direction to the opposite of the gradient.

To follow the boundary, one must follow a direction that keeps us on the boundary. To do this, we need to follow a direction such that $g_i(x)$ for $i \in I(x)$ and $h_j(x)$ for $j \in J$ remain zero. For any $\mathcal{C}^1$ function, a direction that does not change the value of a function is orthogonal to the gradient of that function.


$\nabla f(x) \cdot d > 0$
f increases in this direction
$\nabla f(x)$
x
$\nabla f(x) \cdot d = 0$
f is constant in this direction
$-\nabla f(x)$
$\nabla f(x) \cdot d < 0$
f decreases in this direction

In order to stay on the boundary, we will therefore look for a direction that is orthogonal to the gradient of all the constraints defining this boundary.



Lemma 2.1. *Let $A_s = \{L_i, i \in I(x) \cup J\}$. We assume that $I(x) \cup J \neq \emptyset$. Let $B = \{y | L_i \cdot y = 0 \ \forall i \in I(x) \cup J\}$ The projected gradient d is*

$$\begin{aligned} d &= \textit{Projection of } (-\nabla f(x)) \textit{ on } B \\ &= \left(I_n - {}^tA_s \cdot (A_s \cdot {}^tA_s)^{-1} \cdot A_s\right) \cdot (-\nabla f(x)) \end{aligned}$$

Proof. The space onto which we project, $B = \{y | L_i \cdot y = 0 \ \forall i \in I(x) \cup J\}$, is orthogonal to all linear combinations of the vectors L_i for $i \in I(x) \cup J$. Thus, it is the orthogonal space to all the rows of A_s. Let us suppose that we find the projection p of $-\nabla f(x)$ onto the space spanned by the rows of A_s; then, to find the orthogonal projection d of x onto B, it is sufficient to calculate $d = -\nabla f(x) - p$.

To find p, it suffices to realize that projecting onto the rows of A_s is equivalent to projecting onto the columns of ${}^tA_s$. We can then use the following known result:

Given a vector z and a matrix A such that ${}^tAA$ is invertible, the orthogonal projection of z onto the space spanned by the columns of A is $A({}^tAA)^{-1}\,{}^tAz$.

If we replace A with ${}^tA_s$, we need to show that $A_s \cdot {}^tA_s$ is invertible. However, every point in S satisfies the qualification of linear independence, so the columns of ${}^tA_s$ are linearly independent, making the rank of ${}^tA_s$ equal to the number of its columns: $|J| + |I(x)|$. In other words, its kernel is empty. Finally, if z is in the kernel of $A_s \cdot {}^tA_s$, we have

$$\begin{aligned} A_s \cdot {}^tA_s \cdot z &= 0 \\ ({}^tz \cdot A_s) \cdot {}^tA_s \cdot z &= 0 \\ \|{}^tA_s \cdot z\|^2 &= 0 \end{aligned}$$

This vector is in $\mathbb{R}^n$ thus

$${}^tA_s \cdot z = 0$$

Therefore, z is in the kernel of ${}^tA_s$. Since this is empty, we deduce that the kernel of $A_s \cdot {}^tA_s$ is also empty. Thus, the rank of $A_s \cdot {}^tA_s = |J|+|I(x)|$, so this matrix is indeed invertible.

Thus, the projection p of $-\nabla f(x)$ onto the space spanned by the rows of A_s is

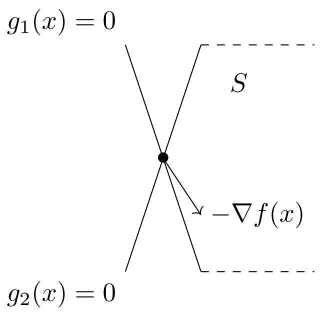
$$\begin{aligned} p &= {}^tA_s \cdot (A_s \cdot {}^tA_s)^{-1} \cdot A_s \cdot (-\nabla f(x) \\ d &= -\nabla f(x) - p \\ d &= \left(I_n - {}^tA_s \cdot (A_s \cdot {}^tA_s)^{-1} \cdot A_s\right) \cdot (-\nabla f(x)) \end{aligned} \qquad \square$$

Remark 1. If $I(x) = J = \emptyset$ then we simply set $d = -\nabla f(x)$ since we project onto the entire space.

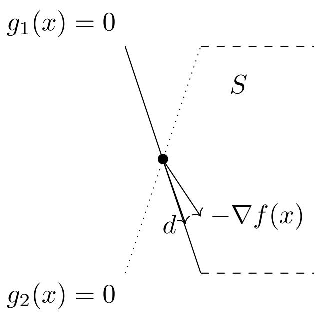
If $d \neq 0$, we can move on to the next step to follow this direction.

One may encounter a problem if $d = \vec{0}$. That is, if the space onto which we project is orthogonal to the gradient. This can occur in two scenarios: either we have reached a local minimum, or we have projected onto too many edges at once.

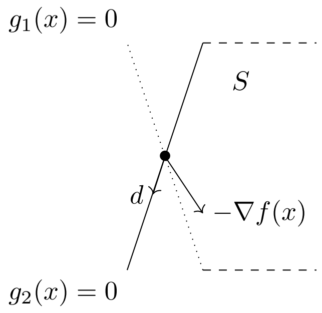
Before demonstrating what happens in this case, let's consider some examples in 2 dimensions. In the following figure, we have a projection of the gradient at a corner of the polygon; this corner is defined by two inequality constraints, g_1 and g_2, which define two edges of the polygon. Specifically, we are trying to project a vector onto two lines simultaneously. The only possible solution is the zero vector: the only vector orthogonal to the normals of the two lines.



The algorithm must follow one of the two edges. To choose which edge to follow, the projected gradient algorithm will attempt to project onto each of the two edges, as if, temporarily, the other edge had been removed.



The direction d is valid; if we follow this direction, we stay within S. Here is what happens if we now try to project onto the edge identified by the constraint g_1 without considering the constraint g_2.



We note this time that following the direction d is not valid, as it causes us to exit the feasible solution set S. Therefore, it is important to make the right choice. We have, in a way, identified that g_1 is a necessary constraint to move forward in this iteration, while the constraint g_2 is inactive. The goal of the algorithm is to identify an inactive constraint, temporarily remove it, project onto the other constraints, and continue as normal. Note that there may be several good choices (thus several inactive constraints), and there may also be no good choice (no inactive constraints). In the latter case, it can be shown that the (KKT) conditions are satisfied, which stops the algorithm. In the former case, it is sufficient to arbitrarily choose one of the possibilities.

To identify an inactive constraint, one could look to remove each constraint g_i one by one and test the direction d_i in each case. To determine if the direction d_i is valid, it is necessary to check that g_i remains negative if we move in this direction. Thus, since g_i was zero at x, it suffices to follow a direction close to the opposite of the gradient of g_i to decrease g_i. Therefore, this new direction d_i must satisfy $\nabla g_i(x) \cdot d_i < 0$. However, rather than testing each constraint, we will see that there is a simpler method that reliably provides the correct direction. Note that this method is only valid if the linear independence qualification assumption holds.

Lemma 2.2. *If $d = \vec{0}$, then, there exists λ_i, for $i \in I(x)$ and μ_j for $j \in J$ such that*

$$-\nabla f(x) = \sum_{i \in I(x)} \lambda_i \cdot L_i + \sum_{j \in J} \mu_j \cdot L_{m+j}$$

.

Proof. We projected $-\nabla f(x)$ onto $B = \{y | L_i \cdot y = 0 \; \forall i \in I(x) \cup J\}$. Thus, in the case where the projection is zero, this means that $-\nabla f(x)$ is orthogonal to B. Therefore, it belongs to the space generated by the lines of A_s, hence the result. $\square$

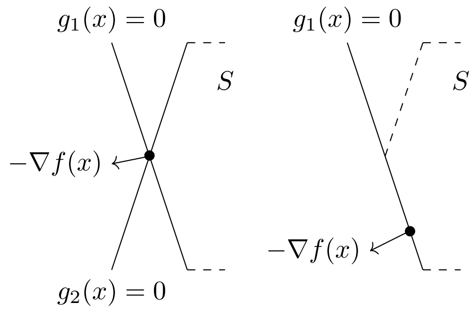
In this case, we can calculate λ and μ by noting that the formula from lemma 2.2 can be rewritten as follows:

$$-\nabla f(x) = {}^t A_s \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$$

$$(A_s \cdot {}^t A_s)^{-1} A_s \cdot (-\nabla f(x)) = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$$

It is the values of λ_i that will indicate the inactive or necessary constraints. The μ_j do not indicate anything. The equality constraints are **always** necessary. The rule is very simple. If $\lambda_i < 0$, then g_i is inactive.

It can be noted that the formula of lemma 2.2 is exactly the conditions (KKT) if $\lambda_i \geq 0$ for all $i \in I(x)$. In this case, we can stop the algorithm; we have achieved our objective. Here are two examples of 2D cases where d is zero and no boundary constraint is inactive. Given the direction of the gradient, we see that we indeed have a local minimum.



The following lemma demonstrates that any constraint such that $\lambda_i < 0$ is inactive.

Lemma 2.3. *Assuming there exists $k \in I(x)$ such that $\lambda_k < 0$ then let $B' = \{y | L_i \cdot y = 0 \ \forall i \in I(x) \backslash \{k\} \cup J\}$ and*

$$d' = \text{Projection of } (-\nabla f(x)) \text{ on } B'$$

Then $d' \neq \vec{0}$ and $\nabla g_k(x) \cdot d' < 0$.

Remark 2. This lemma is true only due to the assumption of the qualification of linear independence.

Proof. According to Lemma 2.2, we have

$$-\nabla f(x) = \sum_{i \in I(x)} \lambda_i \cdot L_i + \sum_{j \in J} \mu_j \cdot L_{m+j}$$

As $d' \in B'$, if $d' = \vec{0}$ then, there exists λ'_i, for $i \in I(x) \backslash \{k\}$ and μ'_j for $j \in J$ such that

$$-\nabla f(x) = \sum_{i \in I(x) \backslash \{k\}} \lambda'_i \cdot L_i + \sum_{j \in J} \mu'_j \cdot L_{m+j}$$

However as x is qualidifed with linear independence.

$$\lambda_i = \lambda'_i \text{ pour } i \in I(x) \backslash \{k\}$$
$$\lambda_k = 0$$
$$\mu_j = \mu'_j \text{ pour } j \in J$$

Or $\lambda_k < 0$ by assumption, so there is a contradiction and thus $d' \neq \vec{0}$. Now, let's show that $L_k \cdot d' < 0$. Since $d' \in B'$, it follows that d' is orthogonal to L_i for $i \in I(x) \backslash \{k\}$, and to L_{m+j} for $j \in J$. Therefore, picking up from the first equality:

$$-\nabla f(x) \cdot d' = \lambda_k \cdot L_k \cdot d'$$

Knowing a vector v and its non-zero projection p onto a subspace, we have $v \cdot p = \|p\|^2 > 0$. Thus

$$0 < \lambda_k \cdot L_k \cdot d'$$

As $\lambda_k < 0$

$$0 > L_k \cdot d'$$ □

One very important point is that the inactive constraint is not permanently removed from the set of constraints. It must be reconsidered in the following iterations if it becomes active again.

2.3 Follow the direction

The last step is a classic gradient descent step, which involves following the given direction until a global or local minimum of that direction is reached. We set the following values:

$$\alpha_1 = \max_{0 \leq \alpha} (x + \alpha \cdot d \in S)$$

$$\alpha_2 = \arg \min_{0 \leq \alpha \leq \alpha_1} (f(x + \alpha \cdot d))$$

The next point is $x + \alpha_2 \cdot d$. A new iteration is started from the beginning with this new point. Iterations are performed until a point is found where the conditions (KKT) are satisfied.

Lemma 2.4. *If $d \neq \vec{0}$ then $\alpha_1, \alpha_2 \neq 0$.*

Proof. □

The pseudo-code of the algorithm is the following:

```
1:  Find $x \in S$
2:  $I(x) \leftarrow \{i \in [\![1; m]\!] | g_i(x) = 0\}$
3:  loop
4:      $A_s \leftarrow \{L_i, i \in I(x) \cup J\}$
5:      $d \leftarrow (I_n - {}^t A_s \cdot (A_s \cdot {}^t A_s)^{-1} \cdot A_s) \cdot (-\nabla f(x))$
6:      if $d = 0$ then
7:          ${}^t(\lambda, \mu) \leftarrow (A_s \cdot {}^t A_s)^{-1} A_s \cdot (-\nabla f(x))$
8:          If $\exists i \in I(x), \lambda_i < 0$, Remove $i$ from $I(x)$
9:          Else return $x$
10:     else
11:         $\alpha_1 \leftarrow \max_{0 \leq \alpha} (x + \alpha \cdot d \in S)$
12:         $\alpha_2 \leftarrow \arg \min_{0 \leq \alpha \leq \alpha_1} (f(x + \alpha \cdot d))$
13:         $x \leftarrow x + \alpha_2 \cdot d$
14:         $I(x) \leftarrow \{i \in [\![1; m]\!] | g_i(x) = 0\}$
```

The projected gradient is relatively simple to implement in the case of linear constraints. In the nonlinear case, the projection is more complicated. It should also be noted that this algorithm behaves like a gradient descent when there are no constraints.

3 Reduced gradient algorithm

Hypothesis: we assume that there are no inequality constraints.

The reduced gradient algorithm is a gradient descent that does not go out of S. Because of the equalities, we can rewrite some of the variables x_B with the rest of the variables x_N. We then *reduce* $f(x)$ and $\nabla f(x)$ to $f(x_N)$ and $\nabla f(x_N)$

It stops when the (KKT) conditions are satisfied. This algorithm is similar to the simplex algorithm.

Definition 4 (Augmented form)**.** Let (P) be the following program

$$\begin{array}{lll} \min & f(x) & \\ s.c. & g_i(x) \leq 0 & \forall\ 1 \leq i \leq m \\ & h_j(x) = 0 & \forall\ 1 \leq j \leq p \end{array}$$

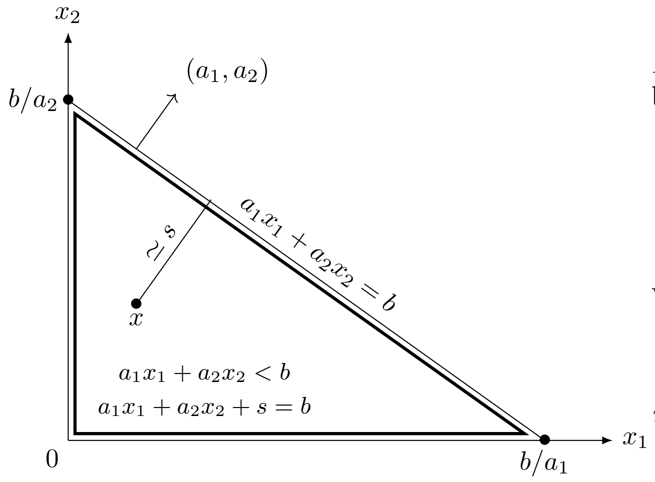
then, there exists a program (P') equivalent to (P) with the following form

$$\begin{array}{lll} \min & f(x) & \\ s.c. & h_j(x) = 0 & \forall\ 1 \leq j \leq p \\ & x_i \geq 0 & \forall\ 1 \leq i \leq n \end{array}$$

This program is equivalent in the sense that an optimal solution of the first program can easily be deduced from an optimal solution of the second, and conversely. A method we can use to produce this second program is as follows:

- If $g_i(x) \leq 0$ then add a slack variable $s_i \geq 0$ and set $h_i(x) = g_i(x) + s_i = 0$.
- If $x_i \in \mathbb{R}$ then add $x_i^+ \geq 0$ and $x_i^- \geq 0$ and replace x_i by $x_i^+ + x_i^-$ in the program.
- If $g_i(x) = -x_i \leq 0$ then **do nothing**.
- If $g_i(x) = x_i \leq 0$ then replace x_i by $-x_i$ in the program.

The concept of the gap variable can be easily explained with drawings:



We use the following notations

$$h_j(x) = \sum_{k=1}^{n}(a_{jk}x_k) - b_j = 0\ \forall j \in [\![1;p]\!]$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pn} \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}$$

We then have $Ax = b$.

3.1 Reduced gradient computation

As in the projected gradient, we assume that we start at a point $x \in S$ that is given. As explained earlier, we will use the equalities to express some of the variables in terms of the others. We thus choose a subset $B \subset [\![1;n]\!]$ of variables of size p, and the remaining variables will be N. We say that B corresponds to the *basic* variables and N to the *non-basic* variables. We denote

- $A_B =$ the columns of A for which the index is in B.
- $A_N =$ the columns of A for which the index is in N.
- $x_B =$ the variables of x for which the index is in B.
- $x_N =$ the variables of x for which the index is in N.
- $\nabla f_B =$ the coeffs of $\nabla f(x)$ for which the index is in B.
- $\nabla f_N =$ the coeffs of $\nabla f(x)$ for which the index is in N.

We can rewrite A, x and $\nabla f(x)$ this way:

$$A = \begin{pmatrix} A_B & A_N \end{pmatrix}, x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}, \nabla f(x) = \begin{pmatrix} \nabla f_B \\ \nabla f_N \end{pmatrix}$$

On suppose que le rang de A est p.

Lemma 3.1. *If A_B is invertible, it is possible to rewrite $f(x)$ with only x_N with*

$$x_B = A_B^{-1} \cdot b - A_B^{-1} \cdot A_N \cdot x_N$$

Proof. We simply need to express the product Ax in block matrix form.

$$\begin{pmatrix} A_B & A_N \end{pmatrix} \cdot \begin{pmatrix} x_B \\ x_N \end{pmatrix} = b$$

$$A_B x_B + A_N x_N = b$$

We obtain the result by multiplying both sides by A_B^{-1}. □

Let $\bar{f} : \mathbb{R}^{n-p} \to \mathbb{R}$ such that $\bar{f}(x_N) = f(x_B, x_N)$. The *reduced gradient* is the gradient of $\bar{f}$.

$${}^t\nabla \bar{f}(x_N) = -\,{}^t\nabla f_B \cdot A_B^{-1} \cdot A_N + {}^t\nabla f_N$$

3.2 Direction computation

This gradient provides a direction to follow in order to minimize $\bar{f}$ (one simply needs to follow the opposite of the reduced gradient). However, unlike the projected gradient, which took into account all active constraints, the reduced gradient does not consider the nonnegativity constraints of the variables. Therefore, to obtain a valid direction d, it is necessary to ensure that if $x_i = 0$, then $d_i \geq 0$.

Given $\nabla\bar{f}(x_N)$, we define the direction d, divided in two parts, d_B and d_N, this way

$$\forall j \in N, d_j = \begin{cases} 0 & \text{if } \nabla\bar{f}(x_N)_j > 0 \text{ and } x_j = 0 \\ -\nabla\bar{f}(x_N)_j & \text{otherwise} \end{cases}$$

The direction d_N sets to 0 all the components that would cause a variable x_N to become negative when following d_N. To obtain d_B, simply refer to the constraint $Ax = b$.

$$d_B = -A_B^{-1} \cdot A_N \cdot d_N$$

Lemma 3.2. *For all $\alpha \in \mathbb{R}$, $A \cdot (x + \alpha \cdot d) = b$.*

Proof.

$$\begin{aligned} A \cdot (x + \alpha \cdot d) &= A \cdot x + \alpha \cdot A \cdot d \\ &= b + \alpha \cdot (A_B \cdot d_B + A_N \cdot d_N) \\ &= b + \alpha \cdot (A_B \cdot (-A_B^{-1} \cdot A_N \cdot d_N) + A_N \cdot d_N) \\ &= b \end{aligned}$$

□

We can therefore follow the direction d without the risk of violating the constraint $Ax = b$. It should be noted that this is not the case for positivity constraints, which can always be unsatisfied for a basic variable that follows the direction d_B. But we will discuss this point later.

Before going any further, it is important to ask an important question here: does the choice of B and N matter? Is the obtained direction always the same regardless of this choice? We can answer negatively in three different ways:

- We already know that A_B must necessarily be invertible to perform this entire process.
- Even assuming A_B is invertible for any choice of basis B, the fact that we reduce certain components of d_N to 0 while not doing so for d_B necessarily implies a break in symmetry. For example, suppose we only have 2 variables x_1 and x_2 with $x_1 = 0$, $x_2 > 0$. And suppose the reduced gradient when $B = (1), N = (2)$ indicates $\nabla\bar{f}(x_2) = (10)$. We then obtain a direction $d_N = (-10)$. We calculate d_B and get a value, let's assume for the example $d_B = (-5)$. We can quickly realize that choosing $N = (1), B = (2)$ cannot lead to the same direction. Indeed, for that to happen, we would have to have $\nabla\bar{f}(x_1) = (5)$. However, in this case, we have $d_N = (0)$ because $\nabla\bar{f}(x_1) > 0$ and $x_1 = 0$. Since $d_N = 0$, then $d_B = -A_B^{-1} \cdot A_N \cdot d_N = 0$. We obtain here a null direction.
- Lastly, let's assume that A_B is invertible regardless of the basis and all variables are strictly positive. Even in this case, it is also possible to have a different direction depending on the basis. Here's an example. Let's suppose we want to solve the following problem:

$$\begin{aligned} \min \quad & x_1^2 \\ s.c. \quad & x_1 + x_2 = x_3 \\ & x_i \geq 0 \qquad \forall\, 1 \leq i \leq 3 \end{aligned}$$

$A = (1; 1; -1)$, $b = (0)$

We assume we start at $x = (5, 5, 10)$. First, let's consider $B = (1), N = (2, 3)$. We rewrite f as a function of x_2 and x_3. We obtain the function $\bar{f}$, the reduced gradient, and the following directions.

$$\begin{aligned} \bar{f}(x_2, x_3) &= (x_3 - x_2)^2 \\ \nabla\bar{f}(x_2, x_3) &= \begin{pmatrix} -2x_2 \\ 2x_3 \end{pmatrix} = \begin{pmatrix} -10 \\ 10 \end{pmatrix} \\ d_N &= \begin{pmatrix} 10 \\ -10 \end{pmatrix} \end{aligned}$$

Il faut donc augmenter x_2 et diminuer x_3 de la même quantité. Pour savoir quelle direction prend x_1 lorsqu'on suit d_N, on calcule d_B:

$$\begin{aligned} d_B &= -A_B^{-1} A_N d_N \\ &= -(1)(1; -1) d_N = (-20) \end{aligned}$$

We reorder the variables of the direction to place x_1, x_2, x_3 in this order. We therefore obtain

$$d = \begin{pmatrix} -20 \\ 10 \\ -10 \end{pmatrix}$$

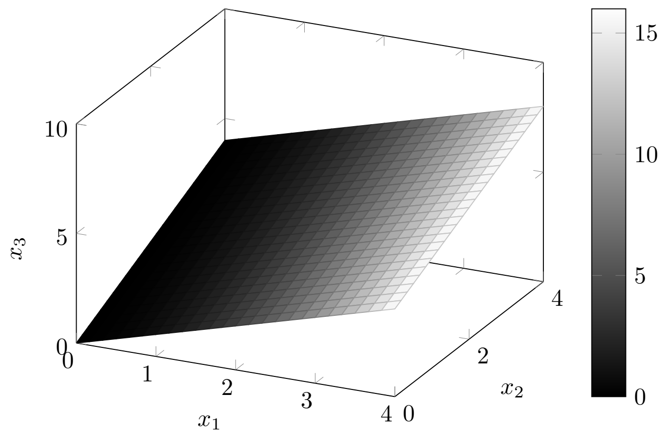
Let us now consider $B = (3), N = (1, 2)$. This time, rewriting f as a function of x_1 and x_2 is straightforward, we have $f = \bar{f}$. We obtain the reduced gradient and the following directions.

$$\begin{aligned} \bar{f}(x_1, x_2) &= x_1^2 \\ \nabla\bar{f}(x_1, x_2) &= \begin{pmatrix} 2x_1 \\ 0 \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \end{pmatrix} \\ d_N &= \begin{pmatrix} -10 \\ 0 \end{pmatrix} \\ d_B &= -A_B^{-1} A_N d_N \\ &= -(-1)(1; 1) d_N = (-10) \end{aligned}$$

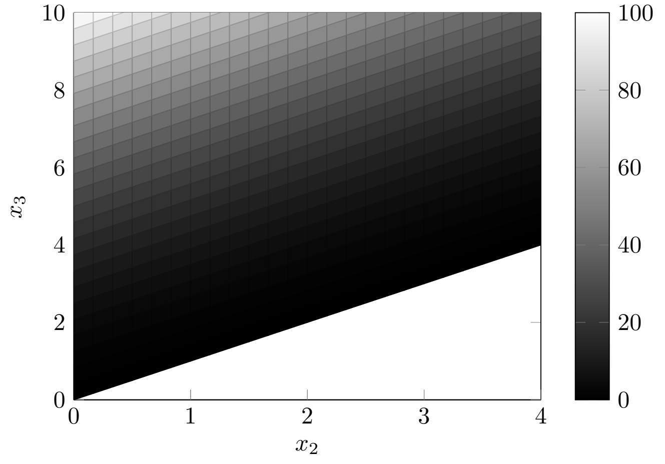
We then get

$$d = \begin{pmatrix} -10 \\ 0 \\ -10 \end{pmatrix}$$

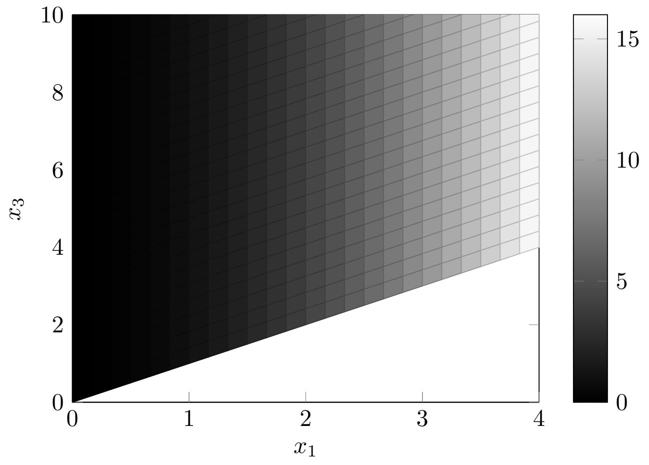
We observe a similar trend but not the same direction. The same phenomenon repeats if we set $B = (2)$ and $N = (1, 3)$. This is quite counterintuitive because $\bar{f}$ and f are indeed the same function, just written differently. However, focusing on certain variables rather than others somewhat prevent us from having the overall vision of the function. Unlike the projected gradient, which projects onto certain constraints, here it is as if we are projecting the function onto a subspace and looking for a direction within that subspace. In our example, the view in space would provide the graph below. The color indicates $f(x)$, which here depends only on x_1.



Calculating the function f in terms of x_2 and x_3 gives the same diagram viewed from the side (on the plane corresponding to these two variables).



The plane is tilted because there are certain forbidden pairs for (x_2, x_3). Indeed, if for example $x_2 = 4$ and $x_3 = 0$, then $x_1 = -4$, which is not allowed. By closely examining the color gradient, we can see that to decrease $f(x)$ as much as possible, we need to increase x_2 and decrease x_3. Now, if we look at another projection.



In this view, to decrease $f(x)$, one must go left. Be careful not to be influenced by the view of the plane that seems tilted (this slope is only due to the forbidden values of (x_1, x_3)). By closely examining the color gradient, we see that the function is constant with respect to x_3, so the opposite of the gradient goes to the left. The direction along x_3 is therefore zero.

It is therefore necessary to carefully choose the basis B to converge more quickly towards the optimal solution. This choice is not trivial and there is no miracle method to obtain it.

The direction d is now chosen, two questions remain unanswered: what happens if $d = 0$ and what should be done if a basic variable becomes negative while following the direction d?

Lemma 3.3. *If $d = \vec{0}$ then, the (KKT) conditions are satisfied.*

Proof. If $d = \vec{0}$ then $d_N = d_B = \vec{0}$. By the definition of d_N, we have $d_j = 0$ so ${}^t\nabla \bar{f}(x_N)_j = 0$ or ${}^t\nabla \bar{f}(x_N)_j > 0$ and $x_j = 0$. In both cases, ${}^t\nabla \bar{f}(x_N)_j \geq 0$. In order to satisfy the (KKT) conditions, one needs to.

$$\exists \lambda_i, \mu_j \in \mathbb{R}$$
$$\lambda_i \geq 0 \; \forall \; i \in I(x)$$
$$\nabla f(x) + \sum_{i \in I(x)} \lambda_i \nabla g_i(x) + \sum_{j=1}^{p} \mu_j \nabla h_j(x) = 0$$

Equivalently

$$\exists \lambda_i, \mu_j \in \mathbb{R}$$
$$\lambda_i \geq 0 \ \forall\ i \in [\![1;m]\!] \qquad (1)$$
$$\lambda_i g_i(x) = 0 \ \forall\ i \in [\![1;m]\!] \qquad (2)$$
$$\nabla f(x) + \sum_{i\in I} \lambda_i \nabla g_i(x) + \sum_{j=1}^{p} \mu_j \nabla h_j(x) = 0 \qquad (3)$$

Here, the inequality constraints are the positivity constraints. $g_i(x) = -x_i \leq 0$. Therefore, $m = n$, we have one variable λ_i for each variable x_i. Moreover

$$\nabla g_i(x) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Moreover, by definition of the matrix A, $\nabla h_j(x)$ is the j-th row of the matrix A. We can thus rewrite the condition (3) as follows:

$$\nabla f(x) - \lambda + {}^tA\mu = 0$$

We perform a block decomposition according to the basis B and the non-basic variables N. Since we have a value λ_i for each variable x_i, we can also decompose this vector according to B and N. Note that this is not the case for μ, which are associated with the equality constraints and not with the variables.

$$\begin{pmatrix} \nabla f_B \\ \nabla f_N \end{pmatrix} - \begin{pmatrix} \lambda_B \\ \lambda_N \end{pmatrix} + \begin{pmatrix} {}^tA_B \\ {}^tA_N \end{pmatrix} \mu = 0$$

We set $\lambda_B = 0$, and we see if this is sufficient to find a valid value for λ_N and μ. Thus, we first have

$$\nabla f_B + {}^tA_B\mu = 0$$
$${}^t\mu = -{}^t\nabla f_B A_B^{-1}$$

We get a value for μ. If we look at the other equality, we have

$$\nabla f_N - \lambda_N + {}^tA_N\mu = 0$$
$$\nabla f_N - \lambda_N + {}^tA_N {}^t(-{}^t\nabla f_B A_B^{-1}) = 0$$

Then

$$-{}^t\nabla f_B \cdot A_B^{-1} \cdot A_N + {}^t\nabla f_N = {}^t\lambda_N$$
$${}^t\nabla \bar{f}(x_N) = {}^t\lambda_N$$

We saw at the beginning that ${}^t\nabla \bar{f}(x_N)_j \geq 0$ for all $j \in N$, so ${}^t\lambda_N \geq 0$. Since $\lambda_B = 0$, the condition (1) of (KKT) is satisfied.

To conclude, if $g_i(x) = 0$ then $\lambda_i g_i(x) = 0$. If $i \in B$ then $\lambda_i = 0$. If $i \in N$ and $g_i(x) < 0$ then $x_i > 0$ and thus ${}^t\nabla \bar{f}(x_N)_i = d_i$. However, since $d_i = 0$ by assumption and since ${}^t\nabla \bar{f}(x_N)_i = \lambda_i$, then λ_i is zero. Therefore, $\lambda_i g_i(x) = 0$. Thus, the condition (2) of (KKT) is also satisfied. We indeed verify all the conditions of (KKT) with the values of λ and μ obtained. □

3.3 Follow the direction and change of basis

If the direction is zero, in this case, we stop the algorithm, we have reached our objective. If the direction is non-zero, as in the case of the projected gradient, we will move in the direction d until we reach the optimum in that direction.

$$\alpha_1 = \max_{0\leq\alpha}(x + \alpha \cdot d \in S)$$
$$\alpha_2 = \arg \min_{0\leq\alpha\leq\alpha_1} (f(x + \alpha \cdot d))$$

However, unlike the projected gradient, we can have $\alpha_1 = 0$. This is due to the fact that a basic variable can be zero and the direction to follow for this variable is negative. We must then proceed with a basis change (changing the variables in N and B). For this, we will make one additional assumption in the problem:

Definition 5 (Non-degeneration hypothesis)**.** The hypothesis assumes that, whatever the basis B is, if A_B is invertible, then $A_B^{-1} \cdot b > 0$.

Let $y = x + \alpha d$. If there exists $s \in B$ such that $y_s = 0$; and if we keep the basis for the next iteration, then the next value of α_1 may be 0, we may be stuck.

Lemma 3.4. *If there exists $s \in B$ such that $y_s = 0$, then, under the non-degeneration hypothesis, there exists $r \in [\![1;n]\!]$ such that $y_r \neq 0$ and $A_{B\cup\{r\}\backslash\{s\}}$ is invertible.*

Proof. Let A_i be the i-th column of A. Let $r \in N$ and $s \in B$ such that $y_s = 0$. The matrix A_B contains column A_s. The matrix A_N contains column A_r. Finally, let $B' = B \cup \{r\}\backslash\{s\}$.

Consider the matrix $A_{B'}$ which is equal to the matrix A_B where column A_s is replaced by A_r.

$$A_B{}^{-1} \cdot A_{B'} = \begin{pmatrix} 1 & 0 & \cdots & q_1 & \cdots & 0 \\ 0 & 1 & \cdots & q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & \cdots & \cdots & p & \cdots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \cdots \\ 0 & \cdots & \cdots & q_n & \cdots & 1 \end{pmatrix}$$

where

$$A_B{}^{-1} \cdot A_r = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ \pi \\ \vdots \\ q_n \end{pmatrix}$$

π is called the pivot of r. Note that the position and value of this pivot in the matrix depend on the position of A_s in A_B. In particular, the position of the pivot of r does not depend on r. If s is in the i^{th} column, then π is in the i^{th} row. The column containing π is simply calculated with $A_B^{-1}A_r$. This matrix is invertible if and only if $\pi \neq 0$. Suppose $\pi \neq 0$, then it can be easily verified that.

$$(A_B{}^{-1} \cdot A_{B'})^{-1} = \begin{pmatrix} 1 & 0 & \cdots & \frac{-q_1}{\pi} & \cdots & 0 \\ 0 & 1 & \cdots & \frac{-q_2}{\pi} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & \cdots & \cdots & \frac{1}{\pi} & \cdots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \cdots \\ 0 & \cdots & \cdots & \frac{-q_n}{\pi} & \cdots & 1 \end{pmatrix}$$

Finally, we obtain $A_{B'}^{-1}$ by $(A_B{}^{-1} \cdot A_{B'})^{-1} \cdot A_B{}^{-1}$. To prove the lemma, it remains to show that there exists a variable $r \in N$ such that $y_r \neq 0$ and such that $\pi \neq 0$.

We denote in the following $B = \{s_1, s_2, \ldots, s, \ldots, s_p\}$ and $N = \{r_1, r_2, \ldots, r_{n-p}\}$. We also denote π_i as the pivot of r_i. According to the non-degeneracy hypothesis, we know that

$$A_B^{-1} \cdot b > 0$$

By lemma 3.1

$$y_B + A_B^{-1} \cdot A_N \cdot y_N > 0$$

Since, in the vector $A_B^{-1} \cdot A_r$, the pivot of r is always on the row corresponding to s, we have

$$\begin{pmatrix} y_{s_1} \\ y_{s_2} \\ \vdots \\ y_s \\ \vdots \\ y_{s_p} \end{pmatrix} + \begin{pmatrix} q_{r_1}1 & q_{r_2}1 & \cdots & q_{r_{n-p}}1 \\ q_{r_1}2 & q_{r_2}2 & \cdots & q_{r_{n-p}}2 \\ \vdots & \vdots & \cdots & \vdots \\ \pi_1 & \pi_2 & \cdots & \pi_{n-p} \\ \vdots & \vdots & \cdots & \vdots \\ q_{r_1}p & q_{r_2}p & \cdots & q_{r_{n-p}}p \end{pmatrix} \cdot \begin{pmatrix} y_{r_1} \\ y_{r_2} \\ \vdots \\ y_{r_{n-p}} \end{pmatrix} > 0$$

We consider in this equality the line corresponding to s.

$$y_s + \sum_{i=1}^{n-p} \pi_i y_{r_i} > 0$$

By hypothesis

$$0 + \sum_{i=1}^{n-p} \pi_i y_{r_i} > 0$$

If p_i is zero for all i such that y_{r_i} is non-zero, there is a contradiction. Thus, there exists i such that $y_{r_i} \neq 0$ and $\pi_i \neq 0$. □

Remark 3. The beginning of the proof of lemma 3.4 provides a technique to compute $A_{B'}$, knowing $A_B{}^{-1}$, without any further complicated matrix inverse calculations. It is sufficient to compute $A_B^{-1}A_r$ to deduce $(A_B{}^{-1} \cdot A_{B'})^{-1}$, and then multiply this matrix by $A_B{}^{-1}$.

This last lemma proves that one can always replace a *dangerous* variable with a *safe* variable in the base, which helps avoid stagnation in the algorithm.

The pseudo-code of the algorithm is the following:

```
1: Find $x \in S$ and $B$ such that $A_B$ is invertible
2: **loop**
3:     ${}^t\nabla\bar{f}(x_N) \leftarrow -{}^t\nabla f_B \cdot A_B^{-1} \cdot A_N + {}^t\nabla f_N$
4:     **for** $j \in N$ **do**
5:         **If** $\nabla\bar{f}(x_N)_j > 0$ and $x_j = 0$ **then** $d_j \leftarrow 0$ **else** $d_j \leftarrow -\nabla\bar{f}(x_N)_j$
6:     $d_B \leftarrow -A_B^{-1}A_N d_N$
7:     **If** $d = 0$ **then return** $x$
8:     $\alpha_1 \leftarrow \max_{0\leq\alpha}(x + \alpha \cdot d \geq 0)$
9:     $\alpha_2 \leftarrow \arg\min_{0\leq\alpha\leq\alpha_1}(f(x + \alpha \cdot d))$
10:    $x \leftarrow x + \alpha_2 \cdot d$
11:    **if** $\exists s$ such that $x_s = 0$ **then**
12:        **for** $r \in N$ by decreasing order of $x_r$ **do**
13:            **if** $A_{B\cup\{r\}\backslash\{s\}}$ is invertible **then**
14:                $B \leftarrow B \cup \{r\}\backslash\{s\}$
15:                Restart the loop line 2
```