CIMPA Course 0 :

Introduction to (continuous) optimization

Part 1 : theoretical background (monday, 14.30 → 16:00 local time)

(-6h in France)

Includes { main definitions and vocabulary
optimality conditions (constrained / unconstrained case)

Part 2 : numerical backgroung (tuesday, 14.00 → 15.30)

Includes { main descent algorithms
examples of numerical implementation (on the Rosenbrock function)

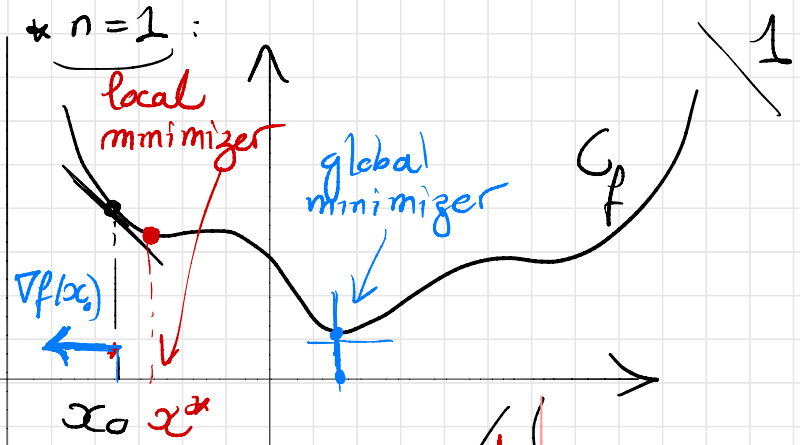
PART 2 : Numerical background

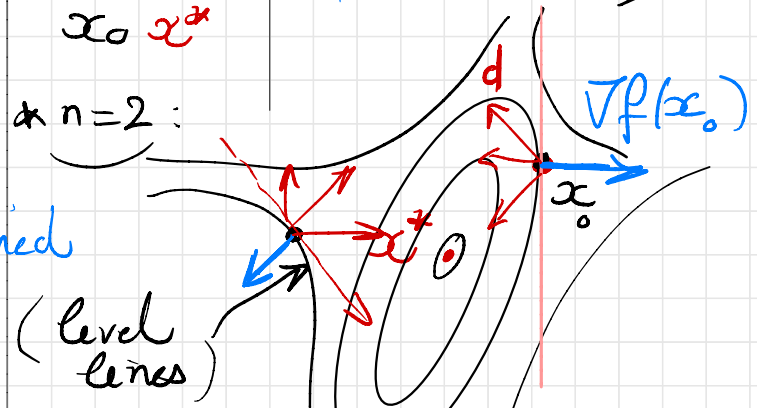
2.1) General principal of descent methods

Min $f(x)$ (P_0) where $x \in \mathbb{R}^n$ (unconstrained)

$f : \mathbb{R}^n \to \mathbb{R}$ is C^1 (or C^2)

(or $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$) (constrained)





* Objective : build a sequence of approximations $(x_k)_{k \in \mathbb{N}}$ that converges to x^*.

* A descent method imposes that:

$$\forall k \in \mathbb{N}, \; f(x_{k+1}) \leq f(x_k)$$

(the approximation improves)

* It is recursively defined by:

- a descent direction: d_k
- a stepsize: $\alpha_k > 0$

and writes as: step (descent direction)

$$x_{k+1} = x_k + \alpha_k d_k$$

* Gradient type methods corresponds to:

$$d_k = -\nabla f(x_k)$$

(it is a descent direction, as $\langle d_k, \nabla f(x_k) \rangle = -\|\nabla f(x_k)\|^2 < 0$ if $\nabla f(x_k) \neq 0$)

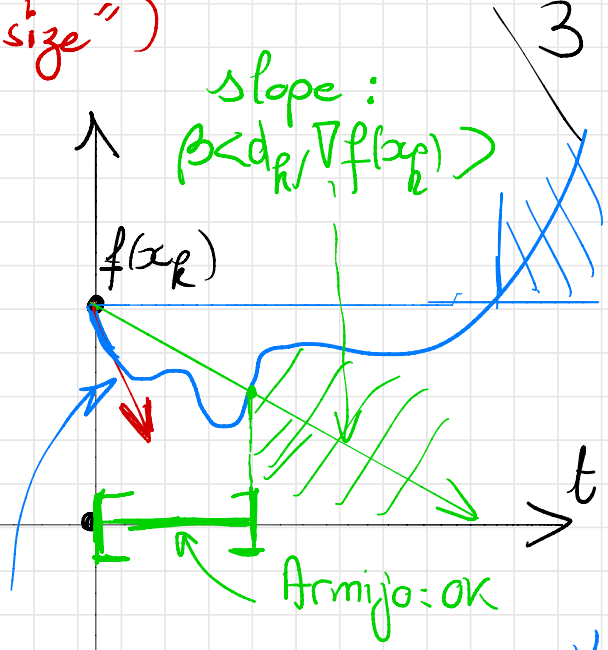
2.2) Linesearch strategies ("find the good step size")

* The Armijo condition:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \beta \alpha_k \langle d_k, \nabla f(x_k) \rangle$$

(x_{k+1} ; $\langle d_k, \nabla f(x_k) \rangle < 0$)

(where $\beta \in]0,1[$ is a fixed parameter) ensures a sufficient decrease, with respect to the step size, between two iterations.

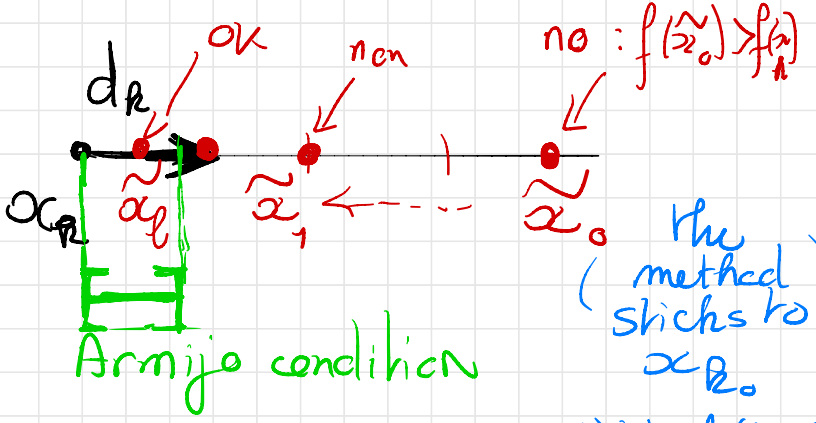


shape of f from x_k in direction d_k : $t \overset{h_k}{\longmapsto} f(x_k + t d_k)$

* Derivative at $t = 0$:

$h'(0) = \langle d_k, \nabla f(x_k) \rangle$ (< 0)

* A backtracking method to find a new point satisfying the Armijo condition, ensures a sufficiently large step size.



* With this strategy and a gradient direction) the sequence of approximations $(x_k)_{k\in\mathbb{N}}$ satisfies either:

(i) $\nabla f(x_{k_0}) = 0$ for $k_0 \in \mathbb{N}^*$

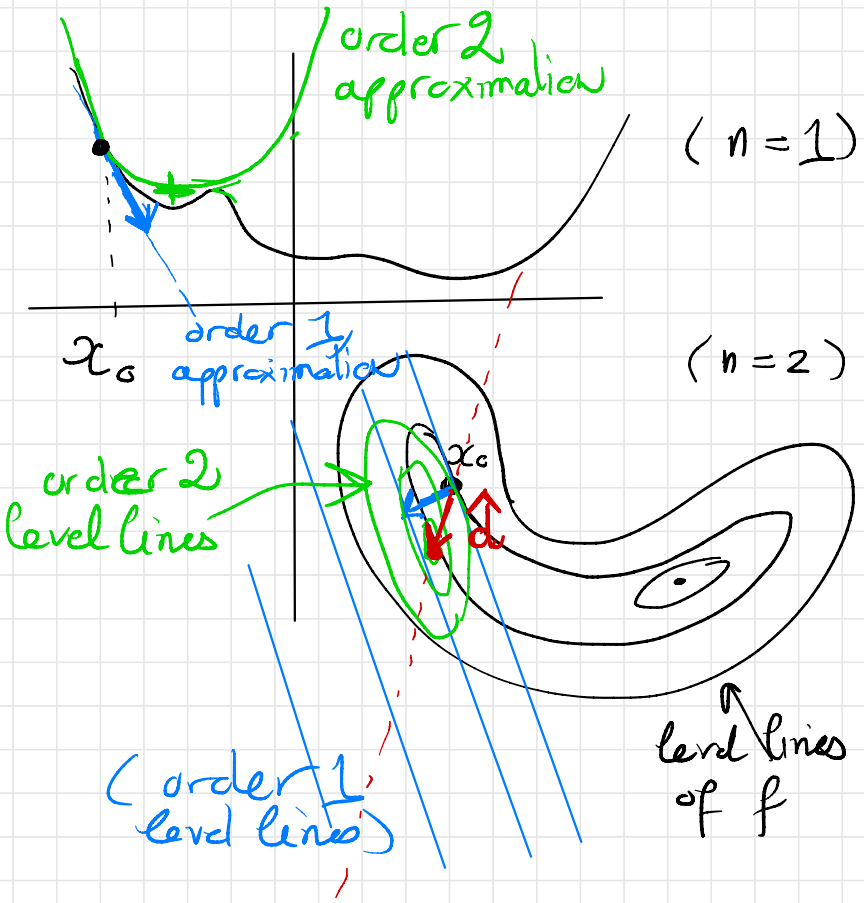
(ii) $\lim_{k\to+\infty} f(x_k) = -\infty$

or:

(iii) $\lim_{k\to+\infty} \nabla f(x_k) = 0$

(the method sticks to x_{k_0} "ideal" case)

2.3 Second order descent methods



* Second order approximation of f at point x_k :

$$m_Q(x_k + h) = f(x_k) + \langle \nabla f(x_k), h \rangle + \langle \frac{1}{2} h, B_k h \rangle$$

($B_k = H(f)(x_k)$ for instance)

* A possible descent direction is:

$$d_k = - B_k^{-1} \cdot \nabla f(x_k)$$

(only if $B_k \gg 0$)

* Newton method :

$$x_{k+1} = x_k - Hf(x_k)^{-1} \cdot \nabla f(x_k)$$

* Quasi-Newton method :

$$x_{k+1} = x_k - B_k^{-1} \cdot \nabla f(x_k)$$

with $B_k \approx Hf(x_k)$

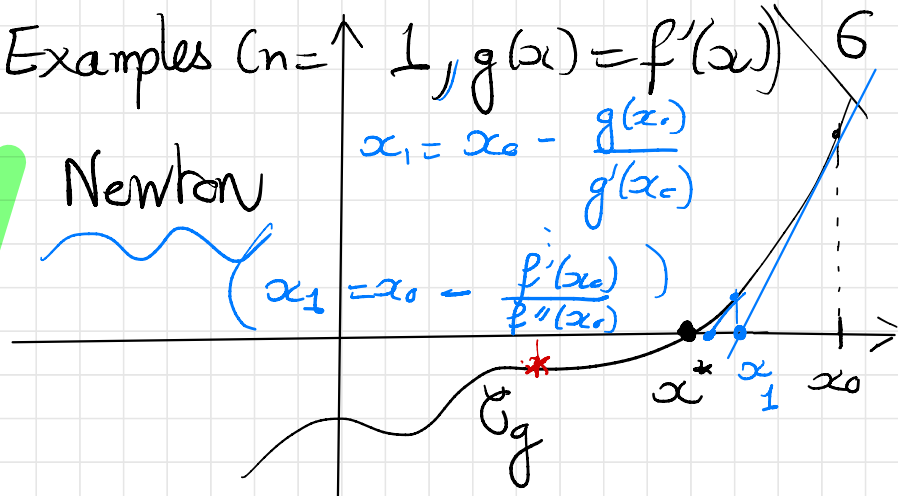
$\rightsquigarrow$ BFGS method

Examples ($n = 1$, $g(x) = f'(x)$)

Newton

$x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}$

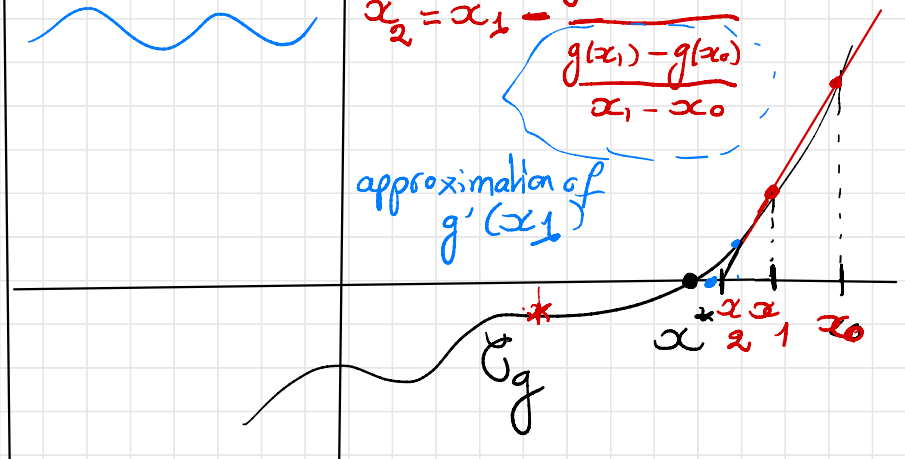
$\left(x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} \right)$



Quasi-Newton

$x_2 = x_1 - \frac{g(x_1)}{\frac{g(x_1) - g(x_0)}{x_1 - x_0}}$

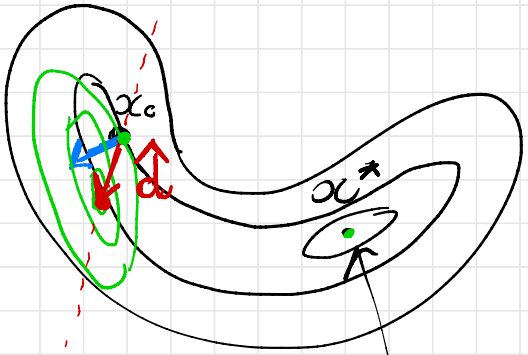
approximation of $g'(x_1)$



2.4 Implementation with Scilab

→ Test case : the Rosenbrock function :

$$f(x,y) = 100(y-x^2)^2 + (x-1)^2$$



one (global) minimizer : (1,1)

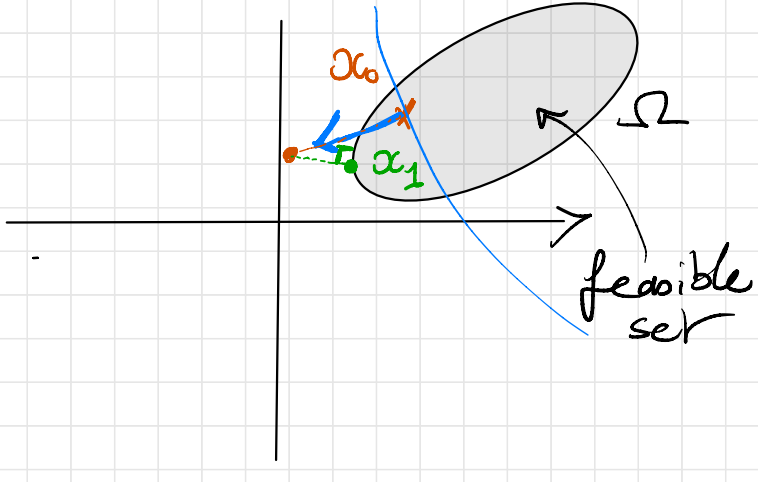
→ descent methods :

* order 1 : gradient
* order 2 : Newton
* BFGS

2.5 Constraints handling

2.5.1) Gradient projection method

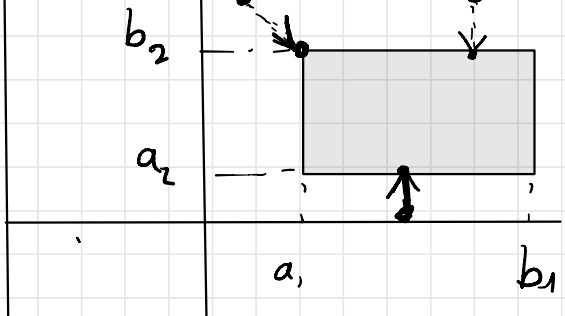
$$x_{k+1} = P_{\Omega}(x_k - \alpha_k \nabla f(x_k))$$



* Main drawback: only in rare cases, the projection function is unknown.

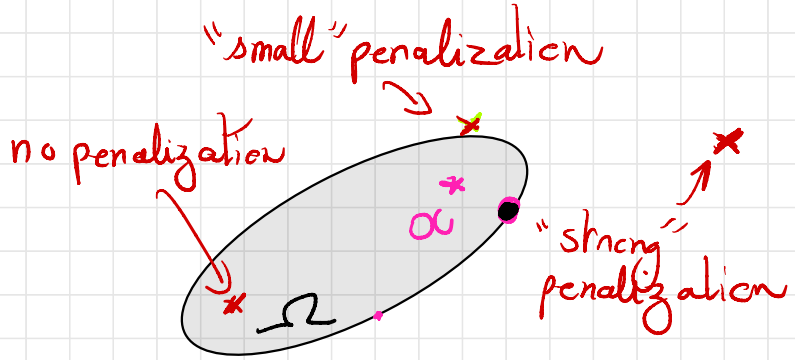
* Favorable case: Ω hypercube:

$$\Omega = \{ x \in \mathbb{R}^n \text{ s.t. } a_i \le x_i \le b_i \}$$
$$= [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_n, b_n]$$



$$P_{\Omega}((x_1, x_2)) = \left(\text{med}(x_1, a_1, b_1), \text{med}(x_2, a_2, b_2) \right)$$

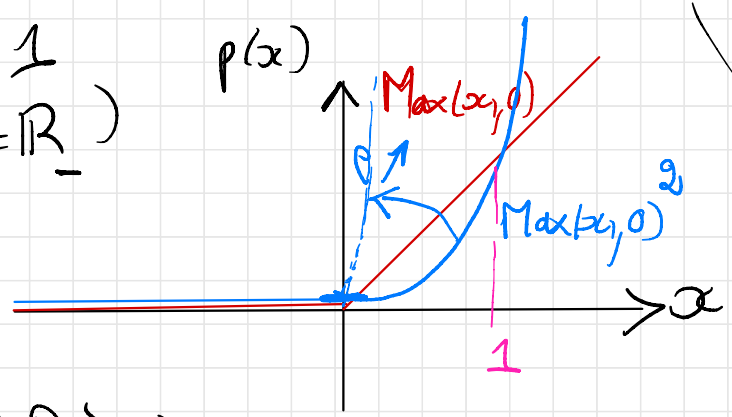
2.5.2 Penalization method



$$\text{Min}_{x \in \mathbb{R}^n} \left(f(x) + \rho\, p(x) \right) \quad (P_\rho)$$

with $p : \begin{pmatrix} \mathbb{R}^n \longrightarrow \mathbb{R} \\ x \longmapsto \begin{cases} 0 \text{ if } x \in \Omega \\ > 0 \text{ else} \end{cases} \end{pmatrix}$

($n = 1$, $\Omega = \mathbb{R}_-$)



and $\rho > 0$.

↖ "intensity" of the penalization

2.5.3) Dual method

Recall the Lagrangian function:

$$\mathcal{L}(x,\lambda,\mu) = f(x) + \sum_{i=1}^{p} \lambda_i c_{E_i}(x) + \sum_{j=1}^{q} \mu_j c_{I_j}(x)$$

(λ: primal variables, μ: dual variables; $\lambda_i \in \mathbb{R}$, $\mu_j \in \mathbb{R}_+$)

$$\Omega = \{ x \in \mathbb{R}^n \,/\, c_E(x) = 0 \text{ et } c_I(x) \leq 0 \}$$

* (P_0) can be rewritten as:

$$\inf_{x \in \Omega} f(x) = \inf_{x \in \mathbb{R}^n} \left(\sup_{(\lambda,\mu) \in \mathbb{R}^p \times \mathbb{R}^q_+} \mathcal{L}(x,\lambda,\mu) \right) \quad (P)$$

* The dual problem writes as:

$$\sup_{(\lambda,\mu) \in \mathbb{R}^p \times \mathbb{R}^q_+} \left(\inf_{x \in \mathbb{R}^n} \mathcal{L}(x,\lambda,\mu) \right) \quad (D)$$

$$= \sup_{(\lambda,\mu) \in X^*} f^*(\lambda,\mu)$$

where

$$X^* = \{ (\lambda,\mu) \in \mathbb{R}^p \times \mathbb{R}^q_+ \,/\, f^*(\lambda,\mu) > -\infty \}$$

is the feasible set of the dual problem.

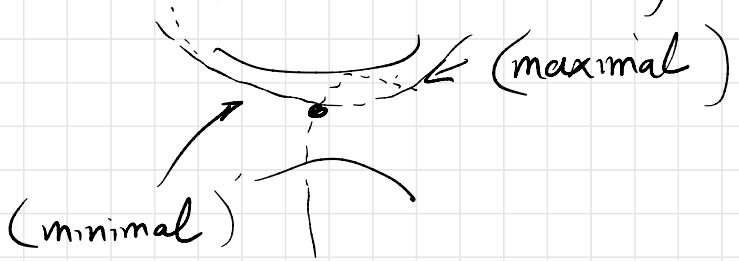
* Definition : $(\bar{x}, \bar{\lambda}, \bar{\mu})$ is a saddle point of $\mathcal{L}$ if :

(i) $\forall (\lambda, \mu) \in \mathbb{R}^p \times \mathbb{R}^q$,

$$\mathcal{L}(\bar{x}, \lambda, \mu) \leq \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu})$$

(ii) $\forall x \in \mathbb{R}^n$,

$$\mathcal{L}(x, \bar{\lambda}, \bar{\mu}) \geq \mathcal{L}(\bar{x}, \bar{\lambda}, \bar{\mu})$$



* Theorem : $(\bar{x}, \bar{\lambda}, \bar{\mu})$ is a saddle point of $\mathcal{L}$ iff

(i) $\bar{x}$ is a solution of (P)

(ii) $(\bar{\lambda}, \bar{\mu})$ is a solution of (D)

(iii) $\inf_{x} (\sup_{\lambda, \mu} \mathcal{L}(x, \lambda, \mu)) = \sup_{(\lambda, \mu)} \inf_{x} (\mathcal{L}(\ldots))$

(no duality step)

* Dual-type method:

find a saddle point of $\mathcal{L}$

by successively:

→ minimize

$$\left(\begin{array}{l} x \mapsto \mathcal{L}(x, \lambda_k, \mu_k) \\ \mathbb{R}^n \to \mathbb{R} \end{array}\right)$$

→ maximize

$$\left(\begin{array}{l} \mathbb{R}^p \times (\mathbb{R}_+^q) \to \mathbb{R} \\ (\lambda, \mu) \mapsto \mathcal{L}(x_{k+1}, \lambda, \mu) \end{array}\right)$$

with a projection gradient:

$$\lambda_{k+1} = \lambda_k + \rho_2 \nabla_\lambda \mathcal{L}(x_{k+1}, \lambda_k, \mu_k)$$

$$\mu_{k+1} = \max(0, \mu_k + \rho_2 \nabla_\mu \mathcal{L}(x_{k+1}, \lambda_{k+1}, \mu_k))$$

(max: projection)

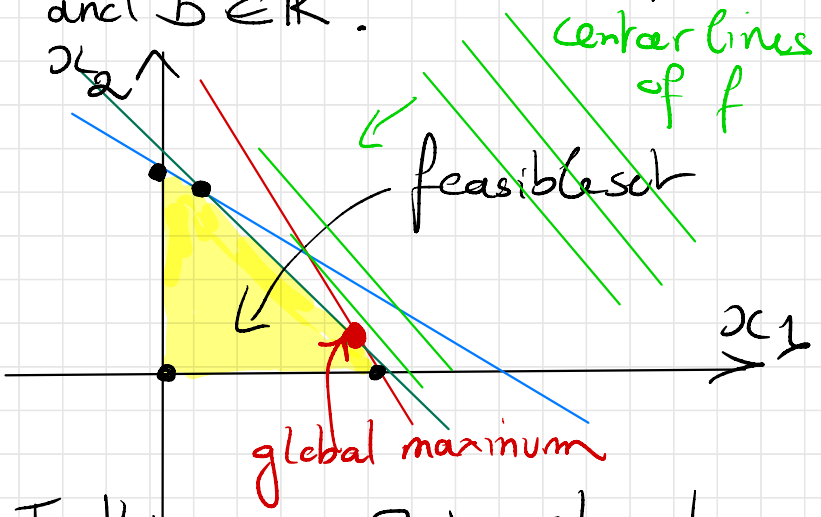
2.5.4 A particular case: linear programming

In this case (maximization) $\sum_{i=1}^{n} c_i x_i$

* f is linear: $f(x) = \langle c, x \rangle$
* C_E and C_I are affine:

$$\Omega = \{x \in (\mathbb{R}_+)^n \mid Ax \leq b\}$$

where $c \in \mathbb{R}^n$, $A \in \mathcal{M}_{m,n}(\mathbb{R})$, and $b \in \mathbb{R}^m$.



In this case Ω is a closed convex polyhedron (bounded or not).

It can be proven that if $(\mathcal{P})$ has a solution, then there exists a solution on the vertices of Ω:

$V(\Omega) = \{x \in \Omega \,/\, \Omega \setminus \{x\} \text{ is convex}\}$

* To find this solution, the simplex algorithm consists to explore vertices along edges until the function can no longer increase //