1ère partie : Optimisation linéaire

Exemples et définitions

Nous commençons ce cours par **optimisation (programmation) linéaire** *(OL)* – la partie de la programmation mathématique la plus simple et la plus souvent utilisée en applications.

Exemple

$$
\begin{array}{ll}
\text{minimiser} & 350x_1 + 300x_2 \\
\text{sous contraintes} & x_1 \geq 0, \;\; x_1 \geq 0 \\
& 9x_1 + 6x_2 \leq 1566 \\
& 12x_1 + 16x_2 \leq 2880
\end{array}
$$

Pourquoi c'est bien ?

- Les raisons de popularité de l'OL sont :
  - — "pouvoir d'expression" interessant – les dépendances linéaires sont souvent des approximation satisfaisantes des dépendances actuelles non-linéaires.
  - — Modèles linéaires sont faciles à specifier – pour "remplir" une fonction linéaire de 1000 variables on a besoin de définir 1000 coefficients ; pour spécifier une fonction quadratique on a besoin de 501 500 coefficients ! Ainsi, il est possible de construire des modèles avec beaucoup de variables et de contraintes
  - — existence d'une théorie complète et élégante
  - — dès le début, la méthodologie de OL était accompagnée par une technique de résolution extrêmement puissant - algorithme de simplex qui a fait de OL un outil de travail.

    Les "moteurs d'optimisation" actuels sont capables de résoudre des problèmes OL avec $10^4 \div 10^6$ variables et contraintes

- Dans cette partie du cours on s'intéressera de

  — *Modélisation OL*, y compris des exemples "instructifs" des modèles et leur "calcul" – ensemble d'outils pour reconnaître la possibilité de formuler le problème comme problème OL
  — *Théorie de l'OL* – la géométrie des programmes linéaires, existence et caractérisation des solutions optimales et dualité ;
  — *Applications* de ces outils pour résoudre quelques problèmes d'optimisation linéaire pertinentes en probabilité et statistique

Modèle d'optimisation linéaire

Programme OL (programme linéaire PL)

$$\begin{array}{rl} \text{minimiser} & \sum_{i=1}^{n} c_i x_i \\ \text{sous contraintes} & \sum_{i=1}^{n} a_{ji} x_i \leq b_j, \quad j = 1, ..., m \\ & \sum_{i=1}^{n} d_{ji} x_i = f_j, \quad j = 1, ..., p \end{array}$$

avec

- n variables d'optimisation : $x_1, ..., x_n$ (scalaires réels)
- donnees de problème (paramètres) : coefficients c_i, a_{ji}, b_j, d_{ji}, f_j
- $\sum c_i x_i$ est la fonction de coût ou fonction objective
- $\sum a_{ji} x_i \leq b_j,\ j = 1, ..., m$, contraintes d'inégalité *(non-strictes)*
- $\sum d_{ji} x_i = f_j,\ j = 1, ..., p$, contraintes d'égalité

Étapes de formulation de modèles d'OL

1. Comprendre le problème (souvent difficile dans la pratique)
2. Identifier les variables de décision
3. Formuler les contraintes comme des combinaisons linéaires du variables de décision
4. Poser la fonction objective comme une combinaison linéaire de les variables de décision

Exemple : problème de transport

- Une compagnie possède n usines de production et m clients.
  - — Chaque usine a la capacité mensuelle u_i, $i = 1, ..., n$ de production, et chaque client a la demande mensuelle d_j, $j = 1, ..., m$.
  - — Soit x_{ij} la quantité de produits fournis par l'usine "i" vers le client "j."
  - — L'objectif est de determiner un plan de transport qui minimise le coût total $\sum_{i,j} c_{ij}x_{ij}$, ou c_{ij} et le coût de transport de "i" vers "j."

Formulation OL

minimiser	$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}x_{ij}$		[cout de transport à minimiser]
sous contraintes	$\sum_{i=1}^{n} x_{ij} \geq d_j,$	$j = 1, ..., m$	[satisfaire toutes les demandes]
	$\sum_{j=1}^{m} x_{ij} \leq u_i,$	$i = 1, ..., n$	[respecter les capacités de production]
	$x_{ij} \geq 0,$	$i = 1, ..., n,$	[contraintes de production
		$j = 1, ..., m$	non-négative]

Exemple : problème de régime optimal

- Il y a n types de produits et m types d'éléments nutritionnels.
  - — Une unité de produit $\#j$ contient p_{ij} grammes d'élément $\#i$ et coute c_j.
  - — La consommation journalière de l'élément $\#i$ doit être entre les limites $[\underline{b}_i, \overline{b}_i]$.
  - — On cherche un "régime" (mélange de produits) le moins cher qui procure les quantités quotidiennes nécessaires d'éléments nutritionnels.

En notant x_j la quantité du $j$ème produit dans le régime, le modèle OL dévient

$$\min_x \sum_{j=1}^n c_j x_j \qquad \text{[cout à minimiser]}$$

sous contraintes

$$\left.\begin{array}{l} \sum_{j=1}^n p_{ij} x_j \geq \underline{b}_i \\ \sum_{j=1}^n p_{ij} x_j \leq \overline{b}_i \\ 1 \leq i \leq m \end{array}\right\} \quad \left[\begin{array}{l} \text{bornes inf et sup du contenu} \\ \text{d'éléments nutritionnels} \\ \text{dans le régime} \end{array}\right]$$

$$x_j \geq 0,\ 1 \leq j \leq n \quad \left[\begin{array}{l} \text{quantité de chaque produit} \\ \text{ne peut pas etre negatif} \end{array}\right]$$

• Le régime optimal est systématiquement utilisé dans les élevages de volaille et de bétail. La nourriture des humains ne peut pas être optimisée de la même façon à cause des facteurs du goût, de diversité, etc, difficiles à prendre en compte.

• Quoi que... Voici le regime optimal pour un humain calculé par le "solveur"

`http://www.neos-guide.org/content/diet-problem-solver`

(le problème est formulé en utilisant 68 produits disponibles dans le logiciel) :

Food	Serving	Cost
Raw Carrots	0.12 cups shredded	0.02
Peanut Butter	7.20 Tbsp	0.25
Popcorn, Air-Popped	4.82 Oz	0.19
Potatoes, Baked	1.77 cups	0.21
Skim Milk	2.17 C	0.28

Coût journalier : $ 0.96

Exemple : problème du flux maximal (Max Flow)

• Étant donné un réseau avec 2 noeuds choisis – une *source* et un *puits*, trouver un flux maximal de la source vers le puits.

Autrement dit, on veut trouver le plus grand s tel que une alimentation externe "s au noeud source, $-s$ au noeud puits, et 0 partout ailleurs" correspond à un flux réalisable respectant les capacités des arcs.

Formulation OL :

$$\max_{f,s} \quad s \left[\begin{array}{l} \text{flux total de la source vers} \\ \text{le puits à maximiser} \end{array} \right]$$

sous contraintes

$$\sum_\gamma P_{i\gamma} f_\gamma = \begin{cases} s, & \text{si } i \text{ est le noeud-source} \\ -s, & \text{si } i \text{ est le noeud-puits} \\ 0, & \text{pour toues les autres noeuds} \end{cases}$$

[loi de concervation des flux]

$f_\gamma \geq 0,\ \gamma \in \Gamma$ [flux dans les arcs sont ≥ 0]

$f_\gamma \leq h_\gamma, \gamma \in \Gamma$ [on doit respecter les capacités des arcs]

où Γ est l'ensemble des arcs du reseau, et $P_{i\gamma} = 1$ si l'arc γ est connecté au noeud i et $P_{i\gamma} = 0$ sinon.

Exemple : problème d'affectation

- On veut trouver la correspondance entre N personnes et N taches de façon que
  - — toute personne est assignée à une seule tache ; toute tache est associée à une seule personne
  - — le cout d'association de la personne i à la tache j est a_{ij}

Formulation combinatoire

$$\begin{array}{rl} \text{minimiser} & \sum_{i,j=1}^{N} a_{ij} x_{ij} \\ \text{sous contraintes} & \sum_{i=1}^{N} x_{ij} = 1, \quad j = 1, ..., N \\ & \sum_{j=1}^{N} x_{ij} = 1, \quad i = 1, ..., N \\ & x_{ij} \in \{0, 1\}, \quad i, j = 1, ..., N \end{array}$$

  - — variable $x_{ij} = 1$ si la personne i est assignée à la tache j ; $x_{ij} = 0$ sinon
  - — il y a $N!$ correspondances possibles – beaucoup trop pour les énumérer toutes

Exemple : problème d'affectation

Formulation OL

$$
\begin{array}{rll}
\text{minimiser} & \sum_{i,j=1}^{N} a_{ij} x_{ij} & \\
\text{sous contraintes} & \sum_{i=1}^{N} x_{ij} = 1, & j = 1, ..., N \\
& \sum_{j=1}^{N} x_{ij} = 1, & i = 1, ..., N \\
& 0 \leq x_{ij} \leq 1, & i, j = 1, ..., N
\end{array}
$$

— nous avons *relaxé* les contraintes binaires $x_{ij} \in \{0, 1\}$

— on peut *démontrer* que dans ce cas tout point optimal satisfait $x_{ij} \in \{0, 1\}$

— ainsi, on peut résoudre ce problème combinatoire (très particulier) de façon efficace (par OL ou des méthodes spécialisées)

Un peu d'histoire

— années 30-40 : Kantorovitch, Koopmans, von Neumann, Dantzig : fondations motivées par des problèmes de l'économie et de logistique

— 1947 : (Dantzig) algorithme de simplexe

— 1948-1949 : application historique : organisation du pont aérien pour ravitaillement de la zone occidentale de Berlin pendant le blocus de Berlin-ouest par les soviétiques. Resolution numérique "à la main" (par algorithme de simplexe avec des milliers de variables)

— 1950-1960 : des nombreuses applications dans les autres disciplines

— 1975 : prix Nobel d'économie pour Kantorovitch et Koopmans

— 1984 (Karmarkar) : premier "algorithme de point intérieur" – avènement de "l'optimisation moderne"

— depuis 1984 : algorithmes pour des problèmes de très grande taille, utilisation de l'OL dans tous les domaines industriels...

Notations :

— **n-vecteurs :** $x = [x_1; ...; x_n] = [x_1, ..., x_n]^T = \begin{bmatrix} x_1 \\ ... \\ x_n \end{bmatrix}$.

On note x_i la i-ème composante de x

On note $x = 0$ si $x_i = 0$, $\forall i$; $x = 1$ si $x_i = 1$, $\forall i$; $x = e_i$ si $x_i = 1$ et $x_j = 0$ pour $j \neq i$ (i-ème vecteur de base canonique).

— **Matrices $m \times n$:**

$$A = \begin{bmatrix} a_{11} & ... & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & ... & a_{mn} \end{bmatrix}$$

avec elements a_{ij} (ou $[A]_{ij}$), $[A]_j$ la j-ème colonne de A (m-vecteurs = matrices $m \times 1$).

On note $A = 0$ (matrice nulle) si $a_{ij} = 0 \, \forall i, j$; $A = I$ (matrice identite) si $a_{ii} = 1$ et $a_{ij} = 0 \, \forall j \neq i$.

Operations matricielles :

- matrice transposée A^T
- multiplication par un scalaire αA
- addition $A + B$ and soustraction $A - B$ de matrices de même taille
- produit matrice-vecteur $y = Ax$ et $y^T = x^T A^T$ (de tailles compatibles)
- produit $C = AB$ de matrices de tailles compatibles (lesquelles ?)
- produit scalaire de 2 vecteurs de même taille :

$$\langle x, y \rangle = x^T y = x_1 y_1 + ... + x_n y_n.$$

- produit scalaire de 2 matrices de même taille $(m \times n)$:

$$\begin{aligned} \langle X, Y \rangle &= \text{Trace}(X^T Y) = \sum_{i=1}^{n} [X^T Y]_{ii} \\ &= \text{Trace}(Y^T X) = ... = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} y_{ij}. \end{aligned}$$

Notations matricielles

— *In extenso*

$$\min \left\{ \sum_{i=1}^{n} c_i x_i : \quad \begin{array}{ll} \sum_{i=1}^{n} a_{ji} x_i \leq b_j, & j = 1, ..., m \\ \sum_{i=1}^{n} d_{ji} x_i = f_j, & j = 1, ..., p \end{array} \right\}$$

— *Notation vectorielle (par contrainte)*

$$\min \left\{ c^T x : \quad \begin{array}{ll} a_j^T x \leq b_j, & j = 1, ..., m \\ d_j^T x = f_j, & j = 1, ..., p \end{array} \right\}$$

avec les n-vecteurs c, a_i et d_i :

$$c = [c_1; ...; c_n], \quad a_j = [a_{j1}; ...; a_{jn}], \quad d_j = [d_{j1}; ...; d_{jn}].$$

— *Notation matricielle :* $\min \left\{ c^T x : \quad Ax \leq b, \; Dx = f \right\}$

où A est une matrice $m \times n$ avec des lignes a_j^T (elements a_{ji}), et D une matrice $p \times n$ avec des lignes d_j^T (elements d_{ji}).

— L'inégalité $a \leq b$ entre les vecteurs est comprise "par composante," dans le sens $a_i \leq b_i$ pour tout i (de même pour "$<$, $>$, $\geq$").

$$\min \left\{ c^T x : \ Ax \leq b, \ Dx = f \right\}$$

Terminologie :

- $x_1, ..., x_n$ *variables de décision* du problème,
  $x = [x_1; ...; x_n]$ est vecteur de decision
- x *solution réalisable (admissible)* ssi elle satisfait les contraintes $Ax \leq b$ et $Dx = f$
- *domaine du problème (ensemble admissible)* $=$ ensemble des solutions réalisables
- x_* est une *solution optimale* ssi elle est réalisable et $c^T x_* \leq c^T x$ pour toute solution réalisable x
- la *valeur optimale* du PL est la valeur $\mathrm{Opt} = c^T x_*$
- on dit que PL est *non-borné intérieurement* si $\mathrm{Opt} = -\infty$
- on dit que PL est *irréalisable (inadmissible)* si le domaine réalisable est vide ; dans ce cas on pose $\mathrm{Opt} = +\infty$.

Pour un problème de maximisation, $\max \left\{ c^T x : \ Ax \leq b, \ Dx = f \right\}$, la situation est inversée : $\mathrm{Opt} = +\infty$ si le problème n'est pas borné, et $\mathrm{Opt} = -\infty$ s'il n'est pas réalisable.

Forme canonique et forme standard

On remarque que

- toute égalité/inégalité linéaire peut être réécrite avec le second membre constant (toutes les variables "à gauche") : $2x_1 \geq 20 - x_2 \Leftrightarrow 2x_1 + x_2 \geq 20$
- toute inégalité non-stricte peut être réécrite comme une inégalité avec "$\leq$" :

$$2x_1 + x_2 \geq 20 \Leftrightarrow -2x_1 - x_2 \leq -20$$

- toute égalité peut être représentée par une paire d'inégalités avec les signes opposés :

$$2x_1 - x_2 = 5 \Leftrightarrow \begin{cases} 2x_1 - x_2 & \leq & 5 \\ -2x_1 + x_2 & \leq & -5 \end{cases}$$

- toute inégalité avec $\leq$, en rajoutant une *variable d'écart* y, peut être réécrite comme une égalité et une inégalité "simple" $y \geq 0$:

$$2x_1 + x_2 \leq 20 \Leftrightarrow \begin{cases} 2x_1 + x_2 + y & = & 20 \\ y & \geq & 0 \end{cases}$$

- minimiser la fonction linéaire $c^T x$ revient exactement à maximiser la fonction linéaire $-c^T x$.

• *Tout programme OL est equivalent à un programme OL en* **forme canonique**, dans laquelle *l'objectif doit être maximisé* et *les contraintes sont les inégalités avec "$\leq$:"*

$$\text{Opt} = \max_x \left\{ \sum_{j=1}^n c_j x_j : \ \sum_{j=1}^n a_{ij} x_j \leq b_i, \ 1 \leq i \leq m \right\}$$

[notation "in extenso"]

$$\Leftrightarrow \quad \text{Opt} = \max_x \left\{ c^T x : \ a_i^T x \leq b_i, \ 1 \leq i \leq m \right\}$$

[notation "par contrainte"]

$$\Leftrightarrow \quad \text{Opt} = \max_x \left\{ c^T x : \ Ax \leq b \right\}$$

[notation "matricielle"]

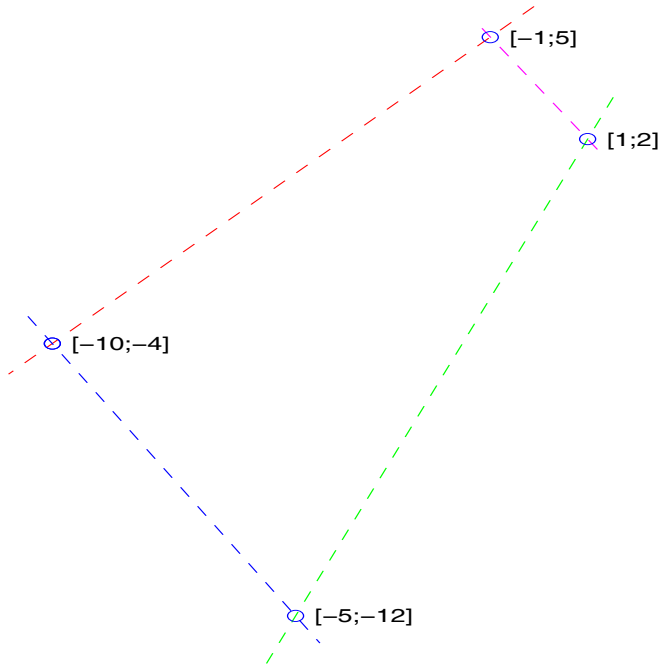
où

$$c = [c_1; ...; c_n], \ b = [b_1; ...; b_m], \ a_i = [a_{i1}; ...; a_{in}], \ A = [a_1^T; a_2^T; ...; a_m^T]$$

• Ensemble $X \subset \mathbb{R}^n$ défini par $X = \{x : Ax \leq b\}$ – l'ensemble de solutions d'un système fini d'inegalites lineaires non-strictes $a_i^T x \leq b_i, \ 1 \leq i \leq m$ en $x \in \mathbb{R}^n$ – s'appel *ensemble polyédrique*, ou un *polyèdre*.

• *Un programme OL en forme canonique consiste à maximiser un objectif linéaire sur un ensemble polyédrique.*

$$\max_x \left\{ x_2 : \left\{ \begin{array}{rcl} -x_1 + x_2 & \leq & 6 \\ 3x_1 + 2x_2 & \leq & 7 \\ 7x_1 - 3x_2 & \leq & 1 \\ -8x_1 - 5x_2 & \leq & 100 \end{array} \right\} \right\}$$



Programme OL et son domaine réalisable

• *Un programme OL en* **forme standard** consiste à *maximiser une forme linéaire* sur l'intersection de *l'orthant non-négatif* $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x \geq 0\}$ et d'un *plan réalisable* $\{x : Ax = b\}$:

$$\text{Opt} = \max_x \left\{ \sum_{j=1}^n c_j x_j : \begin{array}{l} \sum_{j=1}^n a_{ij} x_j = b_i,\ 1 \leq i \leq m \\ x_j \geq 0,\ j = 1, ..., n \end{array} \right\}$$

[notation " in extenso"]

$$\Leftrightarrow \quad \text{Opt} = \max_x \left\{ c^T x : \begin{array}{l} a_i^T x = b_i,\ 1 \leq i \leq m \\ x_j \geq 0,\ 1 \leq j \leq n \end{array} \right\}$$

[notation "par contrainte"]

$$\Leftrightarrow \quad \text{Opt} = \max_x \left\{ c^T x : \ Ax = b,\ x \geq 0 \right\}$$

[notation "matricielle"]

où

$$c = [c_1; ...; c_n],\ b = [b_1; ...; b_m],\ a_i = [a_{i1}; ...; a_{in}],\ A = [a_1^T; a_2^T; ...; a_m^T]$$

• Dans le programme OL standard
  — toute variables sont non-négatives
  — toutes contraintes linéaires "generales" sont des égalités

Remarque : *la forme standard de PL est universelle : tout programme linéaire est equivalent à un PL en forme standard.*

En effet, il suffit de convertir un PL en forme canonique $\max_x\{c^T x : \ Ax \leq b\}$. Pour le faire,

— on introduit des *variables d'écart* (une par inégalité), et on réécrit le problem comme

$$\max_{x,s}\{c^T x : Ax+s = b,\ s \geq 0\}$$

— ensuite, on représente x comme $x = u - v$, la difference de 2 vecteurs non-négatifs, et on arrive à

$$\max_{u,v,s}\left\{c^T u - c^T v : \ Au - Av + s = b,\ [u; v; s] \geq 0\right\}.$$

Illustration :

$$\text{Opt} = \max_{x} [2x_1 + 3x_2 - x_3]$$

s.c.

$$\begin{array}{rcl} 3x_1 + 4x_2 + 5x_3 & \leq & 6 \\ 7x_1 + 8x_2 + 9x_3 & \leq & 10 \end{array}$$

$\Leftrightarrow$

$$\text{Opt} = \max_{x,s} [2x_1 + 3x_2 - x_3]$$

s.c.

$$\begin{array}{rcl} 3x_1 + 4x_2 + 5x_6 + s_1 & = & 6 \\ 7x_1 + 8x_2 + 9x_3 + s_2 & = & 10 \\ s_1 \geq 0, s_2 \geq 0 & & \end{array}$$

$\Leftrightarrow$

$$\text{Opt} = \max_{u,v,s} [2[u_1 - v_1] + 3[u_2 - v_2] - [u_3 - v_3]]$$

s.c.

$$\begin{array}{rcl} 3[u_1 - v_1] + 4[u_2 - v_2] + 5[u_3 - v_3] + s_1 & = & 6 \\ 7[u_1 - v_1] + 8[u_2 - v_2] + 9[u_3 - v_3] + s_2 & = & 10 \\ s_1 \geq 0, s_2 \geq 0, u_1 \geq 0, u_2 \geq 0, u_3 \geq 0, & & \\ v_1 \geq 0, v_2 \geq 0, v_3 \geq 0, & & \end{array}$$

Interprétation géométrique

Normes dans $\mathbb{R}^n$

- Norme euclidienne (ℓ_2) : $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{x^T x}$
- Normes ℓ_1 et ℓ_∞ :

$$\|x\|_1 = |x_1| + ... + |x_n|, \quad \|x\|_\infty = \max\{|x_1|, ...|x_n|\}$$

Propriétés :

- $\|x\| \geq 0$, $\|x\| = 0$ ssi $x = 0$ (positivité)
- $\|x + y\| \leq \|x\| + \|y\|$ (inégalité triangulaire)
- $\forall \alpha \in \mathbb{R}$, $\|\alpha x\| = |\alpha| \|x\|$ (homogénéité)

- *Inégalité de Cauchy-Schwartz :* $|x^T y| \leq \|x\|_2 \|y\|_2$
  - $|x^T y| = \|x\|_2 \|y\|_2$ ssi x et y sont proportionnels
  - implique d'autres inégalités intéressantes, e.g.,

$$\left| \sum_{i=1}^n x_i \right| \leq \sum_{i=1}^n |x_i| \leq \sqrt{n} \|x\|_2$$

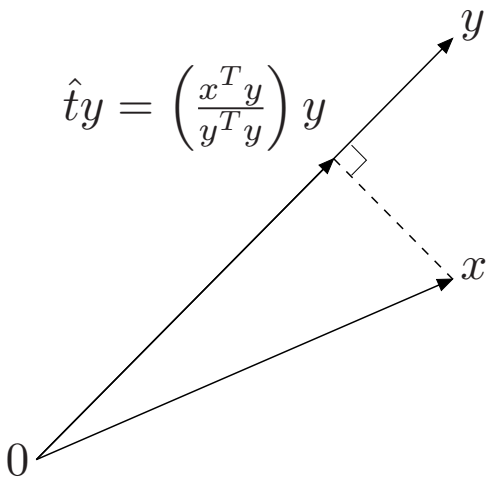
(pour quelle x cette inégalité devient une égalité ?).

Hyperplans et projections

• Projection de x sur la droite avec le vecteur-directeur y : vecteur $\hat{t}y$ ou

$$\hat{t} = \underset{t}{\operatorname{argmin}} \|x - ty\|_2$$

$$\Rightarrow \quad \hat{t} = \frac{x^T y}{\|y\|_2^2} = \frac{\|x\|_2 \cos\theta}{\|y\|_2}$$



• **Un hyperplan** est l'ensemble de solutions de l'équation linéaire $a^T x = b$ avec vecteur normal $a \neq 0$

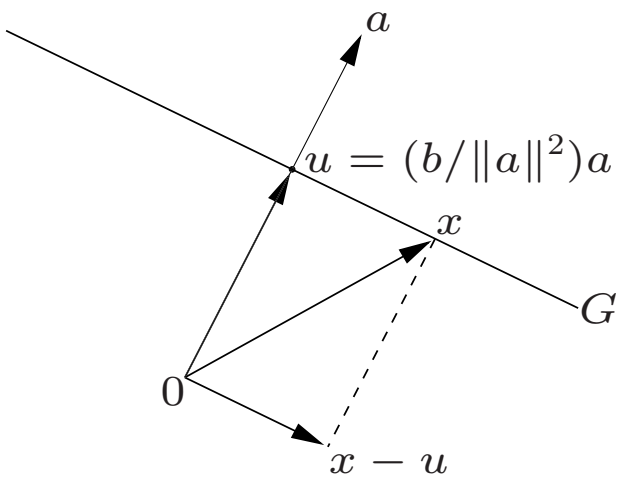
• **Un demi-espace** est l'ensemble de solutions de *l'inégalité linéaire*

$$a^T x \leq b$$

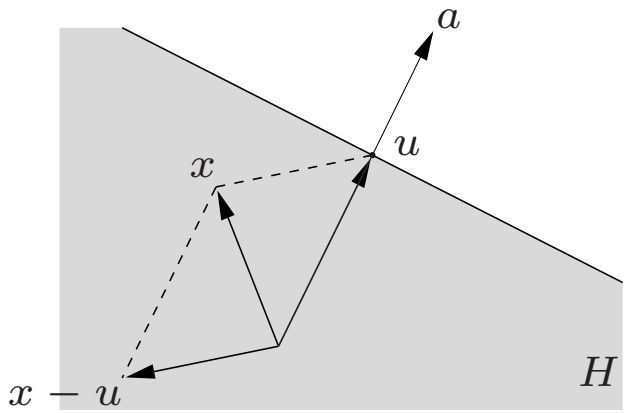
avec vecteur normal $a \neq 0$

Interpretation géométrique

$G = \{x : \ a^T x = b\}$



$H = \{x : \ a^T x \leq b\}$

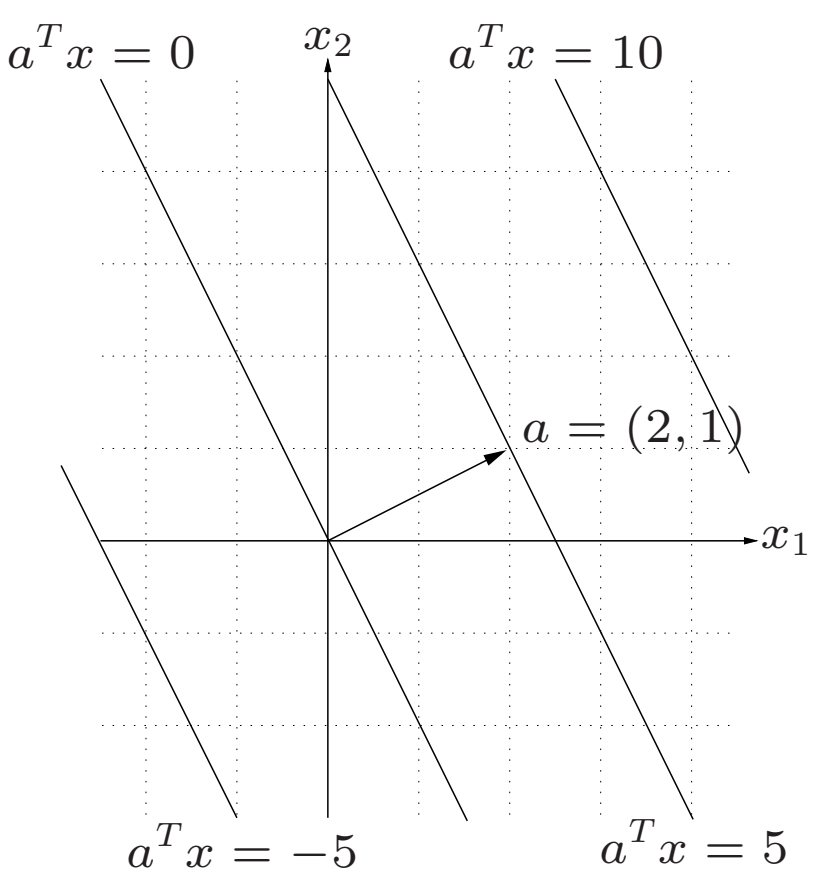


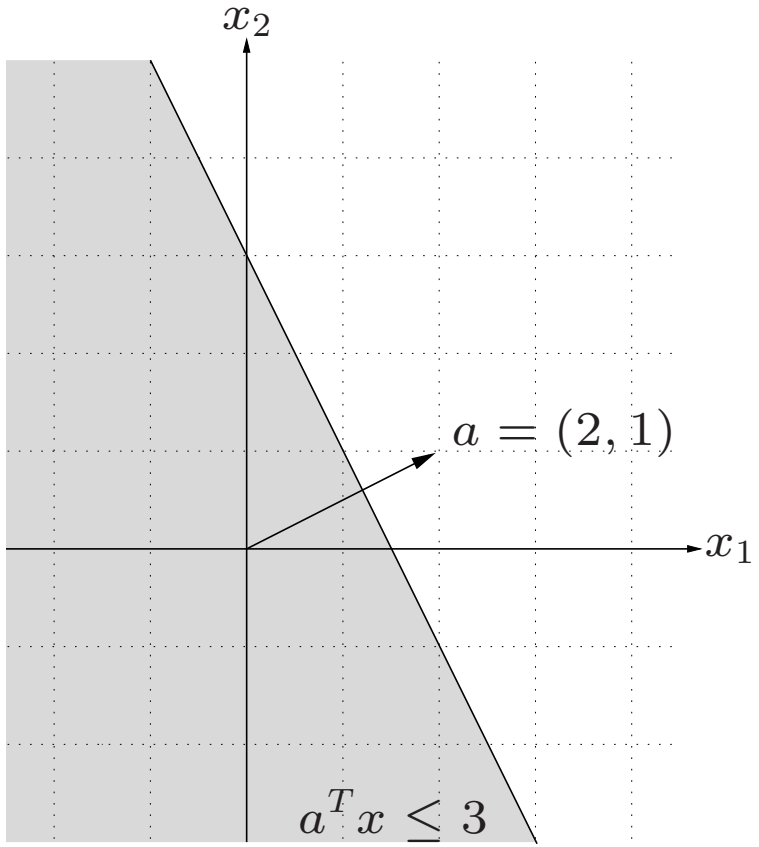
— vecteur $u = \frac{b}{\|a\|_2^2} a$ satisfait $a^T u = b$

— $x \in G$ si $a^T(x-u) = 0$ (i.e. $x - u \perp a$)

— $x \in H$ si $a^T(x-u) \leq 0$ (i.e. angle $\angle(x-u, a) \geq \pi/2$)
($\Leftrightarrow a^T x \leq a^T u = b$).

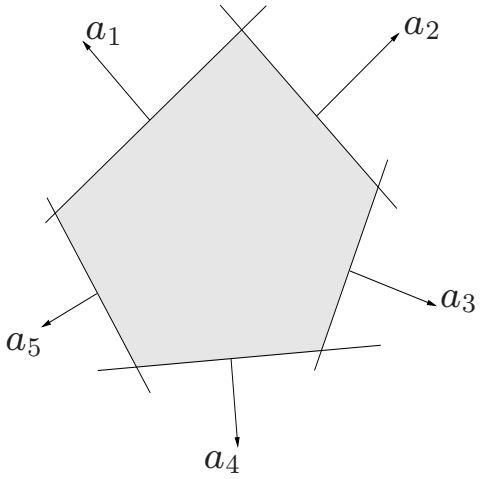
Exemple





Un polyèdre : ensemble de solutions d'un système fini d'inégalités linéaires :

$$a_1^T x \leq b_1, \;\; a_2^T x \leq b_2, \; ..., \; a_m^T x \leq b_m$$



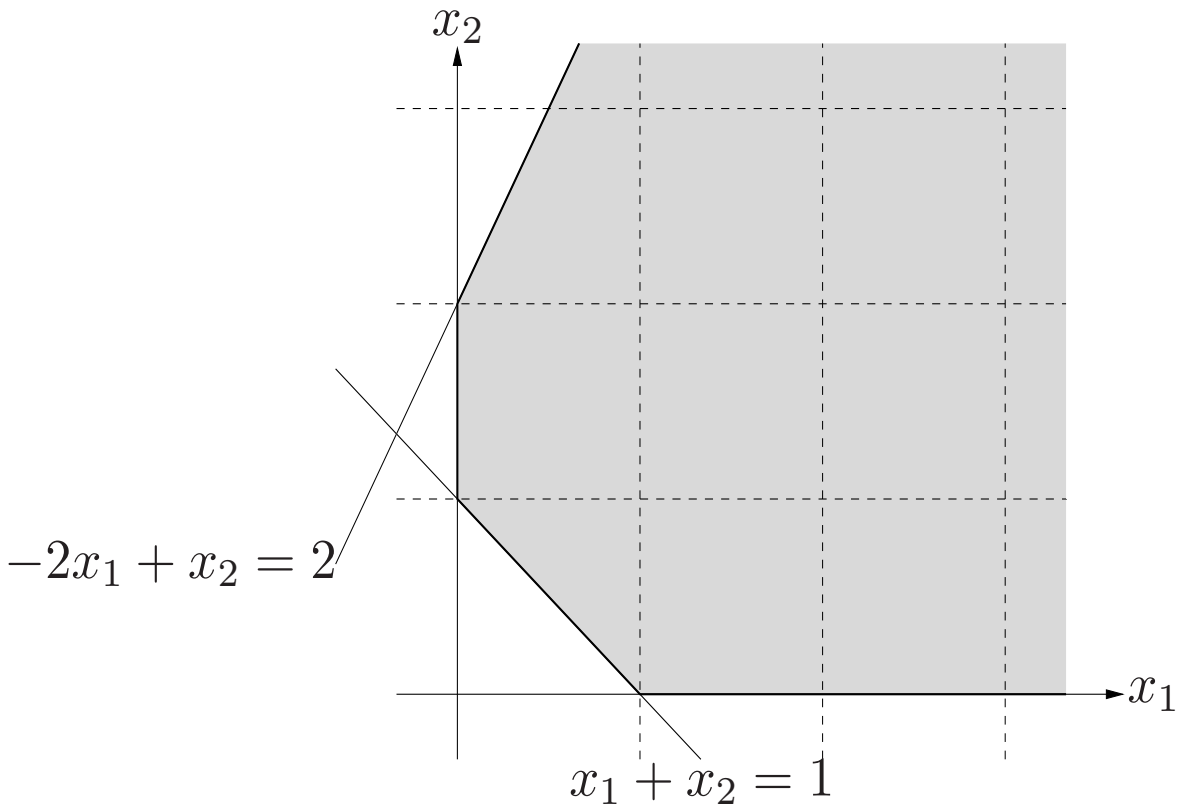
— intersection d'un nombre fini de demi-espaces

— notation matricielle : $Ax \leq b$, ou A la matrice avec les lignes a_i^T

— peut contenir les égalités $a_i^T x = b_i$
(avec les inégalités $a_i^T x \leq b_i$ et $-a_i^T x \leq -b_i$)

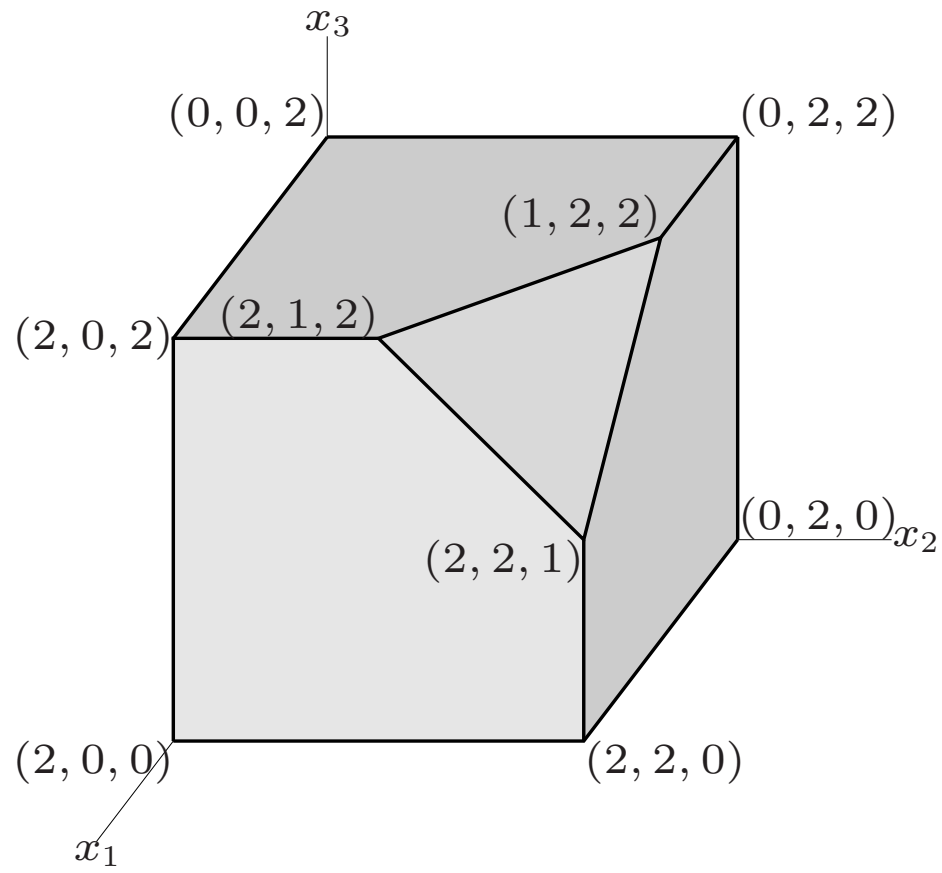
Exemple

$$x_1 + x_2 \geq 1, \qquad -2x_1 + x_2 \leq 2, \qquad x_1 \geq 0, \qquad x_2 \geq 0$$



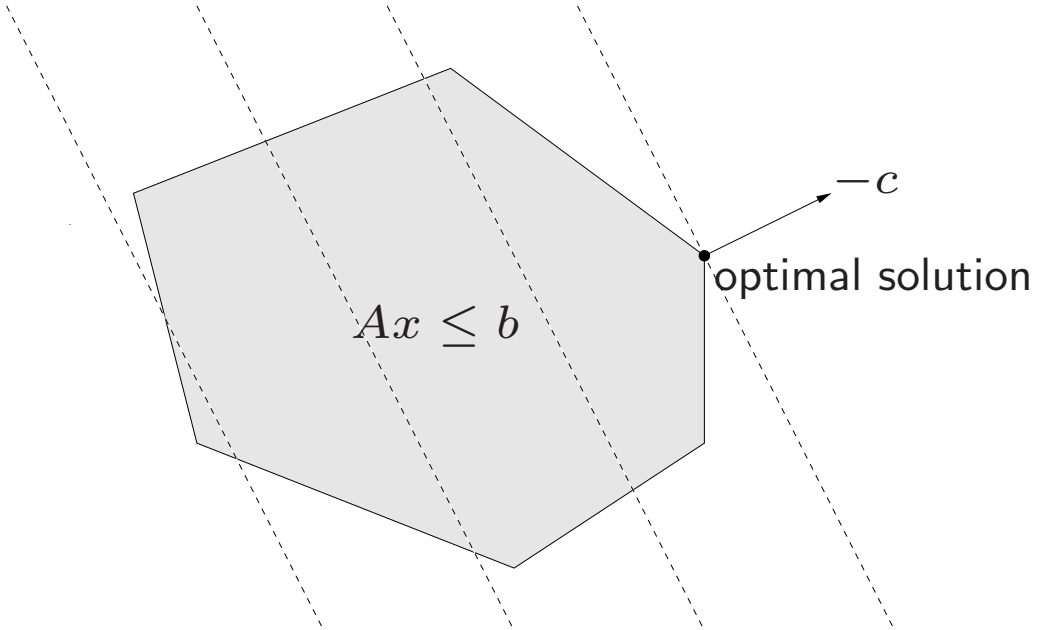
Exemple

$$0 \leq x_1 \leq 2, \qquad 0 \leq x_2 \leq 2, \qquad 0 \leq x_3 \leq 2, \qquad x_1 + x_2 + x_3 \leq 5$$



Interprétation géométrique de PL

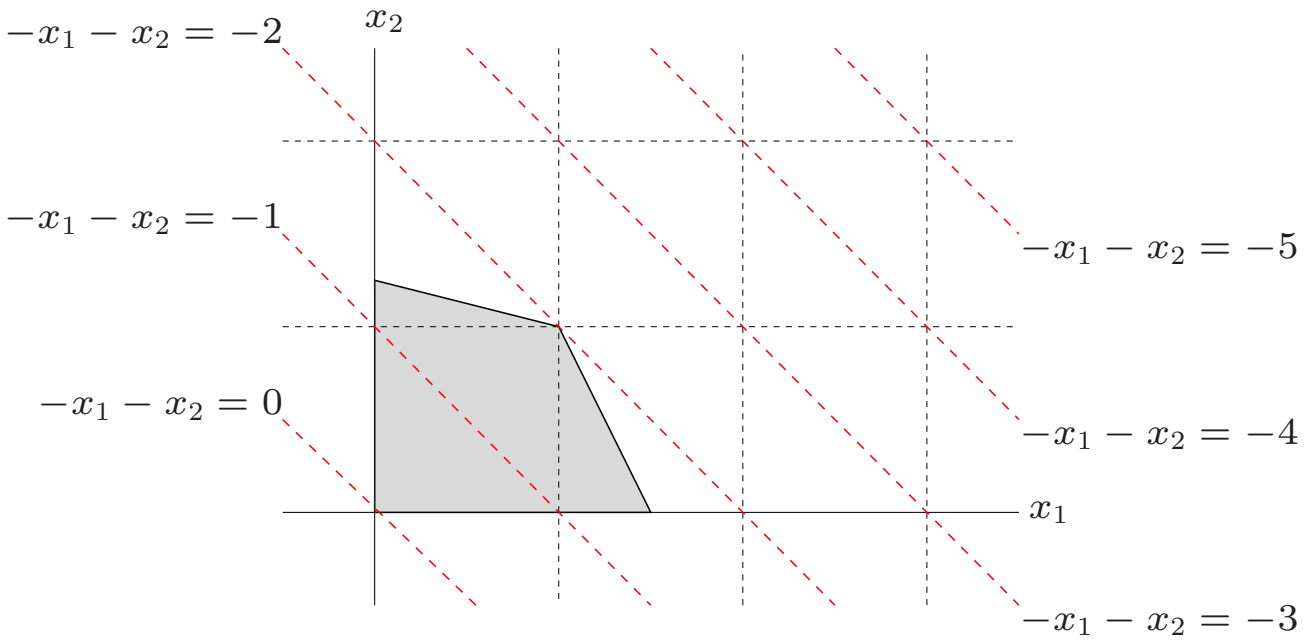
$$\min_x \left\{ c^T x : \ Ax \leq b \right\} = - \max_x \left\{ -c^T x : \ Ax \leq b \right\}$$



lignes (hyperplans) en pointillés sont des ensembles de niveau $\{x : \ c^T x = \alpha\}$ de la forme linéaire $c^T x$ pour les différents α

Exemple

$$\min \left\{ -x_1 - x_2 : \begin{array}{l} 2x_1 + x_2 \leq 3 \\ x_1 + 4x_2 \leq 5 \\ x_1 \geq 0,\ x_2 \geq 0 \end{array} \right\}$$



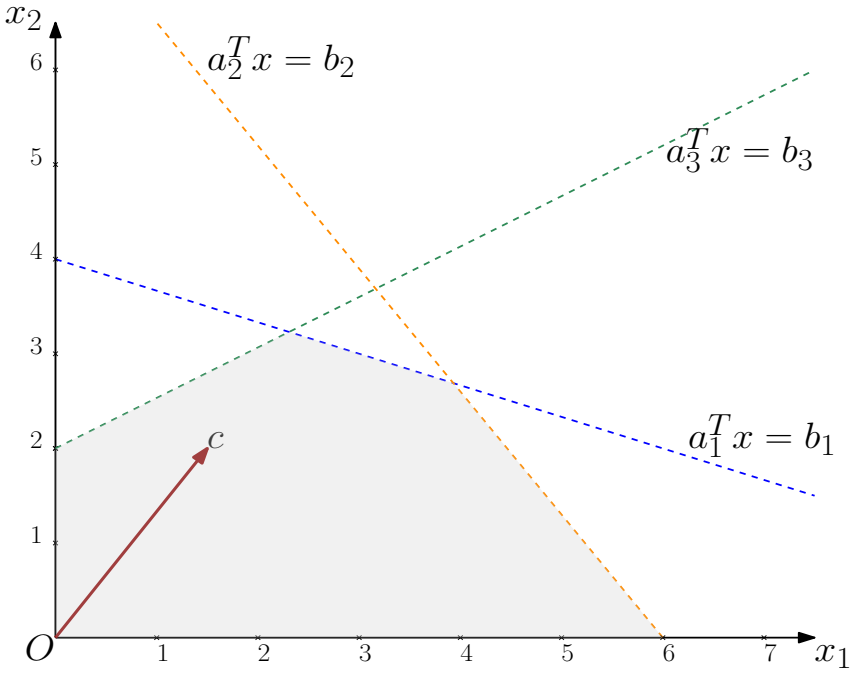
solution optimale : $x = [1; 1]$, $c^T x = -2$

Algorithme du simplexe

Soit un programme linéaire sous la forme spéciale :

$$\max_{x \in \mathbb{R}^2} \left\{ c^T x : \begin{array}{l} a_1^T x \leq b_1 \\ a_2^T x \leq b_2 \\ a_3^T x \leq b_3 \\ x \geq 0 \end{array} \right\}$$

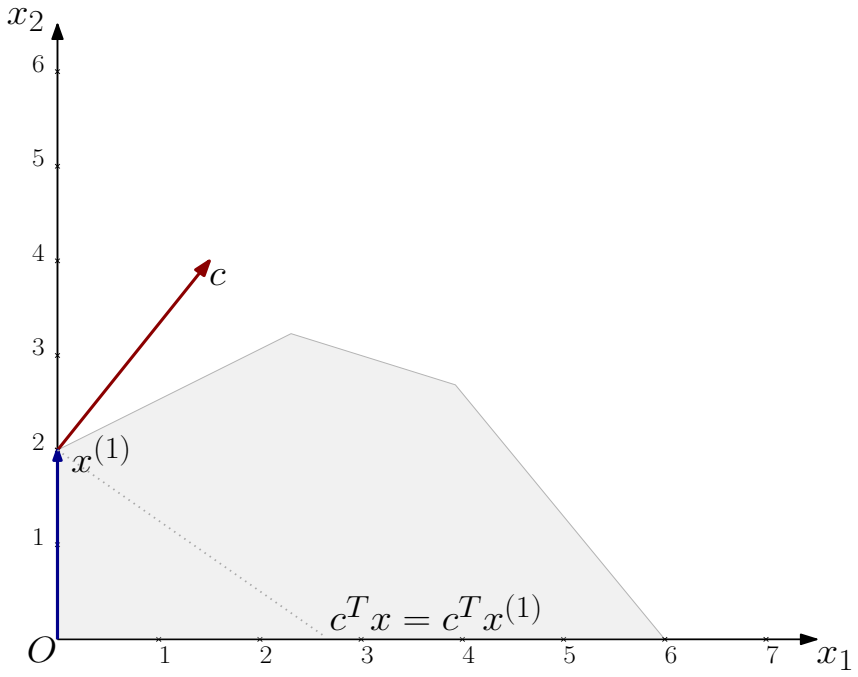
avec $b_1 \geq 0,\ b_2 \geq 0,\ b_3 \geq 0$.



$$\max_{x\in\mathbb{R}^2}\left\{ c^T x : \begin{array}{l} a_1^T x \le b_1 \\ a_2^T x \le b_2 \\ a_3^T x \le b_3 \\ x \ge 0 \end{array} \right\}$$

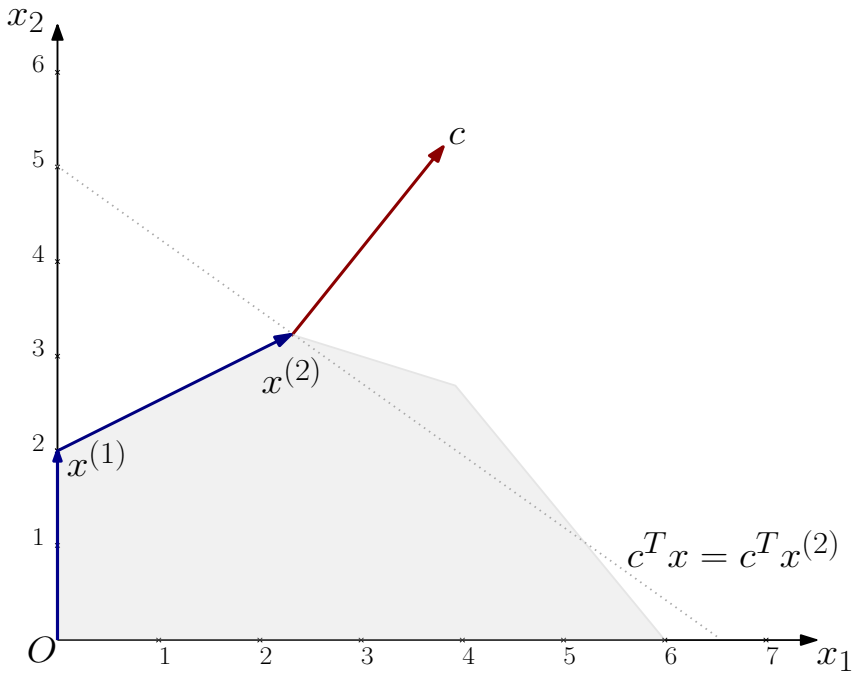
On cherche une solution en parcourant les points extremes à partir de $x^{(0)} = 0$.

$$x^{(1)} : x_1 = 0, \quad a_3^T x = b_3.$$



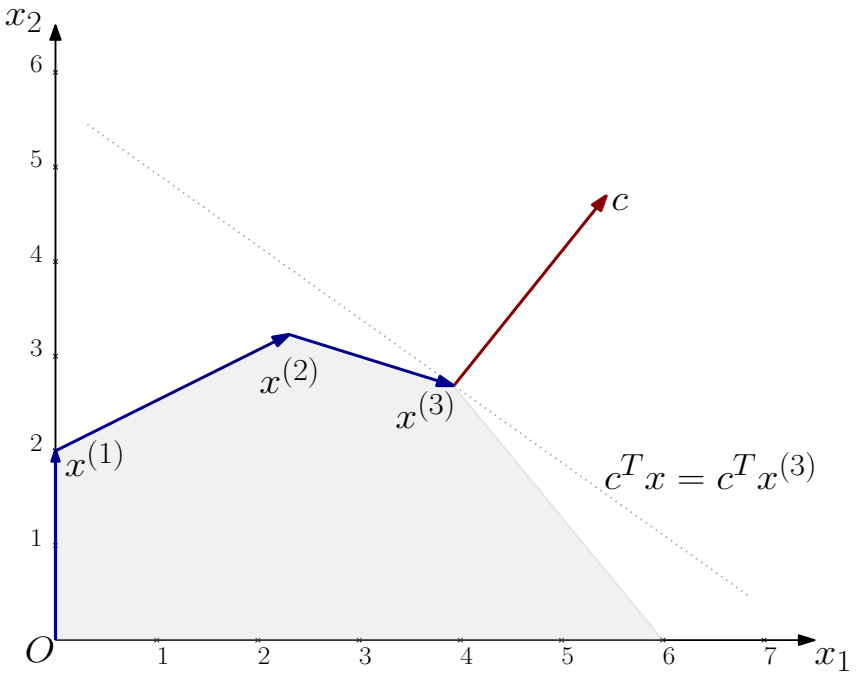
$$\max_{x\in\mathbb{R}^2}\left\{ c^T x : \begin{array}{l} a_1^T x \le b_1 \\ a_2^T x \le b_2 \\ a_3^T x \le b_3 \\ x \ge 0 \end{array} \right\}$$

$$x^{(2)} : a_1^T x = b_1, \;\; a_3^T x = b_3.$$



$$\max_{x\in\mathbb{R}^2}\left\{ c^T x : \begin{array}{l} a_1^T x \le b_1 \\ a_2^T x \le b_2 \\ a_3^T x \le b_3 \\ x \ge 0 \end{array} \right\}$$

$x^{(3)}$: $a_1^T x = b_1, \;\; a_2^T x = b_2.$



Solution optimale : $x^{(3)}$

Algorithme du simplexe – explication

Exemple : résoudre le problème

$$\max_{x_1,x_2} \left\{ x_1 + x_2 : \begin{array}{l} 2x_1 + x_2 \leq 4, \\ x_1 + 2x_2 \leq 3 \\ x_1 \geq 0,\ x_2 \geq 0 \end{array} \right\}$$

1^o. On convertit en forme standard :

$$\max_{x_1,x_2} \left\{ x_1 + x_2 : \begin{array}{llll} 2x_1 & +x_2 & +s_1 & = 4, \\ x_1 & +2x_2 & +s_2 & = 3 \\ x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, \quad s_2 \geq 0 & \end{array} \right\}$$

2^o. On pose $z = x_1 + x_2$, et on considère le système

$$\left\{ \begin{array}{llll} z - x_1 & -x_2 & & = 0 \\ 2x_1 & +x_2 & +s_1 & = 4, \\ x_1 & +2x_2 & +s_2 & = 3 \\ x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, \quad s_2 \geq 0 & \end{array} \right.$$

Notre objectif est de maximiser z.

$$\begin{cases} z & -x_1 & -x_2 & & & = 0 \\ & 2x_1 & +x_2 & +s_1 & & = 4 \\ & x_1 & +2x_2 & & +s_2 & = 3 \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

On dit que une variable est *basique* s'elle n'apparaît que dans une seule equation.

On forme une *solution basique* en mettant à zero toute variable non basique.

- Base : s_1, s_2. Nous avons

$$x_1 = x_2 = 0,\ z = x_1 + x_2 = 0,\ s_1 = 4,\ s_2 = 3$$

Peut on accroître z ?

Règle I *Si toutes les variables en 1ère ligne ont des coefficients non négatifs, la solution basique est optimale. Sinon, on choisit une variable non basique dont le coefficient est négatif et on l'augment tant que le système reste admissible.*

$$\begin{cases} z & -x_1 & -x_2 & & & = 0 \\ & 2x_1 & +x_2 & +s_1 & & = 4 \\ & x_1 & +2x_2 & & +s_2 & = 3 \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

On choisit, par exemple, x_1. Quel pivot d'algorithme de Gauss choisir ?

Règle II *Choisir la ligne qui correspond au plus petit rapport* $\frac{\textit{second membre}}{\textit{coefficient de la variable}}$
(on suppose que coefficient de la variable est strictement positif)

Puisque $4/2 < 3/1$, on choisit la 2ème ligne :

$$\begin{cases} z & & -\frac{1}{2}x_2 & +\frac{1}{2}s_1 & & = 2 \\ & x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}s_1 & & = 2 \\ & & \frac{3}{2}x_2 & -\frac{1}{2}s_1 & +s_2 & = 1 \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

- Base : x_1, s_2. Nous avons

$$s_1 = x_2 = 0,\ x_1 = 2,\ s_2 = 3,\ z = 2$$

$$\begin{cases} z & & -\frac{1}{2}x_2 & +\frac{1}{2}s_1 & & = 2 \\ & x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}s_1 & & = 2 \\ & & \frac{3}{2}x_2 & -\frac{1}{2}s_1 & +s_2 & = 1 \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

On continue avec x_2. Comme $2/\frac{1}{2} > 1/\frac{3}{2}$, on choisit la 3ème ligne :

$$\begin{cases} z & & & +\frac{1}{3}s_1 & +\frac{1}{3}s_2 & = \frac{7}{3} \\ & x_1 & & +\frac{2}{3}s_1 & -\frac{1}{3}s_2 & = \frac{5}{3} \\ & & x_2 & -\frac{1}{3}s_1 & +\frac{2}{3}s_2 & = \frac{2}{3} \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

- Base : x_1, x_2. Nous avons

$$s_1 = s_2 = 0,\ x_1 = \tfrac{5}{3},\ x_2 = \tfrac{2}{3},\ z = \tfrac{7}{3}$$

- *D'après la Règle I, nous avons trouvé la solution optimale*

On récapitule :

$$\begin{cases} z & -x_1 & -x_2 & & & = 0 \\ & 2x_1 & +x_2 & +s_1 & & = 4 \\ & x_1 & +2x_2 & & +s_2 & = 3 \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

On commence avec $x^{(0)} = [0; 0]$, contraintes "actives" :

$$x_1 = 0,\ x_2 = 0$$

$$\Downarrow$$

$$\begin{cases} z & & -\frac{1}{2}x_2 & +\frac{1}{2}s_1 & & = 2 \\ & x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}s_1 & & = 2 \\ & & \frac{3}{2}x_2 & -\frac{1}{2}s_1 & +s_2 & = 1 \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

Solution actuelle $x^{(1)} = [2; 0]$, contraintes "actives" :

$$x_2 = 0,\ \ 2x_1 + \underbrace{x_2}_{=0} = 4$$

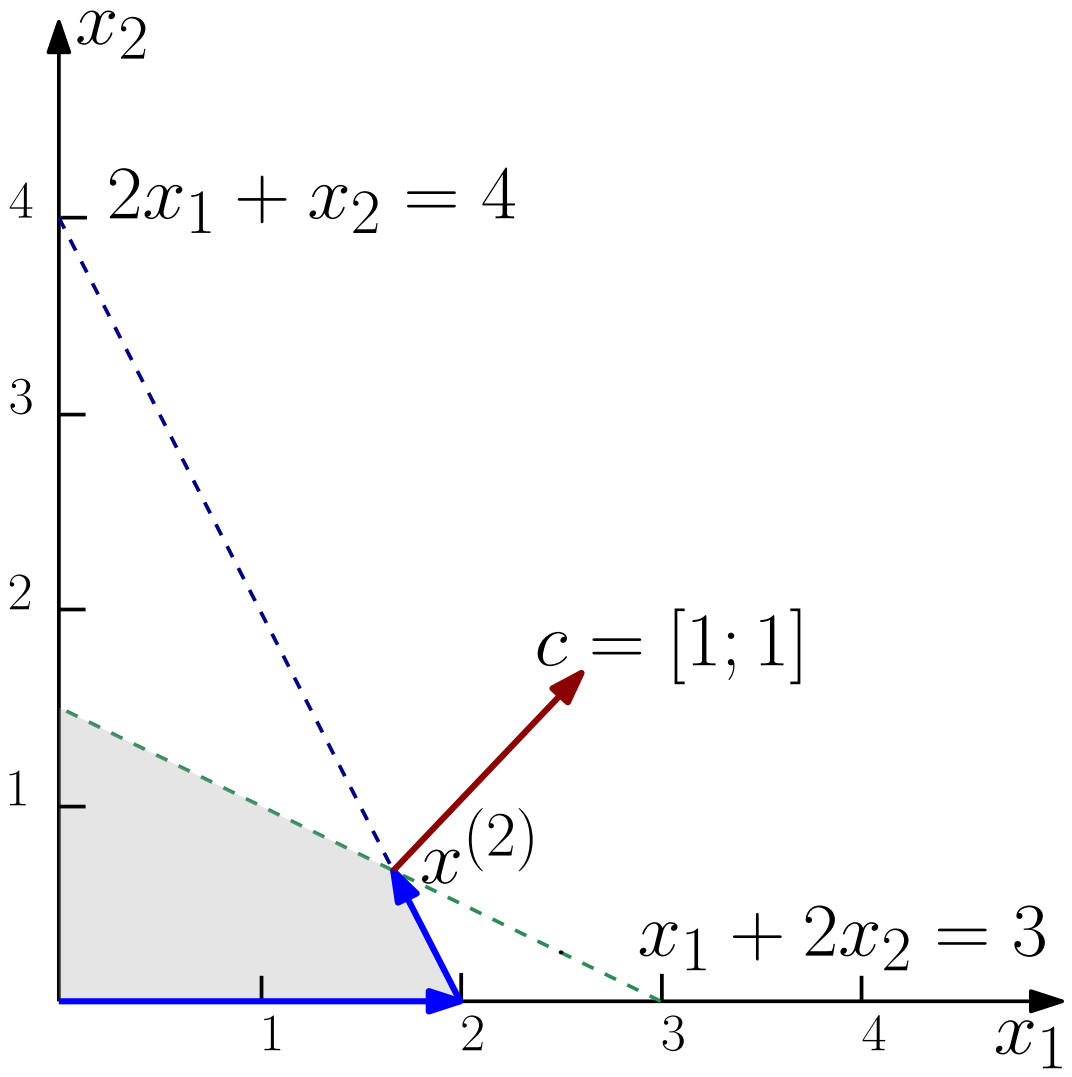
$$\Downarrow$$

$$\begin{cases} z & & & +\frac{1}{3}s_1 & +\frac{1}{3}s_2 & = \frac{7}{3} \\ & x_1 & & +\frac{2}{3}s_1 & -\frac{1}{3}s_2 & = \frac{5}{3} \\ & & x_2 & -\frac{1}{3}s_1 & +\frac{2}{3}s_2 & = \frac{2}{3} \\ & x_1 \geq 0, & x_2 \geq 0, & s_1 \geq 0, & s_2 \geq 0 & \end{cases}$$

Solution actuelle $x^{(2)} = [\frac{5}{3}; \frac{2}{3}]$, contraintes "actives" :

$$2x_1 + x_2 = 4, \quad x_1 + 2x_2 = 3$$

Nous avons trouvé une solution optimale



Interprétation géométrique

- Écriture sous forme de tableau :

z	x_1	x_2	s_1	s_2	RHS	Solution basique
1	-1	-1	0	0	0	basique $s_1 = 4,\ s_2 = 3$
0	2	1	1	0	4	non basique $x_1 = x_2 = 0$
0	1	2	0	1	3	$z = 0$
1	0	$-\frac{1}{2}$	$\frac{1}{2}$	0	2	basique $x_1 = 2,\ s_2 = 1$
0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	2	non basique $x_2 = s_1 = 0$
0	0	$\frac{3}{2}$	$-\frac{1}{2}$	1	1	$z = 2$
1	0	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{7}{3}$	basique $x_1 = \frac{5}{3},\ x_2 = \frac{2}{3}$
0	1	0	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{5}{3}$	non basique $s_1 = s_2 = 0$
0	0	1	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$z = \frac{7}{3}$

Cycles

- Soit

$$\max_{x_1,x_2}\left\{x_1+\tfrac{1}{2}x_2:\quad \begin{array}{l} 2x_1+x_2\le 4,\\ x_1+2x_2\le 3\\ x_1\ge 0,\ x_2\ge 0\end{array}\right\}$$

Le tableau d'algorithme du simplexe :

z	x_1	x_2	s_1	s_2	RHS	Solution basique
1	-1	$-\frac{1}{2}$	0	0	0	basique $s_1=4,\ s_2=3$
0	2	1	1	0	4	non basique $x_1=x_2=0$
0	1	2	0	1	3	$z=0$
1	0	0	$\frac{1}{2}$	0	2	basique $x_1=2,\ s_2=1$
0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	2	non basique $x_2=s_1=0$
0	0	$\frac{3}{2}$	$-\frac{1}{2}$	1	1	$z=2$

Maintenant, Règle I implique que $[x_1;x_2]=[2;0]$ est une solution optimale.

Néanmoins, on peut essayer d'augmenter x_2 pour obtenir une solution basique avec $x_2 \neq 0$. Cela donne :

z	x_1	x_2	s_1	s_2	RHS	Solution basique
1	0	0	$\frac{1}{2}$	0	2	basique $x_1 = \frac{5}{3},\ x_2 = \frac{2}{3}$
0	1	0	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{5}{3}$	non basique $s_1 = s_2 = 0$
0	0	1	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$z = 2$

On observe que le coefficient de s_2 en 1ère ligne est 0. Si on choisi s_2 et le pivot en 3ème ligne on obtient de nouveau

z	x_1	x_2	s_1	s_2	RHS	Solution basique
1	0	0	$\frac{1}{2}$	0	2	basique $x_1 = 2,\ s_2 = 1$
0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	2	non basique $x_2 = s_1 = 0$
0	0	$\frac{3}{2}$	$-\frac{1}{2}$	1	1	$z = 2$

Ce sont exactement le tableau et la solution précédents !

Problème dégénéré

- On considère le programme

$$\max_{x_1,x_2} \left\{ 2x_1 + x_2 : \begin{array}{rcl} 3x_1 + x_2 & \leq & 6, \\ x_1 - x_2 & \leq & 2 \\ x_2 & \leq & 3 \\ x_1 \geq 0,\ x_2 & \geq & 0 \end{array} \right\}$$

L'application de l'algorithme du simplex résulte en

z	x_1	x_2	x_3	s_1	s_2	RHS	Solution basique
1	-2	-1	0	0	0	0	basique $x_3 = 6,\ s_1 = 2,\ s_2 = 3$
0	3	1	1	0	0	6	non basique $x_1 = x_2 = 0$
0	1	-1	0	1	0	2	$z = 0$
0	0	1	0	0	1	3	
1	0	-3	0	2	0	4	basique $x_1 = 2,\ x_3 = 0,\ s_2 = 3$
0	0	4	1	-3	0	0	non basique $x_2 = s_1 = 0$
0	1	-1	0	1	0	2	$z = 4$
0	0	1	0	0	1	3	

On note qu'une des variables basiques – x_3 – est nulle.

Maintenant, Règle I suggère de choisir x_2 comme nouvelle variable basique, et Règle II implique le pivot de la 2ème ligne :

z	x_1	x_2	x_3	s_1	s_2	RHS	Solution basique
1	0	0	$\frac{3}{4}$	$-\frac{1}{4}$	0	4	basique $x_1 = 2,\ x_2 = 0,\ s_2 = 3$
0	0	1	$\frac{1}{4}$	$-\frac{3}{4}$	0	0	non basique $x_3 = s_1 = 0$
0	1	0	$\frac{1}{4}$	$\frac{1}{4}$	0	2	$z = 4$
0	0	0	$-\frac{1}{4}$	$\frac{3}{4}$	1	0	

C'est la même solution, sauf que la variable basique dégénérée est x_2. Si on continue par s_1, avec le pivot de la 4ème ligne,

z	x_1	x_2	x_3	s_1	s_2	RHS	Solution basique
1	0	0	$\frac{2}{3}$	0	$\frac{1}{3}$	5	basique $x_1 = 1,\ x_2 = 3,\ s_1 = 4$
0	0	1	0	0	1	3	non basique $x_3 = s_2 = 0$
0	1	0	$\frac{1}{3}$	0	$-\frac{1}{3}$	1	$z = 5$
0	0	0	$-\frac{1}{3}$	1	$\frac{4}{3}$	4	

La dégénérescence n'a pas empêchée la convergence vers la solution optimale, mais dans certains cas elle peut conduire au cycles.

Problème non borné

- Exemple :

$$\max_{x_1,x_2} \left\{ 2x_1 + x_2 : \begin{array}{rcl} -x_1 + x_2 & \leq & 1, \\ x_1 - 2x_2 & \leq & 2 \\ x_1 \geq 0,\ x_2 & \geq & 0 \end{array} \right\}$$

La resolution par l'algorithme du simplex donne

z	x_1	x_2	s_1	s_2	RHS	Solution basique
1	-2	-1	0	0	0	basique $s_1 = 1,\ s_2 = 2$
0	-1	1	1	0	1	non basique $x_1 = x_2 = 0$
0	1	-2	0	1	2	$z = 0$
1	0	-5	0	2	4	basique $x_1 = 2,\ s_1 = 3$
0	0	-1	1	1	3	non basique $x_2 = s_2 = 0$
0	1	-2	0	1	2	$z = 4$

A ce stade Règle I choisit x_2, mais il n'y a pas de pivot positif dans la colonne correspondante.

$\Rightarrow$ En augmentant x_2, on va jamais attendre une contrainte $\Rightarrow$ Il n'y a pas de limite pour la valeur du problème – problème non borné

Dualité linéaire

Dualité pour les systèmes d'inégalités linéaires

Comment repondre aux questions suivantes :

- Comment savoir si l'ensemble polyédrique

$$X = \{x \in \mathbb{R} : \ Ax \leq b\}$$

est/n'est pas vide ?

- Comment savoir si l'ensemble polyédrique

$$X = \{x \in \mathbb{R} : \ Ax \leq b\}$$

est/n'est pas bornée ?

- Comment comprendre si les deux polyèdres

$$X = \{x \in \mathbb{R} : \ Ax \leq b\}, \quad X' = \{x \in \mathbb{R} : \ A'x \leq b'\}$$

coincident/ne coincident pas ?

- Comment savoir si le programme OL réalisable/irréalisable ?

Notre objectif actuel sera d'étudier les réponses donnés par le *théorème de dualité en programmation linéaire.*

Théorème sur alternative linéaire

- Soit le système de m inégalités linéaires strictes et non-strictes en $x \in \mathbb{R}^n$:

$$\left\{ \begin{array}{ll} a_i^T x < b_i, & i \in I \\ a_i^T x \leq b_i, & i \in \overline{I} \end{array} \right. \qquad (\mathcal{S})$$

où $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $1 \leq i \leq m$, avec $I \subset \{1, ..., m\}$, et $\overline{I} = \{1, ..., m\} \backslash I$.

$(\mathcal{S})$ est un système "universel" d'inégalités linéaires.

Questions importantes (opérationnelles) :

— *Comment trouver une solution de $(\mathcal{S})$ quand elle existe ?*

— *Comment comprendre que $(\mathcal{S})$ est incompatible ?*

Questions importantes (descriptives) :

— *Comment certifier que $(\mathcal{S})$ est soluble ?*

— *Comment certifier que $(\mathcal{S})$ est incompatible ?*

$$\left\{\begin{array}{ll} a_i^T x < b_i, & i \in I \\ a_i^T x \leq b_i, & i \in \overline{I} \end{array}\right. \qquad (\mathcal{S})$$

• Il est facile de certifier que le système est réalisable : il suffit de produire le certificat – une solution candidate qui satisfait le système.

Exemple : Le vecteur $\bar{x} = [10; 10; 10]$ est un certificat d'admissibilité du système

$$\begin{array}{rrrcr} -x_1 & -x_2 & -x_3 & < & -29 \\ x_1 & +x_2 & & \leq & 20 \\ & x_2 & +x_3 & \leq & 20 \\ x_1 & & +x_3 & \leq & 20 \end{array}$$

• Mais comment *certifier* que $(\mathcal{S})$ *n'a pas* de solution ? E.g., comment prouver que le système

$$\begin{array}{rrrcr} -x_1 & -x_2 & -x_3 & < & -30 \\ x_1 & +x_2 & & \leq & 20 \\ & x_2 & +x_3 & \leq & 20 \\ x_1 & & +x_3 & \leq & 20 \end{array}$$

est incompatible ?

$$\left\{ \begin{array}{ll} a_i^T x < b_i, & i \in I \\ a_i^T x \leq b_i, & i \in \overline{I} \end{array} \right. \qquad (\mathcal{S})$$

• **Une idée simple :** *si on fait une somme pondérée d'inégalités de* $(\mathcal{S})$ *avec des coefficients non-négatifs, on obtient une inégalité linéaire qui est une conséquence du système – il est satisfaite sur toute solution de* $(\mathcal{S})$*. Si cette inégalité n'a pas de solutions, alors* $(\mathcal{S})$*, non plus, n'a pas de solutions.*

Exemple : Pour le système

$$\begin{array}{r|rrrcr} 2\times & -x_1 & -x_2 & -x_3 & < & -30 \\ 1\times & x_1 & +x_2 & & \leq & 20 \\ 1\times & & x_2 & +x_3 & \leq & 20 \\ 1\times & x_1 & & +x_3 & \leq & 20 \end{array}$$

il suffit de sommer les inégalités avec des poids en rouge pour obtenir *l'inégalité contradictoire*

$$0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 < 0.$$

$\Rightarrow$ *le vecteur* $\lambda = [2; 1; 1; 1]$ *certifie que ce système n'est pas réalisable.*

$$\left\{ \begin{array}{ll} a_i^T x < b_i, & i \in I \\ a_i^T x \leq b_i, & i \in \bar{I} \end{array} \right. \qquad (\mathcal{S})$$

• Pour certifier l'absence de solution, faire la somme d'inégalités avec des poids $\lambda_i \geq 0$, pour arriver à l'inégalité

$$[\textstyle\sum_{i=1}^m \lambda_i a_i]^T x \ ? \ \sum_{i=1}^m \lambda_i b_i$$

$$\left[\begin{array}{l} ? = " < " \text{ quand } \sum_{i \in I} \lambda_i > 0 \\ ? = " \leq " \text{ quand } \sum_{i \in I} \lambda_i = 0 \end{array} \right] \qquad (!)$$

• Si (!) n'a pas de solutions, $(\mathcal{S})$ est inadmissible.

Remarque : *inégalité (!) n'a pas de solution ssi*

$$\sum_{i=1}^m \lambda_i a_i = 0$$

et, de plus,

$$\textstyle\sum_{i=1}^m \lambda_i b_i \leq 0 \text{ quand } \sum_{i \in I} \lambda_i > 0$$
$$\textstyle\sum_{i=1}^m \lambda_i b_i < 0 \text{ quand } \sum_{i \in I} \lambda_i = 0$$

$$\left\{\begin{array}{ll} a_i^T x < b_i, & i \in I \\ a_i^T x \leq b_i, & i \in \overline{I} \end{array}\right. \qquad (\mathcal{S})$$

Proposition. *On associe avec* $(\mathcal{S})$ *deux systèmes d'inégalités linéaires en* $\lambda_1, ..., \lambda_m$ *:*

$$(\mathrm{I}) : \underbrace{\left\{\begin{array}{l} \lambda_i \geq 0 \, \forall i \\ \sum_{i=1}^m \lambda_i a_i = 0 \\ \sum_{i=1}^m \lambda_i b_i \leq 0 \\ \sum_{i \in I} \lambda_i > 0 \end{array}\right.}_{0^T x < 0}, \qquad (\mathrm{II}) : \underbrace{\left\{\begin{array}{l} \lambda_i \geq 0 \, \forall i \\ \sum_{i=1}^m \lambda_i a_i = 0 \\ \sum_{i=1}^m \lambda_i b_i < 0 \\ \lambda_i = 0, i \in I \end{array}\right.}_{0^T x \leq -1}$$

Si un des systèmes (I), (II) *a une solution, alors* $(\mathcal{S})$ *n'a pas de solutions.*

$$\left\{\begin{array}{ll} a_i^T x < b_i, & i \in I \\ a_i^T x \leq b_i, & i \in \overline{I} \end{array}\right. \qquad (\mathcal{S})$$

Nous avons le résultat bien plus fort :

Théorème sur alternative linéaire. *On associe avec le système* $(\mathcal{S})$ *deux systèmes d'inégalités linéaires en* $\lambda_1, ..., \lambda_m$ *:*

$$(\mathrm{I}) : \left\{\begin{array}{l} \lambda_i \geq 0 \, \forall i \\ \sum_{i=1}^m \lambda_i a_i = 0 \\ \sum_{i=1}^m \lambda_i b_i \leq 0 \\ \sum_{i \in I} \lambda_i > 0 \end{array}\right. , \quad (\mathrm{II}) : \left\{\begin{array}{l} \lambda_i \geq 0 \, \forall i \\ \sum_{i=1}^m \lambda_i a_i = 0 \\ \sum_{i=1}^m \lambda_i b_i < 0 \\ \lambda_i = 0, i \in I \end{array}\right.$$

Système $(\mathcal{S})$ *n'a pas de solutions* **si et seulement si** *un des systèmes* (I), (II) *a une solution.*

Remarque : une solution de (I) ou (II) peut être vue comme un certificat d'incompatibilité de $(\mathcal{S})$: $(\mathcal{S})$ *est irréalisable si et seulement si un tel certificat existe*.

Remarque : les inegalites strictes de $(\mathcal{S})$ ne participent pas dans (II) $\Rightarrow$ (II) a une solution ssi le sous-système "nonstricte"

$$a_i^T x \leq b_i, \ i \in \overline{I}$$

de $(\mathcal{S})$ n'a pas de solution.

Le théorème peut être reformuler de façon suivante :

Un système fini d'inégalités linéaires n'a pas de solution ssi il est possible, en faisant une somme d'inégalités de système avec des poids admissibles (i.e., compatibles avec les operations de base avec des inégalités), *obtenir une inégalité contradictoire – soit l'inégalité* $0^T x \leq -1$, *ou* $0^T x < 0$.

L'avantage de cette formulation c'est que nous n'avons pas besoin de convertir le système en forme canonique.

Exemple : le système

$$\begin{array}{rcl} x_1 + 2x_2 & < & 5 \\ 2x_1 + 3x_2 & \geq & 3 \\ 3x_1 + 4x_2 & = & 1 \end{array}$$

n'a pas de solution – il suffit de faire la somme d'inégalités avec les poids $[-1; 2; -1]$ pour obtenir

$$0 \cdot x_1 + 0 \cdot x_2 > 0$$

Remarque : le théorème sur alternative est toujours vrais *dans une direction* – en faisant les sommes d'inégalités d'un système $(\mathcal{S})$, linéaire ou non linéaire, avec des inégalités triviales (toujours vraies), on obtient toujours une conséquence de $(\mathcal{S})$.

Cependant, "l'autre direction" dans le théorème sur alternative *linéaire* exploite fortement le fait que les inégalités du système original et une conséquence que nous recherchons sont *linéaires.*

Par exemple, l'inégalité quadratique

$$x^2 \leq 1 \qquad (!)$$

est une conséquence du système d'inégalités linéaires (et, donc, quadratiques aussi)

$$-1 \leq x \leq 1 \qquad (*)$$

Par contre, $(!)$ *ne peut pas* être représenté comme une somme d'inégalités de $(*)$ et d'inégalités linéaires et quadratiques triviales, telles que

$$0 \cdot x \leq 1,\ x^2 \geq 0,\ x^2 - 2x + 1 \geq 0, ...$$

Dualité en OL On considère le problème OL

$$\mathrm{Opt}(P) = \max_x \left\{ c^T x : Ax \leq b \right\} \qquad (P)$$

Problème dual permet de borner supérieurement la valeur optimal du *problème primal* (P). Pour le faire, on "agrège" le problème (P) :

— *on attribue aux contraintes* $a_i^T x \leq b_i$ *des coefficients non-négatifs* λ_i ("multiplicateurs de Lagrange") *et on fait la somme des contraintes avec ces coefficients, pour obtenir*

$$[A^T \lambda]^T x \leq b^T \lambda \qquad (!)$$

Observation : par construction, cette inégalité est une conséquence du système $Ax \leq b$, et est ainsi satisfaite sur toute solution realisable de (P).

— *Si* $A^T \lambda = c$, *alors (!) dit que that* $b^T \lambda$ *est une borne supérieure sur* $c^T x$ *sur tout domaine realisable de (P), et donc*

$$b^T \lambda \geq \mathrm{Opt}(P).$$

$$\mathrm{Opt}(P) = \max_x \left\{ c^T x : Ax \leq b \right\} \qquad (P)$$

• *Maintenant nous pouvons rechercher la meilleure – la plus petite – borne supérieure du* $\mathrm{Opt}(P)$ *qu'on puisse obtenir par cette construction. Ainsi on arrive au problème suivant*

$$\mathrm{Opt}(D) = \min_\lambda \left\{ b^T \lambda : \ A^T \lambda = c, \ \lambda \geq 0 \right\}, \qquad (D)$$

appelé problème dual de (P).

Observation : la construction de la borne pour la valeur optimale peut être appliquée à *tout* programme OL, quelque soit le format. Par exemple, en l'appliquant au programme primal

$$\mathrm{Opt}(P) = \max_x \left\{ c^T x : \begin{array}{lll} Px & \underbrace{\leq}_{\lambda_\ell} & p \quad (\ell) \\ Qx & \underbrace{\geq}_{\lambda_g} & q \quad (g) \\ Rx & \underbrace{=}_{\lambda_e} & r \quad (e) \end{array} \right\} \qquad (P)$$

on obtient le problème dual

$$\mathrm{Opt}(D) \;=\; \min_{[\lambda_\ell;\lambda_g,\lambda_e]} \left\{ \begin{array}{l} p^T\lambda_\ell + q^T\lambda_g + r^T\lambda_e : \\ \lambda_\ell \geq 0,\ \lambda_g \leq 0 \\ P^T\lambda_\ell + Q^T\lambda_g + R^T\lambda_e = c \end{array} \right\} \qquad (D)$$

• *Attention aux notations : les types* $\leq$, $\geq$, $=$ *des contraintes de (P) sont préservés par les vecteurs de coefficients de Lagrange affectés* λ_ℓ, λ_g, λ_e.

En résumé :

	Primal "min" Dual "max"	Primal "max" Dual "min"
Contrainte primale	Variable duale	Variable duale
"$\geq$" "$\leq$" "$=$"	"$\geq$" "$\leq$" sans contrainte	"$\leq$" "$\geq$" sans contrainte
Variable primale	Contrainte duale	Contrainte duale
"$\geq$" "$\leq$" sans contrainte	"$\leq$" "$\geq$" "$=$"	"$\geq$" "$\leq$" "$=$"

$$\mathrm{Opt}(P) = \max_x \left\{ c^T x : \ Px \leq p \ (\ell), \ Qx \geq q \ (g), \ Rx = r \ (e) \right\} \qquad (P)$$

$$\mathrm{Opt}(D) = \min_{[\lambda_\ell; \lambda_g, \lambda_e]} \left\{ p^T \lambda_\ell + q^T \lambda_g + r^T \lambda_e : \begin{array}{l} \lambda_\ell \geq 0, \ \lambda_g \leq 0 \\ P^T \lambda_\ell + Q^T \lambda_g + R^T \lambda_e = c \end{array} \right\} \qquad (D)$$

Théorème de dualité en OL : *soit (P) le problème OL primal avec sont dual (D). Alors*

[Symétrie primal-dual] *Dualité est symétrique : (D) est un programme OL, est le dual de (D) est (équivalent à) (P).*

[Dualité faible] *Nous avons toujours* $\mathrm{Opt}(D) \geq \mathrm{Opt}(P)$.

Attention : cette inégalité correspond au problème primal de *maximisation*. Plus généralement, dualité faible dit que dans le couple primal-dual, la valeur optimale du *problème de minimisation* est $\geq$ la valeur optimal du problème de maximisation.

[Dualité forte] *Les propriétés suivantes sont équivalentes :*

- *un des problèmes est réalisable et borné*
- *deux problèmes sont solubles*
- *deux problèmes sont réalisables*

et quand une (et, donc, toutes) de ces propriétés a lieu, nous avons

$$\mathrm{Opt}(P) = \mathrm{Opt}(D).$$

$$\mathrm{Opt}(P) = \max_x \left\{ c^T x : \ Px \le p \ (\ell),\ Qx \ge q \ (g),\ Rx = r \ (e) \right\} \qquad (P)$$

$$\mathrm{Opt}(D) = \min_{[\lambda_\ell;\lambda_g,\lambda_e]} \left\{ p^T\lambda_\ell + q^T\lambda_g + r^T\lambda_e : \begin{array}{l} \lambda_\ell \ge 0,\ \lambda_g \le 0 \\ P^T\lambda_\ell + Q^T\lambda_g + R^T\lambda_e = c \end{array} \right\} \qquad (D)$$

Verification de la symétrie primal-dual. On réécrit (D) comme un problème de maximisation :

$$-\mathrm{Opt}(D) = \max_{[\lambda_\ell;\lambda_g,\lambda_e]} \left\{ -p^T\lambda_\ell - q^T\lambda_g - r^T\lambda_e : \begin{array}{l} \lambda_g \le 0,\ \lambda_\ell \ge 0 \\ P^T\lambda_\ell + Q^T\lambda_g + R^T\lambda_e = c \end{array} \right\}$$

et, en appliquant les règles de construction de dual, on obtient

$$\min_{[x_\ell;x_g;x_e]} \left\{ c^T x_e : \ x_\ell \ge 0,\ x_g \le 0, \begin{array}{l} Px_e + x_g = -p \\ Qx_e + x_\ell = -q \\ Rx_e = -r \end{array} \right\}.$$

En posant $x_e = -x$ et en éliminant x_g et x_e, le problème dual du (D) devient

$$\min_x \left\{ -c^T x : \ Px \le p,\ Qx \ge q,\ Rx = r \right\},$$

qui est équivalent à (P).

$$\mathrm{Opt}(P) = \max_x \left\{ c^T x : \; Px \le p \; (\ell), \; Qx \ge q \; (g), \; Rx = r \; (e) \right\} \qquad (P)$$

$$\mathrm{Opt}(D) = \min_{[\lambda_\ell;\lambda_g,\lambda_e]} \left\{ p^T\lambda_\ell + q^T\lambda_g + r^T\lambda_e : \quad \begin{array}{l} \lambda_\ell \ge 0, \; \lambda_g \le 0 \\ P^T\lambda_\ell + Q^T\lambda_g + R^T\lambda_e = c \end{array} \right\} \qquad (D)$$

Consequences immédiates.

— **Théorème** Si au moins un des problemes (P), (D) est réalisable, nous avons

$$\mathrm{Opt}(P) = \mathrm{Opt}(D). \qquad \text{[pourquoi ça ?]}$$

— **Conditions d'optimalité en OL** *Soit* x *et* $\lambda = [\lambda_\ell; \lambda_g; \lambda_e]$ *une paire des solutions réalisables de* (P) *et* (D). *Elle est comprise des solutions optimales*

• [saut de dualité nul] *si et seulement si le saut de dualité (duality gap) évalué sur cette paire de solutions, est nul :*

$$\mathrm{DualityGap}(x, \lambda) := [p^T\lambda_\ell + q^T\lambda_g + r^T\lambda_e] - c^T x \; = 0$$

• [complémentarité] *si et seulement si les produits de tous les multiplicateurs de Lagrange* λ_i*et des résidus de la contrainte correspondante primale sont nuls :*

$$\forall i : \; [\lambda_\ell]_i [p - Px]_i = 0 \;\; \& \;\; \forall j : \; [\lambda_g]_j [q - Qx]_j = 0.$$

Verification Nous sommes dans le cas quand les deux problemes sont réalisables et donc solubles avec les mêmes valeurs optimales. Alors

$$\begin{aligned}\mathrm{DualityGap}(x,\lambda) \;:=\; & \Big[[p^T\lambda_\ell + q^T\lambda_g + r^T\lambda_e] - \mathrm{Opt}(D)\Big] \\ & + \Big[\mathrm{Opt}(P) - c^T x\Big]\end{aligned}$$

Pour toute paire de solutions primal-dual réalisables, les expressions entre les crochets sont non-négatives $\Rightarrow$ *le saut de dualité, évalué sur une paire primal-dual realisable est* ≥ 0 *et est nul ssi les deux expressions sont nulles* $\Leftrightarrow$ *ssi* x *est une solution primale optimale et* λ *est duale optimale.*

- On remarque que

$$\begin{aligned}\mathrm{DualityGap}(x,\lambda) &= [p^T\lambda_\ell + q^T\lambda_g + r^T\lambda_e] - c^T x \\ &= [p^T\lambda_\ell + q^T\lambda_g + r^T\lambda_e] - [P^T\lambda_\ell + Q^T\lambda_g + R^T\lambda_e]^T x \\ &= \lambda_\ell^T[p - Px] + \lambda_g^T[q - Qx] + \lambda_e^T[r - Rx] \\ &= \textstyle\sum_i [\lambda_\ell]_i [p - Px]_i + \sum_j [\lambda_g]_j [q - Qx]_j\end{aligned}$$

$\Rightarrow$ tous les termes dans la sommes sont non-négatifs

$\Rightarrow$ *le saut de dualité est nul ssi la complémentarité a lieu.*

Fonction de coût d'un programme linéaire I

$$\mathrm{Opt}(c) = \max_x \left\{ c^T x : Ax \leq b \right\}. \qquad (P[c])$$

Maintenant on suppose que A, b sont fixes, et que c varie, et on s'intéresse aux propriétés $\mathrm{Opt}(c)$ comme fonction de c.

Hypothèse : $(P[\cdot])$ est realisable (ce fait est indépendant de la valeur de c).

Théorème *Soit $\bar{c}$ tel que $\mathrm{Opt}(\bar{c}) < \infty$, et soit $\bar{x}$ une solution optimale de $(P[\bar{c}])$. Alors,*

$$\forall c : \ \mathrm{Opt}(c) \geq \mathrm{Opt}(\bar{c}) + \bar{x}^T [c - \bar{c}]. \qquad (!)$$

En effet, nous avons

$$\mathrm{Opt}(c) \geq c^T \bar{x} = \bar{c}^T \bar{x} + [c - \bar{c}]^T \bar{x} = \mathrm{Opt}(\bar{c}) + \bar{x}^T [c - \bar{c}].$$

Fonction de coût d'un programme linéaire II

Soit le programme linéaire

$$\mathrm{Opt}(b) = \max_x \left\{ c^T x : Ax \leq b \right\}. \qquad (P[b])$$

On suppose que A, c sont fixes, et b varie, et nous sommes intéressé par les proprietes de la valeur optimale $\mathrm{Opt}(b)$ comme fonction de b.

Remarque : *Quand b est tel que $(P[b])$ est réalisable, la propriété du problème d'être/ne pas être borné ne dépend pas de la valeur de b.*

En effet, le problème $(P[b])$ n'est pas borné ssi il existe $d : Ad \leq 0,\ c^T d > 0$, et ceci est indépendant de la valeur de b.

Hypothèse : il existe b tel que $P([b])$ est réalisable et borné
$\Rightarrow$ $P([b])$ est borné s'il est réalisable.

Fonction $\mathrm{Opt}(b)$ est monotone en b :

$$b' \leq b'' \Rightarrow \mathrm{Opt}(b') \leq \mathrm{Opt}(b'').$$

$$\mathrm{Opt}(b) = \max_x \left\{ c^T x : Ax \leq b \right\}. \qquad (P[b])$$

L'information supplémentaire sur $\mathrm{Opt}(b)$ peut être obtenue par dualité. Le problème duale de $(P[b])$ est

$$\min_\lambda \left\{ b^T \lambda : \lambda \geq 0,\ A^T \lambda = c \right\}. \qquad (D[b])$$

Par le théorème de dualité en OL, sous l'hypothèse, $(D[b])$ *est realisable pour tout* b, *et*

$$\mathrm{Opt}(b) = \min_\lambda \left\{ b^T \lambda : \lambda \geq 0,\ A^T \lambda = c \right\}. \qquad (*)$$

Observation : *Soit* $\bar{b}$ *tel que* $\mathrm{Opt}(\bar{b}) > -\infty$, *et donc* $(D[\bar{b}])$ *est soluble, et soit* $\bar{\lambda}$ *une solution optimale de* $(D[\bar{b}])$. *Alors nous avons*

$$\forall b : \ \mathrm{Opt}(b) \leq \mathrm{Opt}(\bar{b}) + \bar{\lambda}^T [b - \bar{b}]. \qquad (!)$$

En effet, par $(*)$ nous avons $\mathrm{Opt}(\bar{b}) = \bar{\lambda}^T \bar{b}$ et $\mathrm{Opt}(b) \leq \bar{\lambda}^T b$, donc,

$$\mathrm{Opt}(b) \leq \bar{\lambda}^T \bar{b} + \bar{\lambda}^T [b - \bar{b}] = \mathrm{Opt}(\bar{b}) + \bar{\lambda}^T [b - \bar{b}].$$

Loi de décroissance des rendement marginaux

On considère la fonction de β définie par

$$\mathrm{Opt}(\beta) = \max_x \left\{ c^T x : \ Px \leq p, \ q^T x \leq \beta \right\} \qquad (P_\beta)$$

Interprétation : x est un plan de production, $q^T x$ est le prix des ressources utilisées par x, β est l'investissement dans les ressources, $\mathrm{Opt}(\beta)$ est le retour maximal sur l'investissement β.

Comme ci-dessus, pour β tel que (P_β) est réalisable, indépendamment de la valeur de β, le problème est soit toujours borné, soit toujours non-borné. Supposons que le problème est borne dans notre cas, alors

- Le domaine $\mathrm{Dom}\,\mathrm{Opt}(\cdot)$ de la fonction $\mathrm{Opt}(\cdot)$ est un rayon non vide $\underline{\beta} \leq \beta < \infty$ avec $\underline{\beta} \geq -\infty$, et
- $\mathrm{Opt}(\beta)$ est non-décroissante et *concave*. Monotonie et concavité impliquent que si

$$\underline{\beta} \leq \beta_1 < \beta_2 < \beta_3,$$

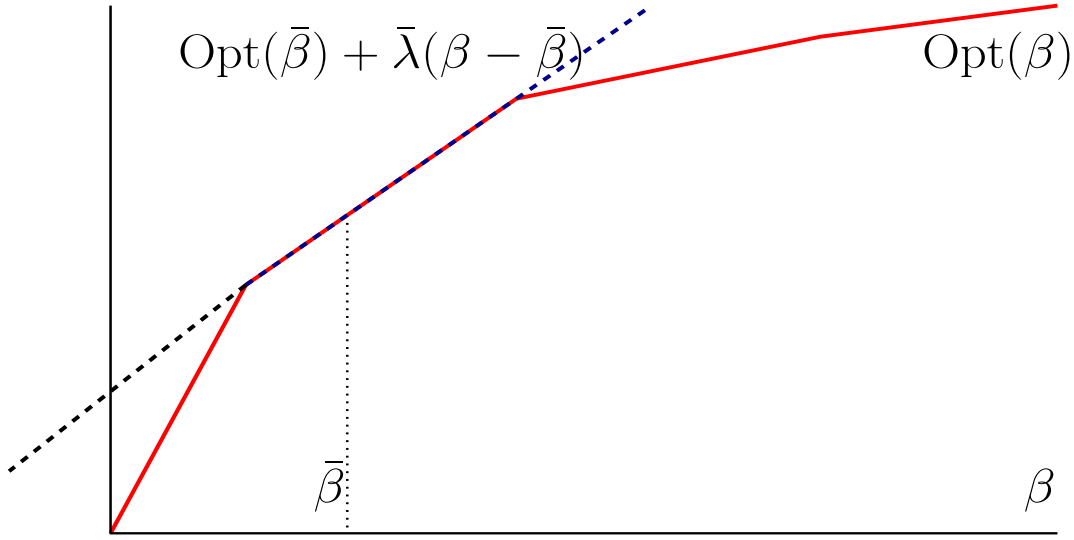
alors

$$\frac{\mathrm{Opt}(\beta_2) - \mathrm{Opt}(\beta_1)}{\beta_2 - \beta_1} \geq \frac{\mathrm{Opt}(\beta_3) - \mathrm{Opt}(\beta_2)}{\beta_3 - \beta_2}.$$

Autrement dit

le retour pour 1€ d'investissement décroît (ne change pas dans le meilleur cas) *quand l'investissement β grandit.*

$\Leftrightarrow$ *Loi de décroissance des rendements marginaux* en économie.



Autre interprétation : accroissement des prix marginaux

On considère la fonction de β définie par

$$\mathrm{Opt}(\beta) = \min_x \left\{ c^T x : \ Px \geq p, \ q^T x \geq \beta \right\} \qquad (P'_\beta)$$

Interprétation : x est un plan de production, $q^T x$ la *quantité du produit fabriqué,* β est la *demande* en produit, $\mathrm{Opt}(\beta)$ est le coût de fabrication des produit pour satisfaire la demande β.

- Comme dans le cas précèdent, le domaine $\mathrm{Dom}\,\mathrm{Opt}(\cdot)$ de la fonction $\mathrm{Opt}(\cdot)$ est un rayon non vide $\overline{\beta} \geq \beta > -\infty$ avec $\overline{\beta} \leq \infty$, et
- $\mathrm{Opt}(\beta)$ est non-croissante et *convexe*. Ainsi, si

$$\beta_1 < \beta_2 < \beta_3 \leq \overline{\beta},$$

alors

$$\frac{\mathrm{Opt}(\beta_2) - \mathrm{Opt}(\beta_1)}{\beta_2 - \beta_1} \leq \frac{\mathrm{Opt}(\beta_3) - \mathrm{Opt}(\beta_2)}{\beta_3 - \beta_2}.$$

Ce qui peut-être exprimée en PL

Exemple : problème d'ordonnancement

On doit planifier n taches sur la grappe de m serveurs de calcul homogènes. Chaque tache a la durée fixe t_i, $i = 1, ..., n$, et peut être traitée par tout serveur. On veut distribuer des taches sur les serveurs de façon à *minimiser la durée totale du traitement.*

Formulation MinMax

$$\begin{array}{rll} \text{minimiser} & \max_{1 \le j \le m} \sum_i^n t_i x_{ij} & \\ \text{sous contraintes} & \sum_{j=1}^m x_{ij} = 1, & i = 1, ..., n \\ & x_{ij} \in \{0, 1\}, & i = 1, ..., n, j = 1, ..., m. \end{array}$$

— variable $x_{ij} = 1$ si la tache i est traitée par le serveur j ; $x_{ij} = 0$ sinon – problème combinatoire (difficile)

— l'objectif du programme n'est pas linéaire. La fonction non-linéaire "max" peut être facilement réduite à un objectif avec des contraintes linéaires :
on pose les contraintes

$$z \ge \sum_i^n t_i x_{ij}, \ j = 1, ..., m,$$

et le nouvel objectif : minimiser z.

Remarque : dans un *programme linéaire*, l'objectif est une fonction linéaire de la variable de decision x et les contraintes sont les equations ou les inégalités linéaires non-strictes.

La propriété d'un problème PM d'etre un programme OL est une propriété de la représentation. Les programmes seront classifiés selon leur présentation, pas selon ce à quoi ils sont equivalents/peuvent être réduits.

Ainsi, le problème de programmation mathématique

$$\min_x \left\{ x_1 : \begin{array}{rcl} x_1 + x_2 & \leq & 20 \\ x_1 - x_2 & = & 5 \\ x_1, x_2 & \geq & 0 \end{array} \right\} \tag{1}$$

est un programme OL.
Mais le problème

$$\min_x \left\{ |x_1 - 2x_2| : \begin{array}{rcl} x_1 + x_2 & \leq & 20 \\ x_1 - x_2 & = & 5 \\ x_1, x_2 & \geq & 0 \end{array} \right\} \tag{1'}$$

n'est pas un programme OL, car l'objectif de $(1')$ est non-linéaire.

Optimisation "linéaire par morceaux"

- Fonction linéaire : une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est linéaire si

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \quad \forall x, y \in \mathbb{R}^n, \quad \forall \alpha, \beta \in \mathbb{R}$$

Caractérisation : f est linéaire ssi $f = a^T x$ pour un $a \in \mathbb{R}^n$.

- Fonction affine : une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est affine si

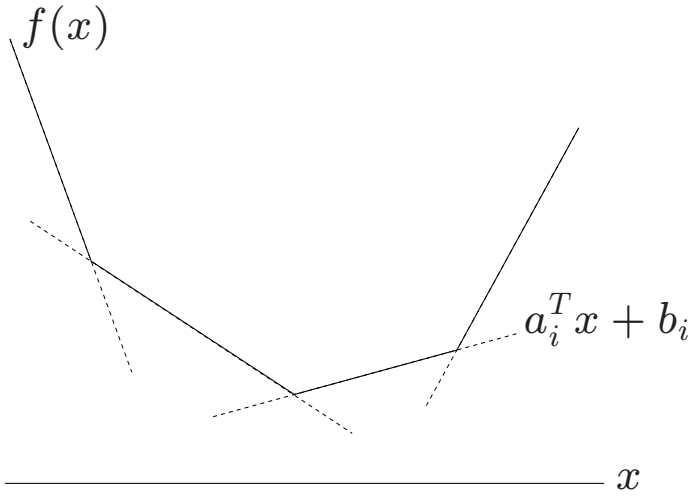
$$f(\alpha x + (1-\alpha) y) = \alpha f(x) + (1-\alpha) f(y) \quad \forall x, y \in \mathbb{R}^n, \quad \forall \alpha \in \mathbb{R}$$

Caractérisation : f est affine ssi $f = a^T x + b$ pour $a \in \mathbb{R}^n$, $b \in \mathbb{R}$.

- Fonction linéaire par morceau : $f : \mathbb{R}^n \to \mathbb{R}$ est *(convexe) linéaire par morceau* si

$$f(x) = \max_{i=1,..,m} (a_i^T x + b_i).$$

f est paramétrée par m n-vecteurs a_i et m scalaires b_i.



Minimisation linéaire par morceaux

$$\min \left\{ f(x) = \max_{i=1,\ldots,m} a_i^T x + b_i \right\}$$

- *Modèle OL équivalent* avec la variable x et *la variable auxiliaire* t :

$$\min\{t : \ a_i^T x + b_i \leq t \ \forall i\}$$

- PL *en forme canonique* (notation matricielle) :

$$\max\{\bar{c}^T \bar{x} : \ \bar{A}\bar{x} \leq \bar{b}\},$$

avec

$$\bar{x} = \begin{bmatrix} x \\ t \end{bmatrix}, \ \bar{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \ \bar{A} = \begin{bmatrix} a_1^T & -1 \\ \vdots & \vdots \\ a_m^T & -1 \end{bmatrix}, \ \bar{b} = \begin{bmatrix} -b_1 \\ \vdots \\ b_m \end{bmatrix}.$$

Minimisation de la somme des fonctions linéaires par morceaux

$$\min\left\{\max_{i=1,\dots,m}(a_i^T x + b_i) + \max_{j=1,\dots,p}(c_j^T x + d_j)\right\} \Leftrightarrow$$

$$\min\left\{\max_{\substack{i=1,\dots,m\\ j=1,\dots,p}} (a_i + c_j)^T x + (b_i + d_j)\right\}$$

- *PL équivalent* avec $m + p$ inégalités

$$\min\left\{t_1 + t_2 :\ a_i^T x + b_i \le t_1\ \forall i,\ c_j^T x + d_j \le t_2\ \forall j\right\}$$

Remarque : pour un x fixe, le minimum en t est

$$t_1 = \max_{i=1,\dots,m}(a_i^T x + b_i), \quad t_2 = \max_{j=1,\dots,p}(c_j^T x + d_j).$$

- *PL en forme canonique :* $\max\{\bar{c}^T \bar{x} :\ \bar{A}\bar{x} \le \bar{b}\}$, avec

$$\bar{x} = \begin{bmatrix} x \\ t_1 \\ t_2 \end{bmatrix}, \quad \bar{c} = \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}, \quad \bar{A} = \begin{bmatrix} a_1^T & -1 & 0 \\ \vdots & \vdots & \vdots \\ a_m^T & -1 & 0 \\ c_1^T & 0 & -1 \\ \vdots & \vdots & \vdots \\ c_p^T & 0 & -1 \end{bmatrix}, \quad \bar{b} = \begin{bmatrix} -b_1 \\ \vdots \\ -b_m \\ -d_1 \\ \vdots \\ -d_p \end{bmatrix}.$$

Approximation de Tchebychev ℓ_∞ : $\min\{\|Ax-b\|_\infty\}$.

- *PL équivalent* après la discrétisation (avec la variable x et variable auxiliaire t) :

$$\min\{t : \ -t\mathbf{1} \leq Ax-b \leq t\mathbf{1}\}$$

- *PL en notation matricielle :* $\min\{\bar{c}^T\bar{x} : \ \bar{A}\bar{x} \leq \bar{b}\}$, avec

$$\bar{x} = \begin{bmatrix} x \\ t \end{bmatrix}, \ \bar{c} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \ \bar{A} = \begin{bmatrix} A & -\mathbf{1} \\ -A & -\mathbf{1} \end{bmatrix}, \ \bar{b} = \begin{bmatrix} b \\ -b \end{bmatrix}.$$

Minimisation de la norme ℓ_1 : $\min\{\|Ax-b\|_1\}$.

- *PL équivalent* avec la variable x et vecteur auxiliaire u :

$$\min\left\{\sum_{i=1}^{m} u_i : \ -u \leq Ax-b \leq u\right\}$$

- *PL en notation matricielle :* $\min\{\tilde{c}^T\tilde{x} : \ \tilde{A}\tilde{x} \leq \tilde{b}\}$, avec

$$\tilde{x} = \begin{bmatrix} x \\ u \end{bmatrix}, \ \tilde{c} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \ \tilde{A} = \begin{bmatrix} A & -I \\ -A & -I \end{bmatrix}, \ \tilde{b} = \begin{bmatrix} b \\ -b \end{bmatrix}.$$

• Le même problème peu être *formulé comme un PL different* en introduisant des variables auxiliaires

$$u \geq 0,\ v \geq 0,\ u - v = Ax - b.$$

On obtiens ainsi le programme

$$\min \left\{ \sum_{i=1}^{m} (u_i + v_i) :\ Ax - b = u - v,\ u \geq 0,\ v \geq 0 \right\}$$

• *PL en notation matricielle :* $\min\{\tilde{c}^T \tilde{x} :\ \tilde{A}\tilde{x} \leq \tilde{b}\}$, avec

$$\tilde{x} = \begin{bmatrix} x \\ u \\ v \end{bmatrix},\ \tilde{c} = \begin{bmatrix} 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix},\ \tilde{A} = \begin{bmatrix} A & -I & I \\ 0 & -I & 0 \\ 0 & 0 & -I \end{bmatrix},\ \tilde{b} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}.$$

Application statistique : régression robuste

Étant données les observations $\{x_i \in \mathbb{R}^n,\, y_i \in \mathbb{R}\}_{i=1}^m$ dans le modèle

$$y_i = \theta_*^T x_i + \xi_i \qquad [\xi_i : \text{ bruit d'observation}]$$

on cherche à retrouver le vecteur de paramètres θ_*.

Dans le cas $m \gg n$, une approche classique pour estimer θ_* consiste à se donner une "fonction de perte" $\phi(u, v)$ et de choisir l'estimation $\widehat{\theta}$ de θ_* qui minimise en θ l'erreur de prediction

$$\phi\Big([y_1; ...; y_m], [\theta^T x_1; ...; \theta^T x_m]\Big)$$

des sorties observées par les sorties du modèle $z = \theta^T x$, appliqué aux régresseurs observés $x_1, ..., x_m$.

• En notant $X = [x_1^T; x_2^T; ...; x_m^T]$ la matrice de régresseurs, la procédure d'estimation s'écrit

$$\widehat{\theta} \in \operatorname*{Argmin}_{\theta} \phi(y, X\theta) \qquad [y = [y_1; ...; y_m]]$$

(notation $\operatorname{Argmin}_x f \quad \Leftrightarrow$ ensemble de minimiseurs de f en x).

• Le choix de la perte $\phi(\cdot,\cdot)$ dépend de la distribution de bruit ξ.

La perte couramment utilisées est la perte quadratique $\phi(u,v) = \|u-v\|_2$, correspondant au cas du bruit blanc normal ($\xi_i \sim \mathcal{N}(0,\sigma^2)$ sont indépendants) ou, plus généralement, au cas e ξ_i i.i.d. avec la moyenne nulle et la variance finie
$\Rightarrow$ méthode de moindres carres $\min_\theta \sum_{i=1}^m (y_i - x_i^T\theta)^2$.

Dans certains cas l'estimation se réduit au problème OL :

— régression ℓ_1 : $\phi(u,v) = \|u-v\|_1 := \sum_{i=1}^m |u_i - v_i|$. Dans ce cas le problème d'estimation s'écrit

$$\min_\theta \sum_{i=1}^m |y_i - x_i^T\theta| \;\Leftrightarrow\; \min_{\theta,\tau} \left\{ \tau : \sum_{i=1}^m |y_i - x_i^T\theta| \le \tau \right\} \qquad (\ell_1)$$

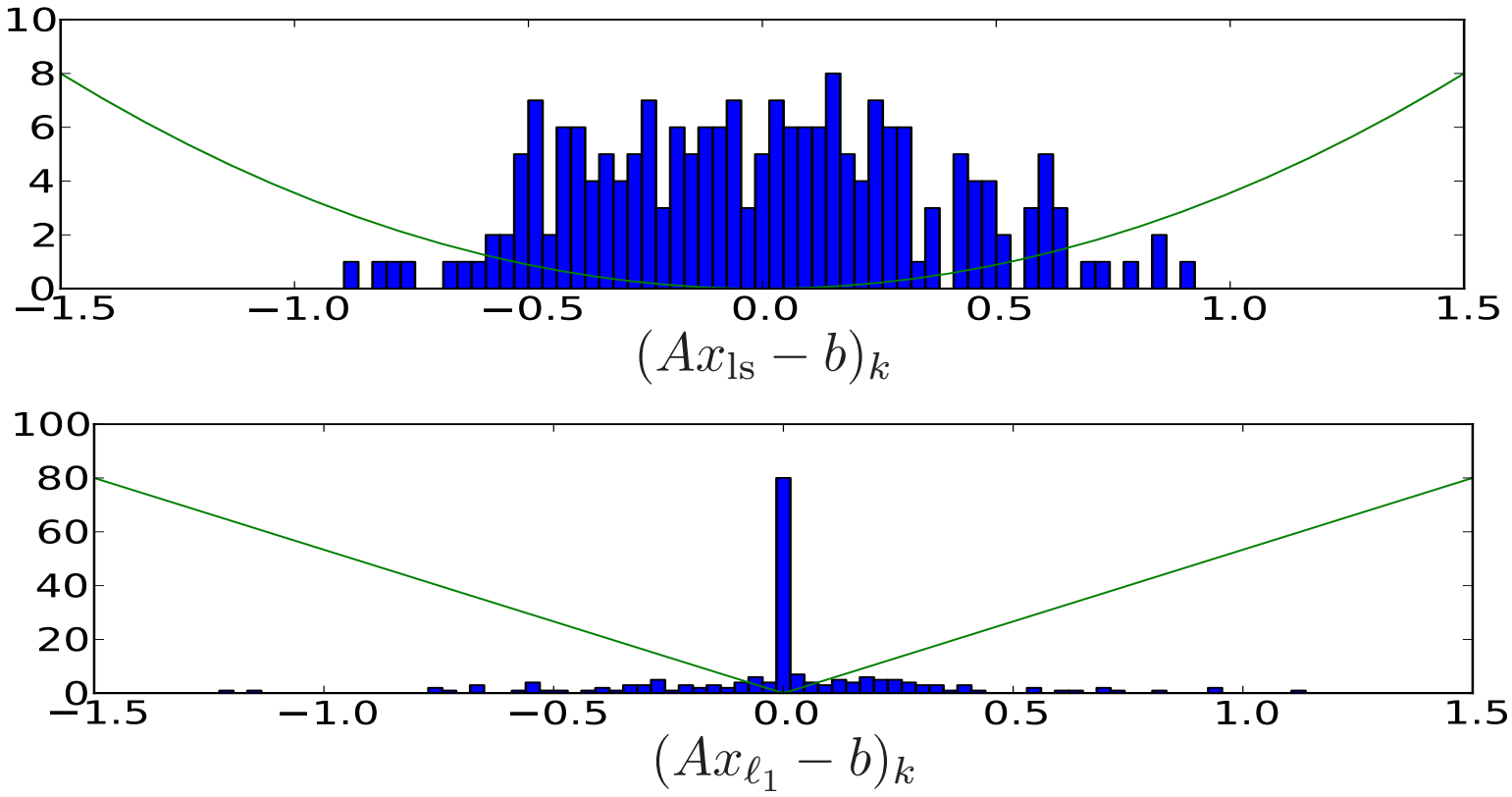
— régression ℓ_∞ : $\phi(u,v) = \|u-v\|_\infty := \max_i |u_i - v_i|$, on doit résoudre

$$\min_\theta \max_{1\le i\le m} |y_i - x_i^T\theta| \;\Leftrightarrow\; \min_{\theta,\tau} \left\{ \tau : \max_{1\le i\le m} |y_i - x_i^T\theta| \le \tau \right\} \qquad (\ell_\infty)$$

Comment ça marche – comparaison avec les moindres carrés

Soit $A \in \mathbb{R}^{200\times 80}$, $b \in \mathbb{R}^{200}$ matrices aléatoires, et soit

$$x_{\ell s} \in \operatorname*{Argmin}_x \|Ax-b\|_2, \quad x_{\ell_1} \in \operatorname*{Argmin}_x \|Ax-b\|_1.$$

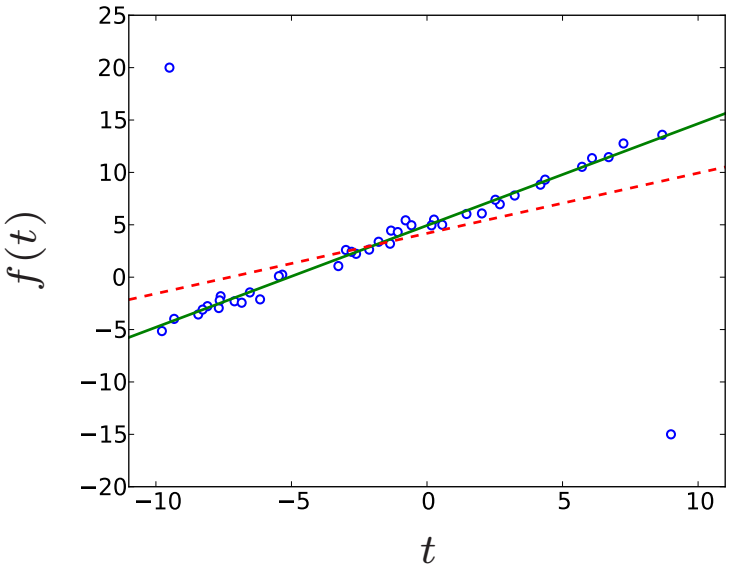


Comment ça marche – régression simple

— m observations bruitées (t_i, y_i) de la fonction affine $f(t) = \alpha + \beta t$

— Problème à résoudre : min $\|Ax - b\|$ avec

$$x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad A = \begin{bmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix}, \quad b = \begin{bmatrix} y_1 \\ \vdots y_m \end{bmatrix}.$$



- en pointillé : $\min_x \|Ax - b\|_2$
- en continu : $\min_x \|Ax - b\|_1$

$\Rightarrow$ régression ℓ_1 est plus robuste par rapport aux observation aberrantes

Application statistique : acquisition compressée (Compressed Sensing)

Nous avons une observation m-dimensionnel

$$y = [y_1; ...; y_m] = X\theta_* + \xi$$
$$[X \in \mathbb{R}^{m\times n} : \text{ matrice d'acquisition}, \xi : \text{ bruit d'observation}]$$

d'un "signal" inconnu $\theta_* \in \mathbb{R}^n$ avec $m \ll n$, et on cherche à estimer θ_*.

On s'interesse ici au cas $m \ll n$, quand le système soluble

$$X\theta = y$$

en variables θ possède l'infinité de solutions

$\Rightarrow$ Même sans bruit d'observation, *on ne peut pas identifier θ_* sans hypothèses supplémentaires.*

• En *Compressed Sensing (acquisition compressée)* on suppose que θ_* *est creux* — *ait au plus $s \ll m$ coefficients non-nuls.*

Remarque : *soit* $\xi = 0$, *et soit toute sous-matrice* $m \times 2s$ *de* X *de rang* $2s$ (ce qui est souvent le cas quand $m \gg 2s$). *Alors* θ_* *est la solution optimal du problème d'optimisation*

$$\min_{\theta} \{\|\theta\|_0 : \ X\theta = y\} \quad [\|\theta\|_0 = \mathrm{Card}\{j : \ \theta_j \neq 0\}] \qquad (\ell_0)$$

Mauvaise nouvelle : (ℓ_0) *est un problème combinatoire difficile.*

• Un remède partiel : on remplace l'objectif "difficile" $\|\theta\|_0$ par un objective "facile :"

$$\|\theta\|_1 = \sum_i |\theta_i|,$$

et on arrive au problème de *minimisation de la norme* ℓ_1 *:*

$$\begin{aligned} \widehat{\theta} \ &= \ \operatorname*{argmin}_{\theta} \{\|\theta\|_1 : \ X\theta = y\} \\ &\Leftrightarrow \ \min_{\theta,z} \left\{ \sum_j z_j : \ X\theta = y, \ -z_j \leq \theta_j \leq z_j \ \forall j \leq n \right\}. \end{aligned}$$

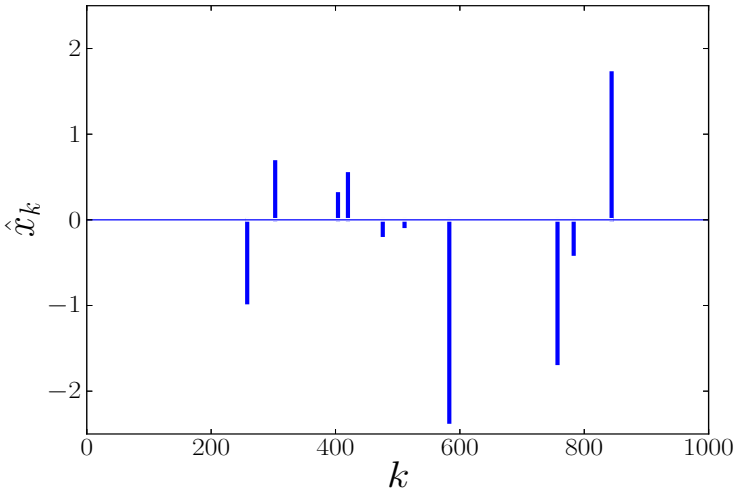
Comment ça marche – acquisition compressée

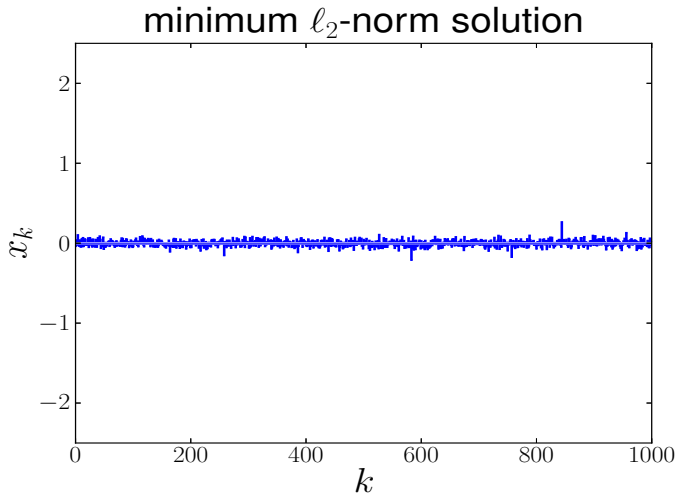
- signal exacte $x_* \in \mathbb{R}^{1000}$,

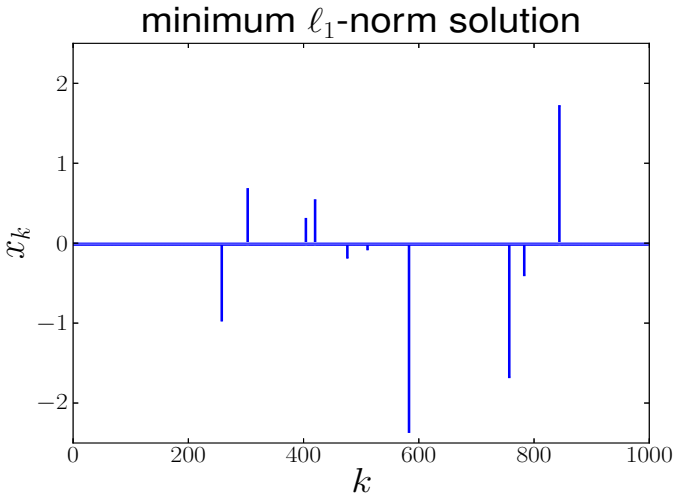
10 coefficients non-nuls

- matrice $A \in \mathbb{R}^{100\times 1000}$ aléatoire,

cas sans bruit $b = Ax_*$







• Quand l'observation est bruitée, c.-à-d. que

$$y = X\theta_* + \xi,$$

et on connaît une borne δ de norme $\|\xi\|$ du bruit, *l'estimateur de θ_* par minimisation de la norme ℓ_1* devient

$$\widehat{\theta} \in \operatorname*{Argmin}_{\theta} \{\|\theta\|_1 : \|X\theta - y\| \leq \delta\}.$$

Un autre estimateur

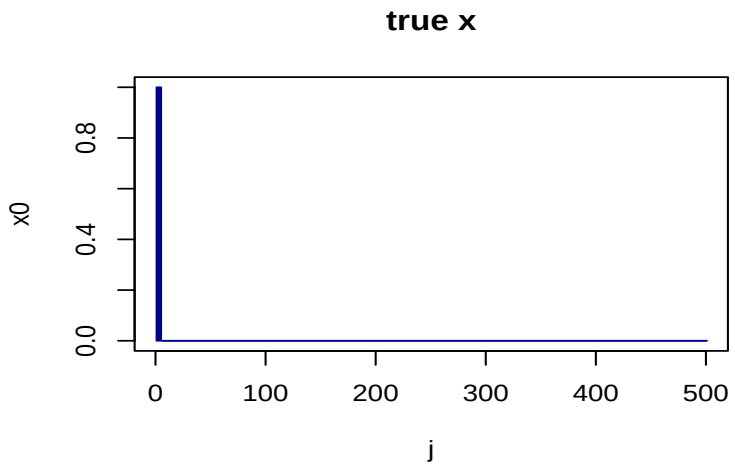
$$\widehat{\theta}_{DS} \in \operatorname*{Argmin}_{\theta} \left\{\|\theta\|_1 : \|X^T(X\theta - y)\|_\infty \leq \delta\right\}.$$

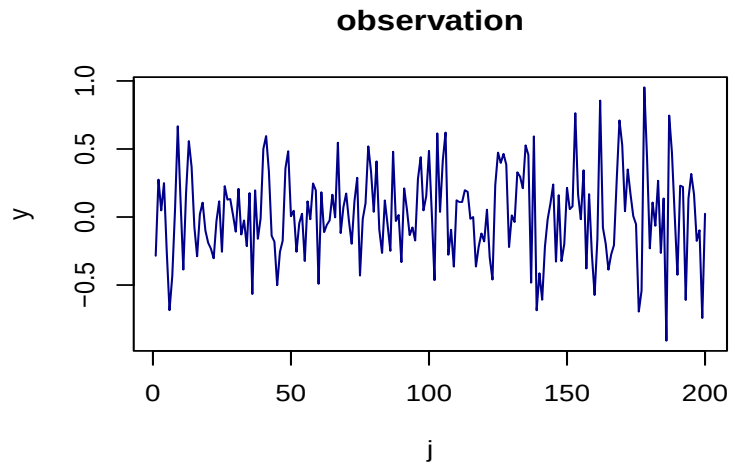
par *"Dantzig Selector"* de θ_*, est la "composante θ" de solution optimale du programme OL :

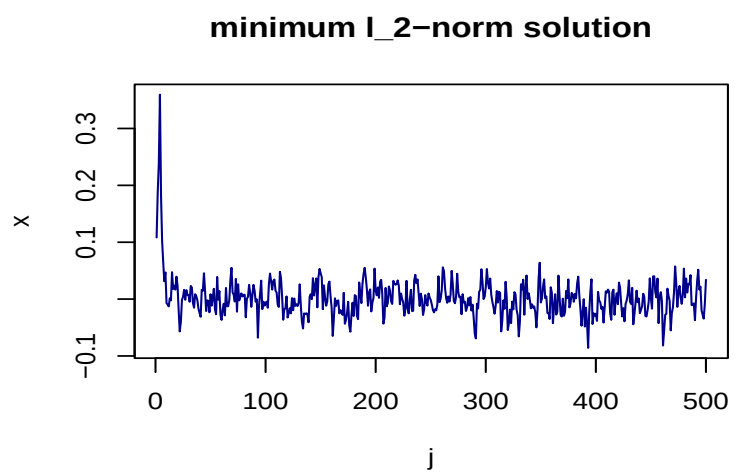
$$\operatorname*{Argmin}_{\theta,\, z} \left\{ \sum\nolimits_j z_j : \begin{array}{l} -\delta \leq [X^T(X\theta - y)]_i \leq \delta \ \forall i \leq m \\ -z_j \leq \theta_j \leq z_j,\ z_j \geq 0 \ \forall j \leq n \end{array} \right\}.$$

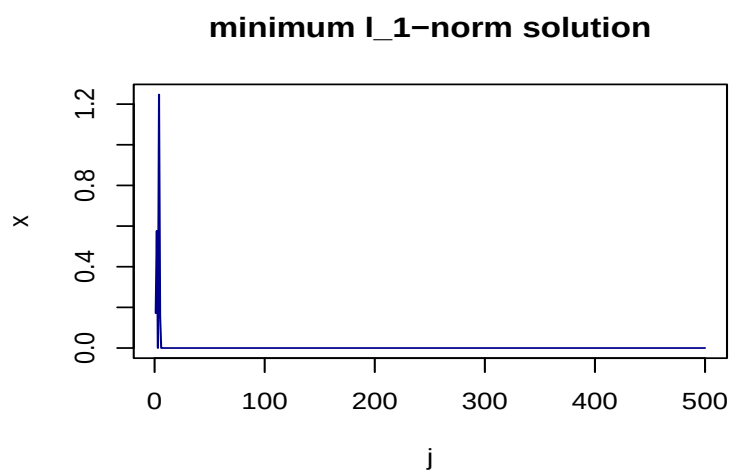
Comment ça marche – Dantzig Selector

- matrice $A \in \mathbb{R}^{200\times 500}$ aléatoire, $\sigma = 0.25$







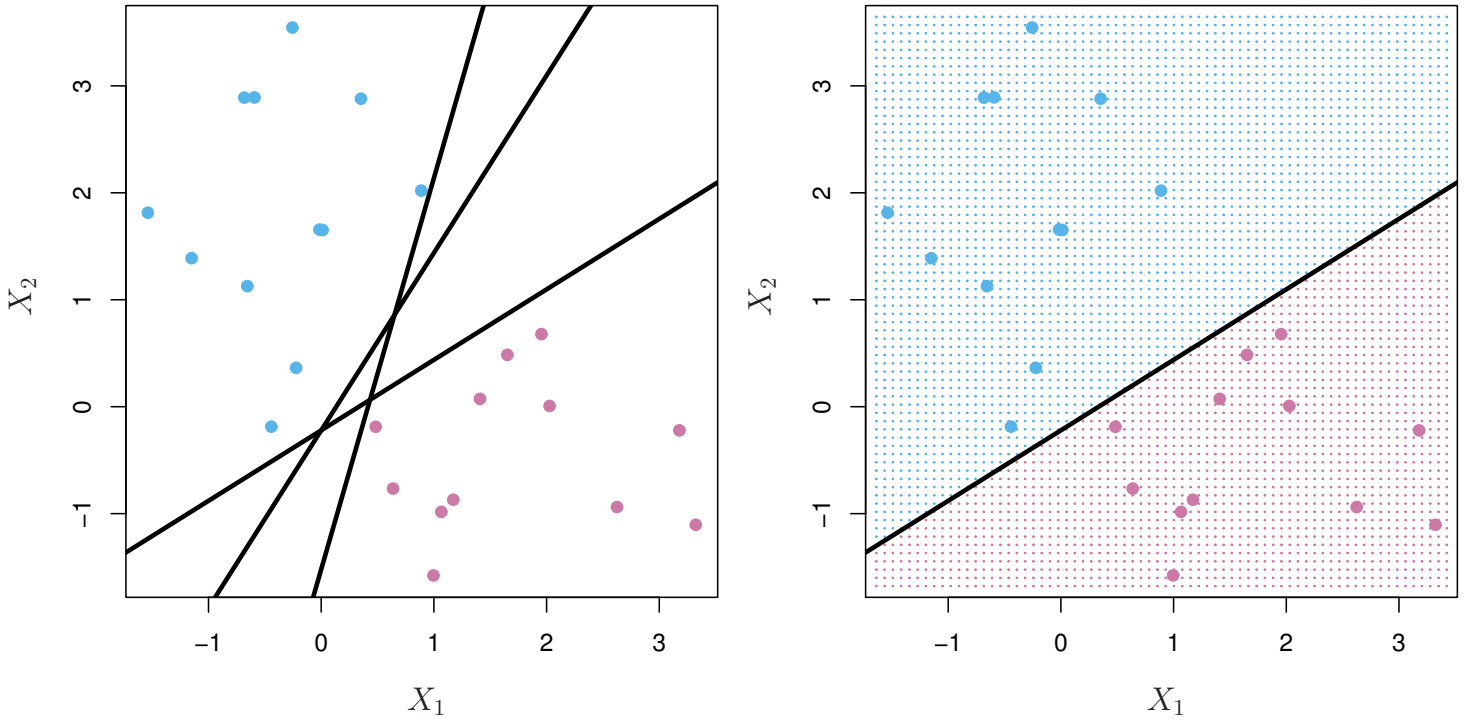


Application statistique : classification linéaire

— Étant donné un ensemble de points $v_1, ..., v_m$ avec les etiquettes $s_i \in \{-1, 1\}$

— trouver un hyperplan $\alpha^T x + \beta$ tels que les points avec les etiquettes "+1" et "-1" se trouvent dans les deux demi-espaces différents

$$\alpha^T v_i + \beta > 0 \text{ si } s_i = 1, \quad \alpha^T v_i + \beta < 0 \text{ si } s_i = -1$$



Note : un plan de separation satisfait $s_i(\beta + \alpha^T v_i) > 0, \forall i$.

Remarque : les inégalités sont homogènes en α, β, et donc équivalentes aux inégalités linéaires (en α, β)

$$s_i^T(\alpha^T v_i + \beta) \geq 1 \ \forall i.$$

Dans le cas de points "non séparables," on peut chercher un classifieur qui minimise le coût

$$\min \left\{ \sum_{i=1}^{m} \max\{0, 1 - s_i(\alpha^T v_i + \beta)\} \right\}$$

— la quantité

$$h_i = \max\{0, 1 - s_i(\alpha^T v_i + \beta)\}$$

(hinge loss) peut être vue comme une pénalité pour la mauvaise classification

— $\sum h_i$ est une borne supérieure pour le nombre de points "mal classés."

- *PL équivalent* en variables $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$ et variable auxiliaire $u \in \mathbb{R}^m$:

$$\min \left\{ \sum_{i=1}^{m} u_i : \begin{array}{r} 1 - s_i(v_i^T \alpha + \beta) \leq u_i \; i = 1, ..., m \\ 0 \leq u_i, \; i = 1, ..., m \end{array} \right\}$$

- *PL en notation matricielle :* $\min\{c^T x : \; Ax \leq b\}$, avec

$$x = \begin{bmatrix} \alpha \\ \beta \\ u \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad A = \begin{bmatrix} -s_1 v_1^T & -s_1 & -1 & 0 & ... & 0 \\ -s_2 v_2^T & -s_2 & 0 & -1 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ -s_m v_m^T & -s_m & 0 & 0 & ... & -1 \\ 0 & 0 & -1 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & ... & -1 \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ \vdots \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Logiciels

Les logiciels d'optimisation peuvent être classés en 2 groupes :

• *"Solveurs" – moteurs d'optimisation*, commerciaux (`Mosek, CPLEX, Gurobi, ...`) ou libres (`GLPK, LP_Solve, SDPT3, ...`) – ce sont eux qui, proprement dit, résolvent des problèmes d'optimisation.

Un solveur accepte en entrée un problème d'optimisation dans un format special (propre à chaque solveur), pour rendre à la sortie une *solution,* si le problème en question est soluble, ou décide que le problème n'est pas soluble et produit un *certificat* pour justifier cette décision.

Certains moteurs d'optimisation sont interfacé avec `R`. Dans ce cours nous allons utiliser `RMosek` – l'interface `R` du moteur `Mosek` (logiciel commercial, disponible gratuitement pour les universitaires) https ://www.mosek.com/

• *Outils de modélisation – "modeleurs"*, dont le but est de simplifier la formulation et analyse d'un problème d'optimisation. En ce qui concerne les problèmes équivalents aux programmes OL, ces outils peuvent

— accepter le problème dans une forme standard simplifiée
(e.g., avec les max, $\|\cdot\|_1$, $\|\cdot\|_\infty$, etc)

— reconnaître les problèmes qui peuvent être convertis en OL

— transformer le problème dans le format accepté par le moteur d'optimisation utilisé

Les outils de modélisations sont disponibles sous différentes formes et en différents langages de programmation :

— `AMPL, GAMS` (outils autonomes)

— `CVX, YALMIP (MATLAB)`

— `CVXPY, CVXOpt, Pyomo (Python)`

— `CVXR (R)`

CVXR exemple

$$\min \{\|x\|_1 : \ \|Ax - b\|_\infty \leq r, \ \|x\|_\infty \leq s\} .$$

```
> library(mvtnorm)
> n=500; p=50; sig=0.01;epsn=0.01;s=2;
> r=qnorm(1-epsn/n)*sig;
> S=0.5^toeplitz(1:p)
> A=rmvnorm(n, sigma = S)
> b=apply(A[,1:5], 1, sum) + sig*rnorm(n)
# CVXR proprement dit
> library(CVXR)
> x=Variable(p)
> constraints=list(p_norm(x, Inf)<=s, p_norm(b-A%*%x, Inf)<=r)
> prob=Problem(Minimize(p_norm(x,1)), constraints)
> result=solve(prob)
> x=result$getValue(x)
> plot(x)
```

RMosek Moteur "commercial" (accès gratuit pour les etudiants)

- Format standard `RMosek` :
  - — variable $x \in \mathbb{R}^n$,
  - — matrice de contraintes $A \in \mathbb{R}^{m \times n}$
  - — objectif $c \in \mathbb{R}^n$,
  - — bornes inférieure ℓ_c et supérieure u_c de contraintes
  - — bornes inférieure ℓ_x et supérieure u_x de x

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad \ell_c \leq \quad & Ax \quad \leq u_c, \\ \ell_x \leq \quad & x \quad \leq u_x \end{aligned}$$

La page du projet : https ://cran.r-project.org/web/packages/Rmosek/index.html

Documentation `RMosek` : https ://docs.mosek.com/9.0/rmosek/index.html

Utilisation de `RMosek`

• Problème de régime McDonald's optimal :

"En mots :" minimizer *les calories*

sous contraintes *des nutriments*

$$\min_x \left\{ \sum_{j=1}^{n} c_j x_j : \begin{array}{l} \sum_{j=1}^{n} p_{ij} x_j \geq \underline{b}_i,\ i = 1, \ldots m, \\ \sum_{j=1}^{n} p_{ij} x_j \leq \bar{b}_i,\ \ i = 1, \ldots m, \\ x_j \geq 0,\ 1 \leq j \leq n \end{array} \right\}$$

Ou encore :

$$\min\{c^T x :\ \underline{b} \leq Ax \leq \bar{b},\ x \geq 0\},$$

ou $A \in \mathbb{R}^{m \times n}$ est la matrice avec les éléments $[A]_{ij} = p_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq n$.

```
> foods=read.table("food.dat", header=T) #Lecture des donnees
> foods[1:4,]
                                    Food Cal CalFat Fat SatFat Chol Sodium Carbo Protein Vi
1  1%_Low_Fat_Milk_Jug 1_carton_(236_ml) 100     20   2      1   10    125    12       8
2             Apple_Slices 1.2_oz_(34_g)  15      0   0      0    0      0     4       0
3        BBQ_Ranch_Burger 4.1_oz_(116_g) 350    140  16      6   45    680    37      16
4 Bacon,_Egg_&_Cheese_Bagel 7_oz_(199_g) 630    290  32     11  275   1490    57      30
>
> fnames=foods[,1]
> nutr.norm=read.table("nutr_ideal.dat", header=T)
> names(nutr.norm)
 [1] "Cal"     "CalFat"  "Fat"     "SatFat"  "Chol"    "Sodium"  "Carbo"   "Protein" "VitA"
[11] "Calcium" "Iron"
>
> diet1 = list()                          # creation du probleme
> diet1$sense = "min"                     # definir le sens d'optimisation
> diet1$c = as.matrix(foods[,2])          # definir l'objectif: la valeur calorique
> A = t(as.matrix(foods[,3:13]))          # definir la matrice des contraintes
> diet1$A = Matrix(A,sparse=TRUE)
> b = as.matrix(nutr.norm[2:12])          # definir les bornes admissibles
> blc = 0.8*b; buc=1.2*b;                 # pour les nutriments
> diet1$bc = rbind(blc, buc);
> blx = rep(0,305); bux <- rep(Inf,305); # contraintes de non-negativite
> diet1$bx = rbind(blx, bux);
> r = mosek(diet1)

Computer
  Platform             : Windows/64-X86
```

```
  Cores                  : 1

Problem
  Name                   :
  Objective sense        : min
  Type                   : LO (linear optimization problem)
  Constraints            : 11
  Cones                  : 0
  Scalar variables       : 305
  Matrix variables       : 0
  Integer variables      : 0

Optimizer started.
Interior-point optimizer started.
Presolve started.
...



Factor     - nonzeros before factor : 66              after factor           : 66
Factor     - dense dim.             : 0               flops                  : 3.18e+004
ITE PFEAS    DFEAS    GFEAS    PRSTATUS   POBJ              DOBJ              MU       TIME
0   2.4e+002 6.6e+000 6.5e+002 2.26e+000  1.372955304e+005  0.000000000e+000  2.3e+000 0.00
1   2.2e+002 8.2e+000 6.6e+002 0.00e+000  1.418094583e+005  1.664025593e+003  7.4e+000 0.00
2   1.3e+001 4.6e-001 3.7e+001 -8.43e-001 5.591419263e+004  2.518824208e+003  4.2e-001 0.00
3   4.2e+000 1.6e-001 1.3e+001 9.61e-001  1.568108462e+004  2.162233225e+003  1.4e-001 0.00
4   2.4e+000 8.9e-002 7.2e+000 2.87e+000  6.625715907e+003  2.275951097e+003  8.1e-002 0.00
```

```
5   1.4e+000 5.2e-002 4.2e+000 2.90e+000  3.387566741e+003  2.192592773e+003  4.7e-002 0.00
6   8.4e-001 3.1e-002 2.5e+000 2.10e+000  2.434783666e+003  1.883464080e+003  2.8e-002 0.00
7   4.5e-001 1.7e-002 1.3e+000 1.43e+000  1.899310800e+003  1.640642415e+003  1.5e-002 0.00
8   2.1e-001 7.8e-003 6.3e-001 1.15e+000  1.573381052e+003  1.455345733e+003  7.0e-003 0.00
9   2.7e-002 9.8e-004 7.9e-002 1.23e+000  1.366259235e+003  1.352122956e+003  8.9e-004 0.00
10  5.5e-004 2.0e-005 1.6e-003 9.99e-001  1.335288862e+003  1.334995770e+003  1.8e-005 0.00
11  8.6e-007 3.2e-008 2.5e-006 1.00e+000  1.334857456e+003  1.334857002e+003  2.8e-008 0.00
12  8.6e-011 3.2e-012 2.5e-010 1.00e+000  1.334856193e+003  1.334856193e+003  2.8e-012 0.00

Basis identification started.
...
Optimizer terminated. Time: 0.00

Interior-point solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: 1.3348561927e+003   Viol.  con: 7e-008   var: 0e+000
  Dual.    obj: 1.3348561929e+003   Viol.  con: 4e-010   var: 6e-010

...
```

```
> x=r$sol$itr$xx                  # extraire la solution x
> t(diet1$c)%*%r$sol$itr$xx       #valeur optimale c^Tx --la valeur calorique du regime
         [,1]
[1,] 1334.856
>
...
> mydiet                                  # impression du contenu de regime
[1] "Chocolate_Chip_Cookie 1_cookie_(33_g):     0.699908803471238"
[2] "EQUAL_0_Calorie_Sweetener 1_pkg_(1.0_g):     84.2645722693934"
[3] "Fat_Free_Chocolate_Milk_Jug 1_carton_(236_ml):     0.898566545455502"
[4] "Hamburger 3.5_oz_(100_g):     2.14750840210163"
[5] "Sausage_Burrito 3.9_oz_(111_g):     1.35453792177156"
[6] "Side_Salad 3.1_oz_(87_g):     1.98962488832494"
>
```

On peut ajouter des contraintes pour rendre le menu “mangeable,” par exemple

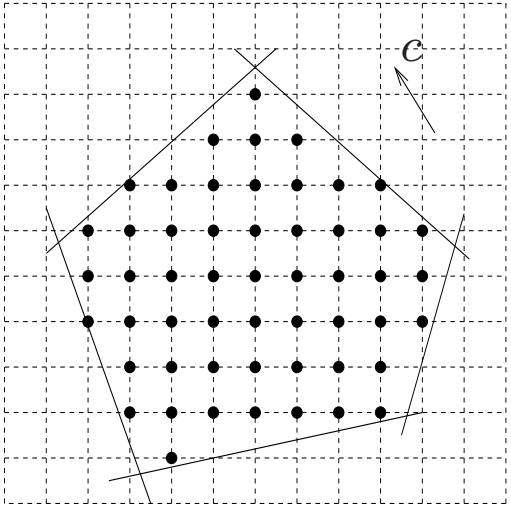
```
> bux=rep(10,n); % quantite de tous les produits ne depasse pas 10
```

et ainsi de suite...

Programmation en nombres entiers

- Définitions
  - — *(PLE) – programme linéaire en nombres entiers* (Integer Linear program, ILP)

$$\min_x \{c^T x : \ Ax \leq b, \, x \in \mathbf{Z}^n\}$$



  - — *(PLEM) – programme linéaire d'entiers mixte* (Mixed Integer Linear program) : certaines variables (mais pas toutes) sont des entiers
  - — *(PLB) – programme lineaire booléen* (0-1, Boolean Linear program) : variables à valeurs dans $\{0, 1\}$

Exemples

- Probleme d'emplacement
  - n emplacement possibles pour les installation industrielles avec coût d'emplacement c_j
  - m clients
  - d_{ij} le coût de service du client i de l'emplacement j

Variables $y_j,\ x_{ij}$:

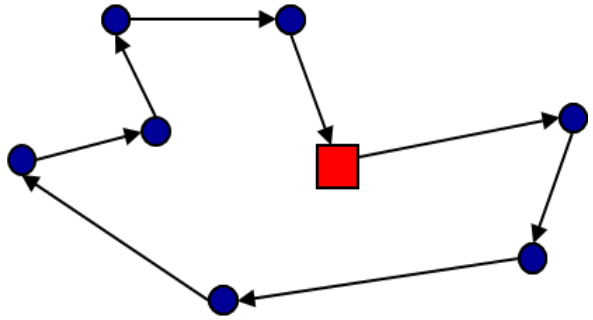
- $y_j = 1$ si l'emplacement j est sélectionné et 0 sinon
- $x_{ij} = 1$ si l'emplacement j sert le client i et 0 sinon

Formulation booléenne :

$$\min \left\{ \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} : \begin{array}{l} \sum_{j=1}^{n} x_{ij} = 1, i = 1, ..., m \\ x_{ij} \leq y_j,\ i = 1, ..., m,\ j = 1, ..., n \\ x_{ij}, y_j \in \{0, 1\} \end{array} \right\}$$

- L'exemple le plus connu d'un problème difficile – *problème du voyageur de commerce* :

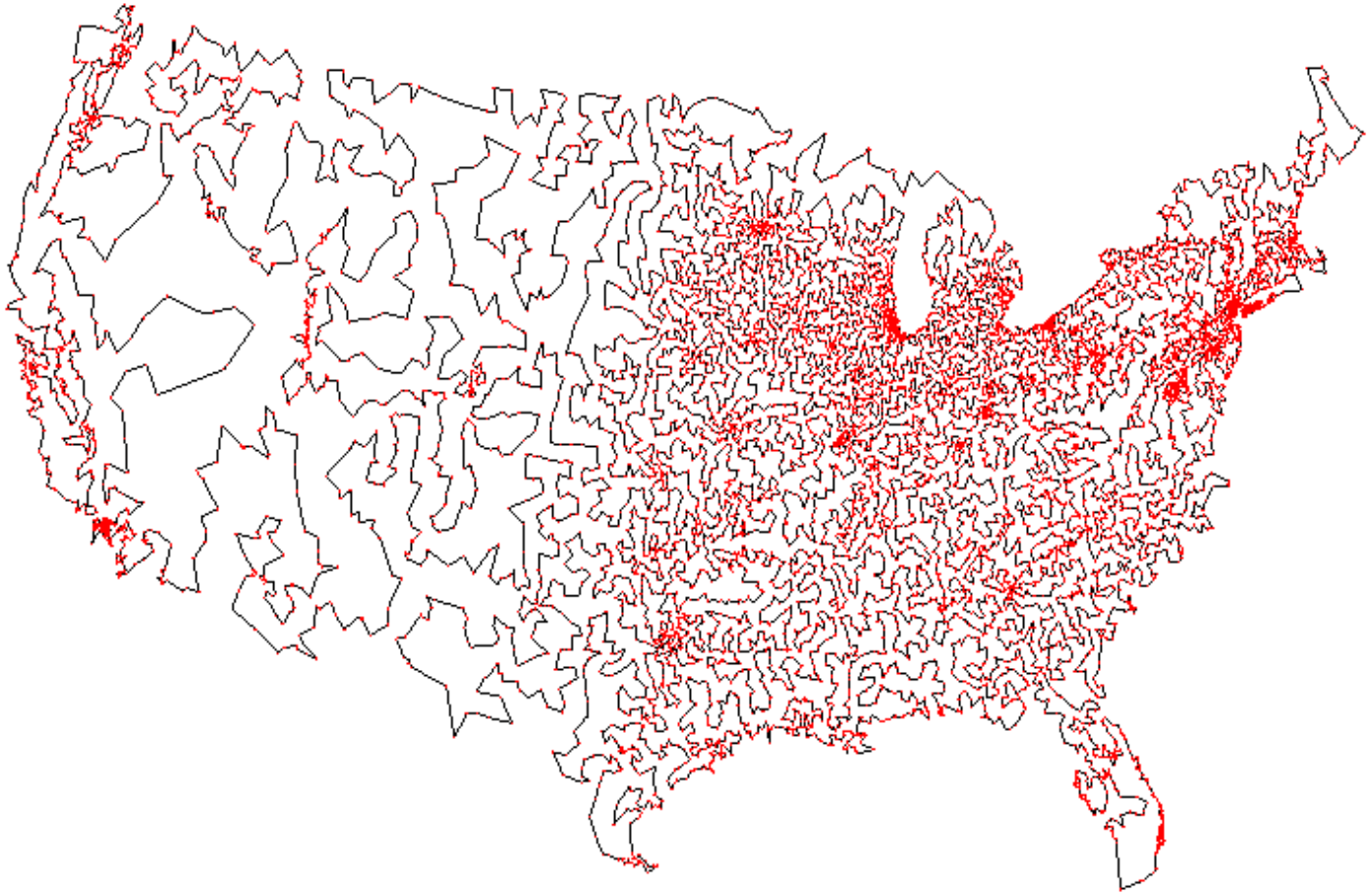
– *étant donnés n sites, déterminer l'ordre dans lequel les visiter pour minimiser la distance totale parcourue*



Pour n cites il y a $n!$ parcours possibles (!)

n	$n!$	n^4	2^n
7	5040	2401	128
8	40320	4096	256
...	...	...	
20	2.4329e+18	160 000	1048576
...	...	...	
40	8.1592e+47	2 560 000	1.0995e+12

13509 villes aux États-Unis



(Applegate, Bixby, Chvatal & Cook, 1998), algorithme spécialisé
Voir article *NY Times* "Le probleme du politicien voyageur" pour Iowa (99 contés)

https ://campaignstops.blogs.nytimes.com/2011/12/21/the-problem-of-the-traveling-politician/

Relaxation par PL

Probleme general de programmation en nombres entiers :

$$\min_x \{c^T x : \ Ax \leq b, \ x \text{ est un entier}\}$$

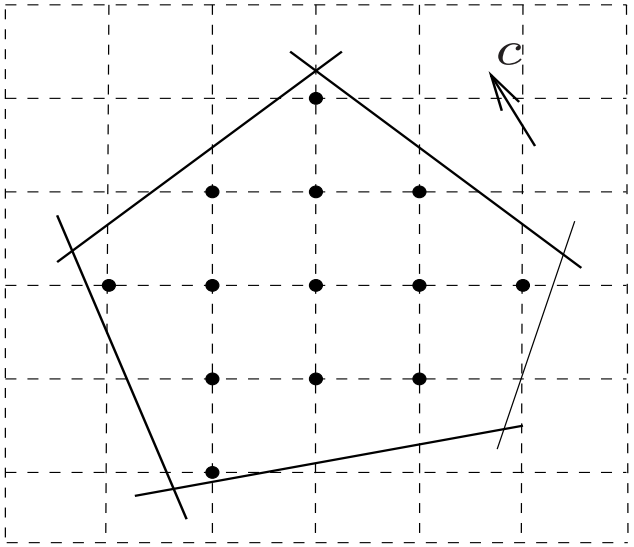
$\Rightarrow$ (relaxation)

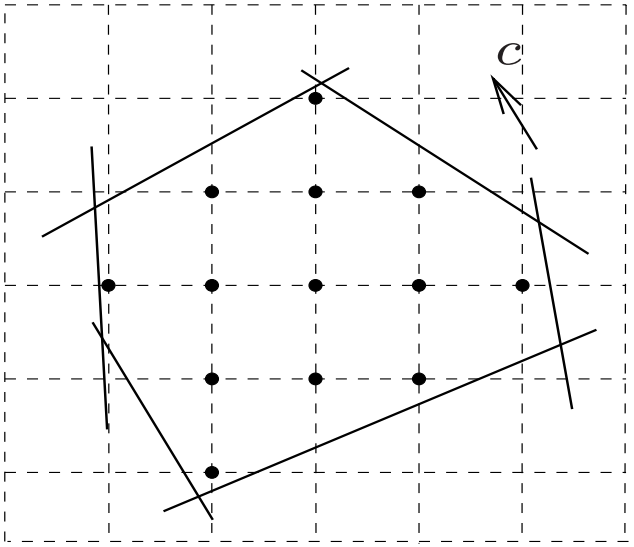
$$\min_x \{c^T x : \ Ax \leq b, \ x \ \sout{\text{est un entier}}\}$$

— Permet d'obtenir une borne intérieure sur la valeur optimale du PLE

— Si la solution est en nombres entiers, c'est aussi la solution du PLE

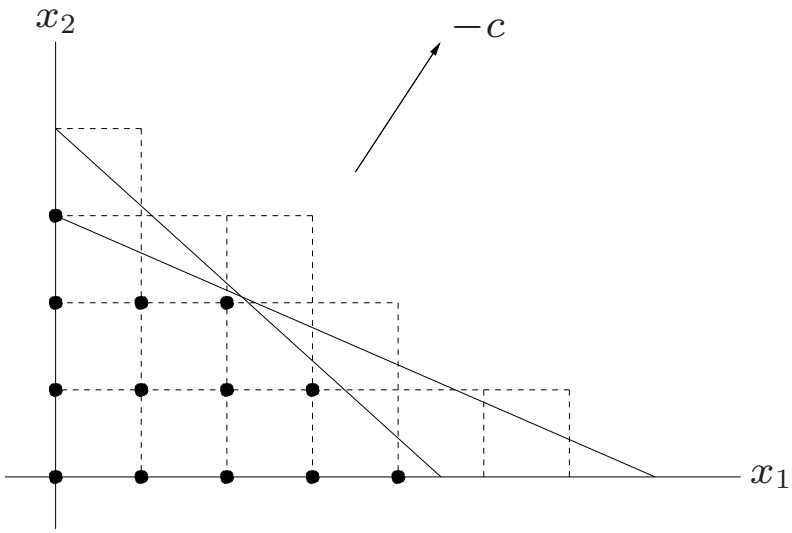
— *Attention :* on peut avoir des formulations relaxées différentes pour le même PLE





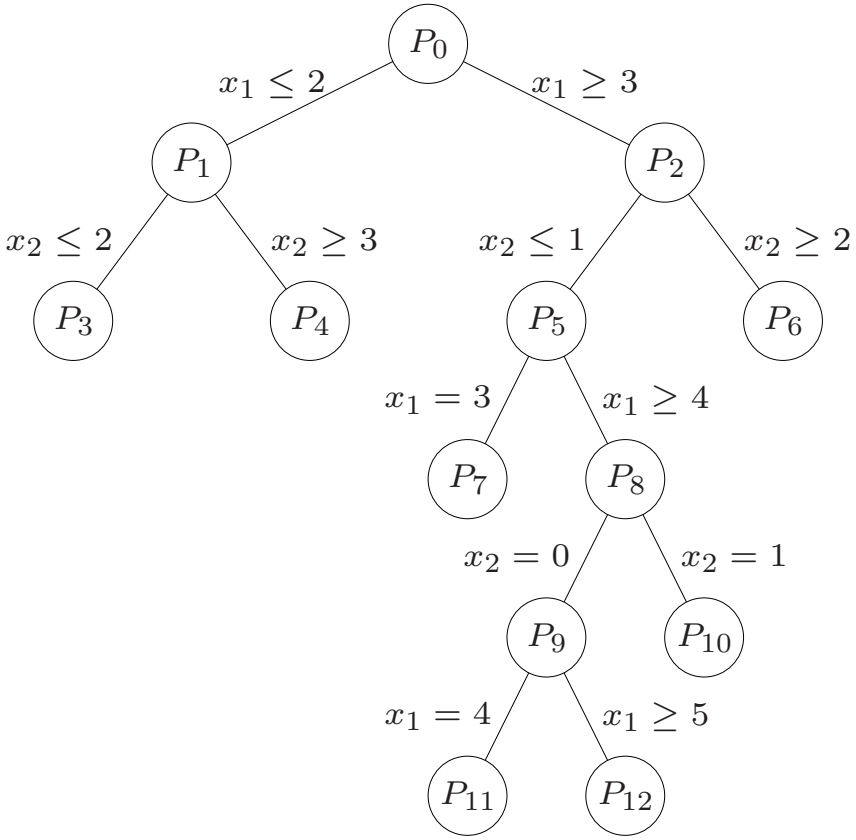
Exemple

$$\min_x \left\{ -2x_1 - 3x_2 : \begin{array}{l} \underbrace{\begin{array}{l} \frac{2}{9}x_1 + \frac{1}{4}x_2 \leq 1 \\ \frac{1}{7}x_1 + \frac{1}{3}x_2 \leq 1 \end{array}}_{x \in X} \\ x_1, x_2 \in \mathbf{Z}_+ \end{array} \right\}$$



Optimal solution : [2; 2]

Methode de séparation et d'évaluation progressive (Brunch and Bound)



	x_*	Opt
P_0	[2.17 ; 2.07]	-10.56
P_1	[2.00 ; 2.14]	-10.43
P_2	[3.00 ; 1.33]	-10.00
P_3	[2.00 ; 2.00]	-10.00
P_4	[0.00 ; 3.00]	-9.00
P_5	[3.38 ; 1.00]	-9.75
P_6		$+\infty$
P_7	[3.00 ; 1.00]	-9.00
P_8	[4.00 ; 0.44]	-9.33
P_9	[4.50 ; 0.00]	-9.00
P_{10}		$+\infty$
P_{11}	[4.00 ; 0.00]	-8.00
P_{12}		$+\infty$

Séparation et évaluation

$\Rightarrow P_2$: min $c^T x$ sous contrainte $x \in X$ et $x_1 \geq 3$

valeur optimale ≥ -10.00

$\Rightarrow P_3$: min $c^T x$ sous contrainte $x \in X$, $x_1 \leq 2$, et $x_2 \leq 2$

solution $x = [2; 2]$, valeur optimale $\mathrm{Opt} = -10$

...

$\Rightarrow P_6$: min $c^T x$ sous contrainte $x \in X$, $x_1 \geq 3$, $x_2 \geq 2$

problème irréalisable

...

Apres avoir résolu les relaxations pour P_0, P_1, P_2, P_3, P_4 on peut déduire que $[2; 2]$ est la solution optimale du PLE

Utilisation du solveur d'entiers mixte (MIP) de Mosek

- il suffit de déclarer pour `RMosek` des variables entières :
  - le vecteur `intsub` doit contenir les indices des variables entières
  - variables *booléennes*, $x \in \{0, 1\}$, doivent être déclarer comme entières et satisfaisant la contrainte $0 \leq x \leq 1$

Par exemple, dans le problème de *regime McDonald's*, pour convertir les variables en entiers, il suffit de faire

```
> diet4=diet3
> diet4$intsub=c(1:305)
> r = mosek(diet4)

Computer
  Platform                : Windows/64-X86
  Cores                   : 1

Problem
  Name                    :
  Objective sense         : min
  Type                    : LO (linear optimization problem)
  Constraints             : 12
```

```
  Cones                  : 0
  Scalar variables       : 305
  Matrix variables       : 0
  Integer variables      : 305

Optimizer started.
Mixed integer optimizer started.
Optimizer  - threads                : 1
0        1        0        NA                   1.5106771305e+003    NA           0.0
0        2        0        NA                   1.5222410122e+003    NA           0.0
...
405      366      10       1.5950000000e+003    1.5800000000e+003    0.94         0.2
415      371      0        1.5950000000e+003    1.5950000000e+003    0.00e+000    0.2
An optimal solution satisfying the absolute gap tolerance of 0.00e+000 has been located.
The absolute gap is 0.00e+000.


Objective of best integer solution : 1.595000000000e+003
Best objective bound               : 1.595000000000e+003
Construct solution objective       : Not employed
Construct solution # roundings     : 0
User objective cut value           : 0
Number of cuts generated           : 5
Number of branches                 : 415
Number of relaxations solved       : 371
Number of interior point iterations: 15
Number of simplex iterations       : 999
Time spend presolving the root     : 0.00
```

```
Time spend in the heuristic        : 0.00
Time spend in the sub optimizers   : 0.00
  Time spend optimizing the root   : 0.02
Mixed integer optimizer terminated. Time: 0.19

Optimizer terminated. Time: 0.19

Integer solution solution summary
  Problem status  : PRIMAL_FEASIBLE
  Solution status : INTEGER_OPTIMAL
  Primal.  obj: 1.5950000000e+003   Viol.  con: 0e+000   var: 0e+000   itg: 0e+000
Optimizer summary
...
    Mixed integer           - relaxations: 371       time: 0.19

...

> t(diet4$c)%*%r$sol$int$xx
     [,1]
[1,] 1595
```

```
> mydiet
 [1] "Chocolate_Chip_Cookie 1_cookie_(33_g):     4"
 [2] "Coffee_(Large) 16_fl_oz_cup:     10"
 [3] "Coffee_(Medium) 16_fl_oz_cup:     10"
 [4] "Coffee_(Small) 12_fl_oz_cup:     10"
 [5] "Diet_Coke_(Medium) 21_fl_oz_cup:     10"
 [6] "EQUAL_0_Calorie_Sweetener 1_pkg_(1.0_g):     10"
 [7] "Egg_McMuffin 4.8_oz_(135_g):     1"
 [8] "Fat_Free_Chocolate_Milk_Jug 1_carton_(236_ml):     1"
 [9] "Hamburger 3.5_oz_(100_g):     1"
[10] "Newman's_Own_Low_Fat_Balsamic_Vinaigrette 1.5_fl_oz_(44_ml):     1"
[11] "SPLENDA_No_Calorie_Sweetener 1_pkg_(1.0_g):     10"
[12] "Side_Salad 3.1_oz_(87_g):     2"
[13] "Strawberry_Banana_Smoothie_(Small) 12_fl_oz_cup:     1"
```