CAHIER DES CHARGES

TER 2008-2009 MASTER 1 INFORMATIQUE option CASAR

Sujet du TER : Programmation en 2D et en 3D

- ✓ Visualisation en 2D et 3D de la succession des étapes issues de la résolution d'un programme linéaire par la méthode du simplexe

  ⇨ Objectif : Constater le déplacement sur l'enveloppe convexe des solutions intermédiaires de l'algorithme.

MEMBRES DU GROUPE :

- *BEN YAHIA Zakaria*
- *GALINDO Benjamin*
- *LACOMBE Jonathan*
- *LAGHA Nazim*

Sommaire

I. Introduction aux problèmes de programmation linéaire

Afin d'illustrer ce qu'est la « *Programmation linéaire* », commençons par un petit exemple simple. Celui-ci nous permettra en outre d'introduire certaines propriétés des problèmes relevant de ce domaine, propriétés qui seront ensuite exploitées pour fonder l'algorithme du simplexe.

Une usine fabrique deux sortes de produits, p_1 et p_2, à l'aide de deux machines m_1 et m_2. Chaque unité de produit en cours de fabrication doit passer successivement sur les deux machines dans un ordre indifférent et pendant les temps suivants (en minutes) :

	p_1	p_2
m_1	30	20
m_2	40	10

De plus, la machine m_1 est disponible 6000 min/mois et la machine m_2 est disponible 4000 min/mois. Le profit réalisé sur une unité du produit p_1 est de 400 €. Le profit réalisé sur une unité du produit p_2 est de 200 €.

On souhaite trouver le plan de fabrication mensuel qui maximise le profit.

Pour cela, appelons x (respectivement y) le nombre d'unités du produit p_1 (respectivement p_2) à fabriquer mensuellement ; on voit que ce problème peut s'exprimer sous la forme :

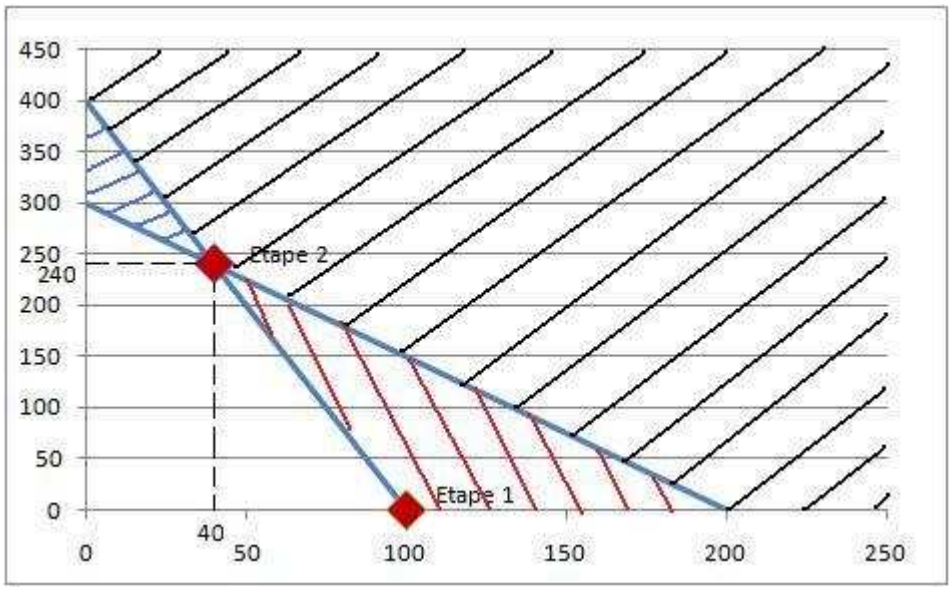
$$\textbf{Maximiser } z = 400\,x + 200\,y$$

Sous les contraintes :

$$\begin{cases} 30x + 20y \le 6000 \\ 40x + 10y \le 4000 \\ x \ge 0, y \ge 0 \end{cases}$$

II. Méthode de Résolution graphique

Le problème étant en deux variables, il admet une solution graphique facile á mettre en œuvre :



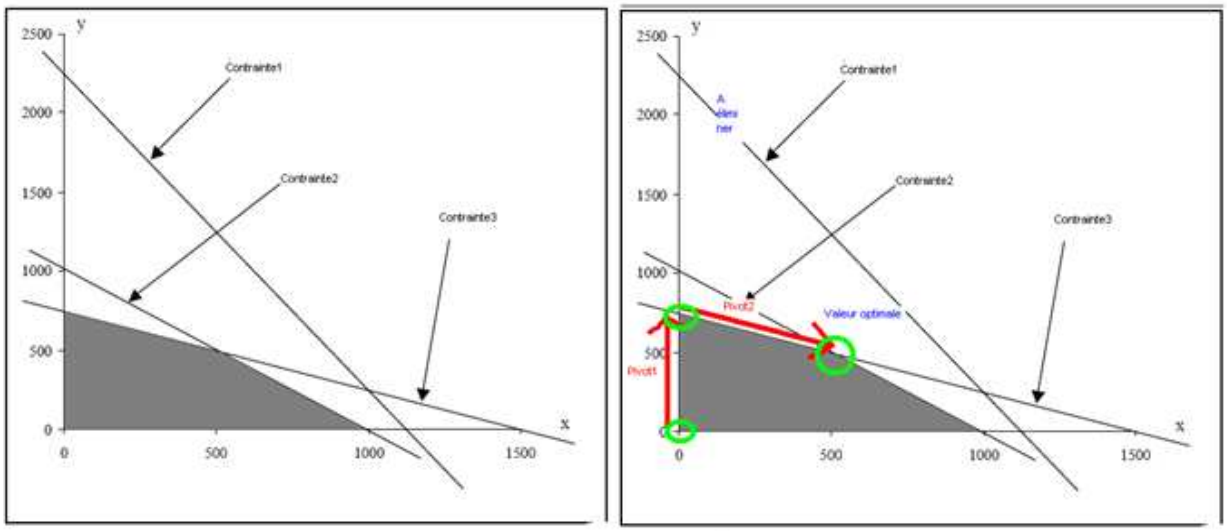
Les points (*x*, *y*) qui satisfont les contraintes appartiennent au quadrilatère *ABCO*. La famille des droites D_y = {(*x*, y) / 400 *x* + 200 *y* = λ} est une famille de droites parallèles. Parmi celles de ces droites qui ont une intersection non vide avec le quadrilatère, c'est celle qui passe par *B* qui correspond á la plus grande valeur de λ : elle rencontre le quadrilatère des contraintes au point de coordonnées (40, 240). La solution optimale de notre problème est donc *x* = 40, *y* = 240. (*z* = 64000).

Cet exercice est une illustration de la « programmation linéaire ». Plus généralement, un problème de programmation linéaire est un problème qui peut se formuler comme suit :

- Maximiser une forme linéaire de n variables $x_1, x_2 \ldots\ldots\ldots x_{n-1}, x_n$ les variables etant soumises :

  - à *m* contraintes linéaires : $\sum_{j=1}^{n} a_{ij} x_j \leq b_i, avec : i = 1,...,m$
  - aux *n* contraintes de non-négativité : $x_j \geq 0, pour : j = 1,...,n$

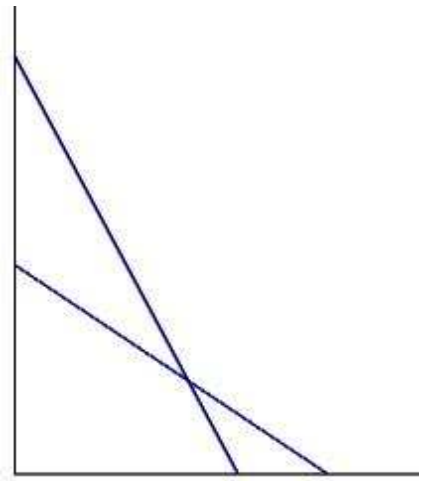
A. Cas général et Implémentation



Le graphique à gauche ci-dessus représente la forme graphique d'un problème de programmation linéaire, résolu par la méthode du simplexe (avec en gris l'enveloppe convexe). Celui de droite illustre la correspondance avec la les étapes intermédiaires de la résolution par cette méthode. Ainsi, pour chaque pivot, les coordonnées du point représentent l'évolution de la solution. Les traits rouges représentent les différents pivots, les ronds verts la valeur des solutions intermédiaires, jusqu'au résultat définitif obtenu lors de la convergence.

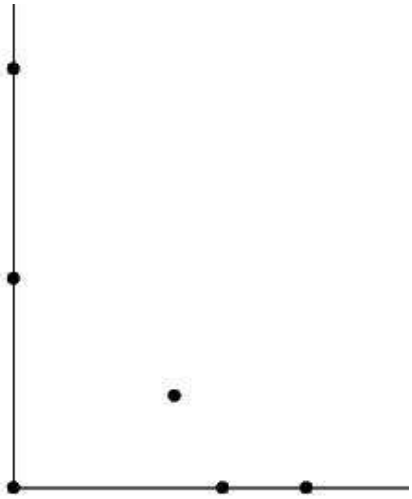
Pour l'implémentation de l'enveloppe convexe, on suit les étapes suivantes :

ETAPE I : On récupère les contraintes sous forme de droites

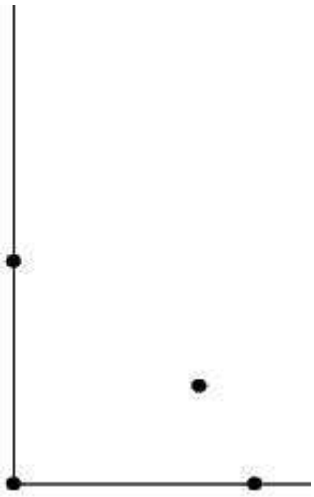


ETAPE II : On calcule les intersections entre toutes les droites

- ⇨ Système de 2 équations à 2 inconnues pour la 2D
- ⇨ Système de 3 équations à 3 inconnues pour la 3D

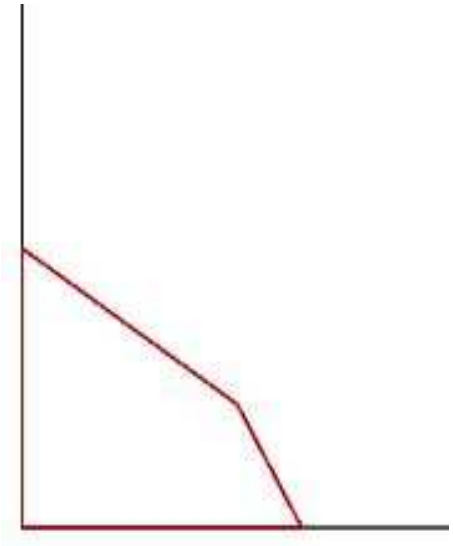


ETAPE III : On supprime les points qui ne vérifient pas toutes les contraintes



ETAPE VI : On dessine l’enveloppe convexe

- ⇨ On utilise les coordonnées polaires pour la 2D
- ⇨ On utilise les coordonnées cylindriques pour la 3D



Il ne reste plus qu'à faire correspondre avec les résultats intermédiaires de la résolution matricielle et afficher les points successifs avant la convergence.

- ⇨ Le langage choisi pour la 2D est QT.
- ⇨ Le langage choisi pour la 3D est OpenGL.

III. Méthode de Résolution Matricielle

A. Principe de l'algorithme

Nous allons maintenant résumer l'algorithme du simplexe (méthode Matricielle) pour la résolution d'un problème de programmation linéaire.

Considérons le problème:

Maximiser c. X sous les contraintes $A.X = b$ et $X \geq 0$

Supposons connue une base réalisable. On appelle c_B et B les parties de c et de A correspondant à cette base réalisable X_B. Il nous faut initialiser X_B^* qui vaut $B^{-1}b$.

Une étape se déroule alors comme suit :

1. Résoudre le système $yB = c_B$.

2. Choisir une colonne entrante s'il en existe une, c'est-à-dire une colonne a_j de A non dans B, telle que $y.a_j$ soit plus petit que c_j (s'il s'agit de minimiser, on chercherait une colonne a_j telle que $y.a_j$ soit plus grand que c_j). S'il n'existe pas de colonne entrante, la base est optimale et l'algorithme s'arrête.

3. Résoudre le système $Bd = a_j$.

4. Trouver le plus grand t tel que $X_B^* - td \geq 0$. S'il n'existe pas de telle valeur de t, le problème est non borné. Sinon, lorsque t atteint cette valeur maximum, une au moins des composantes de $X_B^* - td$ vaut 0 et la variable correspondante peut quitter la base, on dira d'elle sortante.

5. Remplacer dans X_B la variable sortante par la variable entrante. Mettre à t dans X_B^* la valeur de la variable entrante, et pour les autres variables remplacer leur valeur dans $X_B^* - td$. Dans B, remplacer la colonne sortante, c'est-à-dire la colonne associée à la variable sortante, par la colonne entrante.

REMARQUE.

Lorsque le problème est donné sous forme standard, si la solution nulle est réalisable, on peut initialiser l'algorithme avec $B=I$ et $X_B^*=b$, ceci correspond au choix de l'ensemble des variables d'écart pour constituer la première base.

B. Application à un cas 2D

Nous allons appliquer l'algorithme au cas énoncé précédemment, et dont la solution graphique á été déterminée. (Parcours du polyèdre des contraintes de sommet en sommet).

On a :

Maximiser z = 400 *x* + 200 *y*

Sous les contraintes : $\begin{cases} 30x + 20y \le 6000 \\ 40x + 10y \le 4000 \\ x \ge 0, y \ge 0 \end{cases}$

Le problème s'écrit alors :

$$\begin{cases} 30x_1 + 20x_2 + x_3 = 6000 \\ 40x_1 + 10x_2 + x_4 = 4000 \\ x_1, x_2, x_3, x_4 \ge 0 \end{cases}$$

On a ici :

$$A = \begin{pmatrix} 30 & 20 & 1 & 0 \\ 40 & 10 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 6000 \\ 4000 \end{pmatrix}, \quad c = \begin{pmatrix} 400 & 200 & 0 & 0 \end{pmatrix}.$$

On choisit comme base de départ les variables d'écart et donc :

$$X_B = \begin{pmatrix} x_3 \\ x_4 \end{pmatrix}, \quad X_B^* \begin{pmatrix} 6000 \\ 4000 \end{pmatrix}$$

D'où :

$$X_N = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A_N = \begin{pmatrix} 30 & 20 \\ 40 & 10 \end{pmatrix}, \quad c_B = \begin{pmatrix} 0 & 0 \end{pmatrix}, \quad c_N = \begin{pmatrix} 400 & 200 \end{pmatrix}.$$

ETAPE I : Première itération de l'algorithme

I-1 : On résout : $y.B = c_B$ avec $y = \begin{pmatrix} y_1 & y_2 \end{pmatrix} \in \Re^2$

$$y.B = c_B \Rightarrow \begin{pmatrix} y_1 & y_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

$$\Rightarrow \begin{cases} y_1 = 0 \\ y_2 = 0 \end{cases}$$

$$\Rightarrow \quad y = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

I-2 : Calcul de la variable entrante : Soient : a_n la nième colonne de la matrice A_N et : c_n la nième composante de c_N

$$ya_1 = \begin{pmatrix} 0 & 0 \end{pmatrix} \begin{pmatrix} 30 \\ 40 \end{pmatrix} = 0 < c_1 = 400$$

La première variable (notée x_1) est donc candidate à entrer car $ya_1 < c_1$.

$$ya_2 = \begin{pmatrix} 0 & 0 \end{pmatrix} \begin{pmatrix} 20 \\ 10 \end{pmatrix} = 0 < c_2 = 200$$

La deuxième variable (notée x_2) est donc aussi candidate à entrer car $ya_2 < c_2$.

On a le choix. Soit donc x_1 la variable entrante.

I-3 : On résout : $Bd = a_1$ avec $d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \in \Re^2$

$$Bd = a_1 \quad \Rightarrow \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = a_1 \quad \Rightarrow \quad \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 30 \\ 40 \end{pmatrix} = a_1$$

I-4 : Calcul de la variable de sortie en déterminant t *maximum* tel que : $X_B^* - td \geq 0$

$$X_B^* - td \geq 0 \quad => \quad \begin{pmatrix} 6000 \\ 4000 \end{pmatrix} - t \begin{pmatrix} 30 \\ 40 \end{pmatrix} \geq 0 \text{ , avec : } t \in \Re$$

$$=> \quad \begin{cases} 6000 - 30t \geq 0 \\ 4000 - 40t \geq 0 \end{cases}$$

$$=> \quad t = 100$$

La plus grande valeur possible de t est 100. Donner cette valeur à la variable x_1 annule la deuxième variable de base, c'est-à-dire x_4. Cette dernière est donc la variable sortante. **On a donc augmenté x_1 de 100.**

I – 5 : Actualisation :

$$X_B = \begin{pmatrix} x_3 \\ x_1 \end{pmatrix}, \qquad X_B^* = \begin{pmatrix} 6000 - 300 * 100 \\ 100 \end{pmatrix} = \begin{pmatrix} 3000 \\ 100 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 30 \\ 0 & 40 \end{pmatrix}, \quad A_N = \begin{pmatrix} 0 & 20 \\ 1 & 10 \end{pmatrix}, \quad c_B = \begin{pmatrix} 0 & 400 \end{pmatrix}, \quad c_N = \begin{pmatrix} 0 & 200 \end{pmatrix}$$

ETAPE II : Deuxième itération de l'algorithme

II-1 : On résout : $y.B = c_B$ avec $y = \begin{pmatrix} y_1 & y_2 \end{pmatrix} \in \Re^2$

$$y.B = c_B \quad => \quad \begin{pmatrix} y_1 & y_2 \end{pmatrix} \begin{pmatrix} 1 & 30 \\ 0 & 40 \end{pmatrix} = \begin{pmatrix} 0 & 400 \end{pmatrix}$$

$$=> \quad \begin{cases} y_1 = 0 \\ 30y_1 + 40y_2 = 400 \end{cases}$$

$$=> \quad \begin{cases} y_1 = 0 \\ y_2 = 10 \end{cases}$$

II-2 : Calcul de la variable entrante : Soient : a_n la nième colonne de la matrice A_N et : c_n la nième composante de c_N

$$ya_1 = \begin{pmatrix} 0 & 10 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 10 > 0 = c_1$$

La première variable (notée x_3) n'est donc pas candidate à entrer car $ya_1 > c_1$.

$$ya_2 = \begin{pmatrix} 0 & 10 \end{pmatrix} \begin{pmatrix} 20 \\ 10 \end{pmatrix} = 100 < 200 = c_2$$

La deuxième variable (notée x_2) est par contre candidate à entrer car $ya_2 < c_2$.

On n'a pas le choix cette fois-ci. Seule la variable x_2 est candidate à entrer. Soit donc cette dernière la variable entrante.

II-3 : On résout : $Bd = a_2$ avec $d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \in \Re^2$

$$Bd = a_2 \quad => \quad \begin{pmatrix} 1 & 30 \\ 0 & 40 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 20 \\ 10 \end{pmatrix} \quad => \quad \begin{cases} d_1 + 30d_2 = 20 \\ 40d_2 = 10 \end{cases} \quad => \quad \begin{cases} d_1 = \frac{25}{2} \\ d_2 = \frac{1}{4} \end{cases}$$

II-4 : Calcul de la variable de sortie en déterminant t *maximum* tel que : $X_B^* - td \geq 0$

$$X_B^* - td \geq 0 \quad => \quad \begin{pmatrix} 3000 \\ 100 \end{pmatrix} - t \begin{pmatrix} \frac{25}{2} \\ \frac{1}{4} \end{pmatrix} \geq 0 \quad => \quad \begin{cases} 3000 - t\frac{50}{4} \geq 0 \\ 100 - t\frac{1}{4} \geq 0 \end{cases} \quad => \quad t = 240$$

La plus grande valeur possible de t est 240. Donner cette valeur à la variable x_2 annule la première variable de base, c'est-à-dire x_3. Cette dernière est donc la variable sortante. **On a donc augmenté x_2 de 240.**

II – 5 : Actualisation :

$$X_B = \begin{pmatrix} x_2 \\ x_1 \end{pmatrix}, \qquad X_B^* = \begin{pmatrix} 240 \\ 100 - 240 * \frac{1}{4} \end{pmatrix} = \begin{pmatrix} 240 \\ 40 \end{pmatrix}$$

$$B = \begin{pmatrix} 20 & 30 \\ 10 & 40 \end{pmatrix}, \quad A_N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad c_B = \begin{pmatrix} 200 & 400 \end{pmatrix}, \quad c_N = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

ETAPE III : Troisième itération de l'algorithme

III-1 : On résout : $y.B = c_B$ avec $y = \begin{pmatrix} y_1 & y_2 \end{pmatrix} \in \Re^2$

$$y.B = c_B \Rightarrow \begin{pmatrix} y_1 & y_2 \end{pmatrix} \begin{pmatrix} 20 & 30 \\ 10 & 40 \end{pmatrix} = \begin{pmatrix} 200 & 400 \end{pmatrix}$$

$$\Rightarrow \begin{cases} 20y_1 + 10y_2 = 200 \\ 30y_1 + 40y_2 = 400 \end{cases} \Rightarrow \begin{cases} y_1 = 8 \\ y_2 = 4 \end{cases}$$

III-2 : Calcul de la variable entrante : Soient : a_n la nième colonne de la matrice A_N
et : c_n la nième composante de c_N

$$ya_1 = \begin{pmatrix} 8 & 4 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 4 > 0 = c_1$$

La première variable (notée x_2) n'est donc pas candidate à entrer car $ya_1 > c_1$.

$$ya_2 = \begin{pmatrix} 8 & 4 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 8 > 0 = c_2$$

La deuxième variable (notée x_1) non plus n'est pas candidate à entrer car $ya_2 < c_2$.

Il n'y a aucune variable candidate à entrer, car $ya_1 > c_1$ et $ya_2 < c_2$

La méthode du simplexe a donc convergée. La solution optimale de notre problème est donc $x = 40$, $y = 240$ et $z = 64000$, car $40 * 400 + 200 * 240 = 64000$. Evidement, celle-ci correspond á la solution trouvée á travers la résolution graphique faite dans le deuxième paragraphe.

IV. Forme standard

C'est la *forme standard* d'un problème de programmation linéaire. Des problèmes où il s'agit de minimisation, ou pour lesquels apparaissent des contraintes d'égalités ou d'inégalité large dans l'autre sens (on ne considérera pas d'inégalité stricte), ou encore pour lesquels les variables sont non contraintes ou encore contraintes à être non positives peuvent facilement se mettre sous forme standard, comme le précisent les indications suivantes :

- minimiser une fonction f (linéaire ou non) revient à maximiser $-f$, puisqu'on a la relation : Minimum de f = - Maximiser f ;
- on transforme une inégalité de genre « $\geq$ » en une inégalité du genre « $\leq$ » en la multipliant par -1 ;
- une égalité $\alpha = \beta$ revient aux deux inégalités $\alpha \leq \beta$ et $\alpha \geq \beta$;
- on remplace une variable x contrainte à être négative ou nulle par $-x$;
- on exprime une variable x qui n'a pas de signe imposé par la différence de deux variables positives ou nulles : $x = x^+ - x^-$ avec $x^+ \geq 0$ et $x^- \leq 0$.

On peut alors se demander si la démarche proposée pour l'exemple précédent est susceptible d'être généralisée á la résolution de tous problèmes de programmation linéaire. Puisqu'il est toujours possible d'exprimer un problème de programmation linéaire sous forme standard, considérons un problème de la forme :

$$\text{Maximiser } \sum_{j=1}^{n} c_j x_j \quad \text{avec les contraintes} \quad \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i, avec : i = 1,...,m \\ x_j \geq 0, pour : j = 1,...,n \end{cases}$$

L'ensemble des points de $\mathbb{R}^n$ de coordonnées $x_1, \ldots, x_n$ (par rapport á la base déterminée) et vérifiant les m+n contraintes précédentes déterminent ce qu'on appelle un polytope convexe ou, lorsque la distance de ces points á l'origine est bornée, un polyèdre convexe appelé *polyèdre des contraintes.*

V. Convexité

Le terme convexe est justifie par la remarque suivante :

Soient $M = (x_{1, \ldots,} x_n)$ et $P = (y_1,\ldots, y_n)$ deux points quelconques du polytope (ou du polyèdre) déterminés par les contraintes ; alors, quelque soit le réel λ avec $0 \leq \lambda \leq 1$, le point $\lambda M + (1 - \lambda)P$ *(de coordonnées* $\lambda x_i + (1 - \lambda) y_i$*)* appartient au polytope (ou au polyèdre). Les n-uplets $(x_{1, \ldots,} x_n)$ qui satisfont les contraintes s'appellent *solutions réalisables* du problème, Ce sont les coordonnées des points intérieurs au polyèdre ou au polytope des contraintes, aui dans notre exemple était le quadrilatère *OABC*.

Le théorème qui suit corrobore les explications précédentes :

THEOREME. *On considère une forme linéaire des n variables* $x_{1, \ldots,} x_n$ *soumises á des contraintes linéaires. Son maximum, qui existe si cette forme est majorée, est atteint au moins en un sommet du polyèdre des contraintes.*

VI. Implémentation

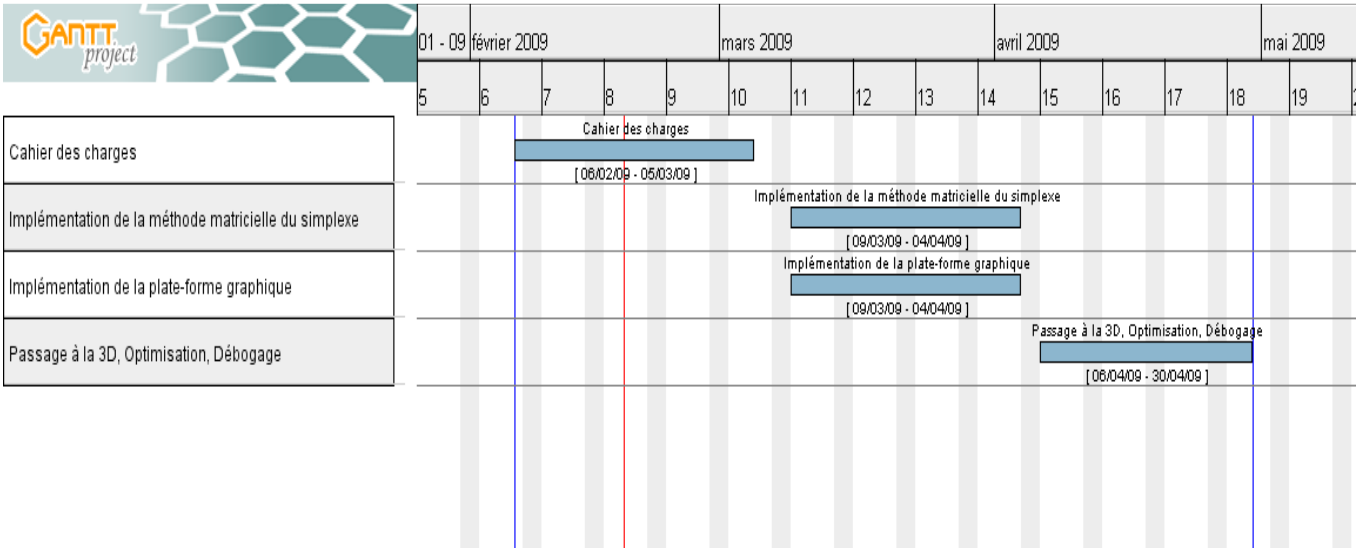
Pour l'implémentation de la méthode matricielle du simplexe, nous avons opté pour le langage C++ car il permet de garder tous les avantages du C : portabilité, possibilité d'utiliser différents niveaux d'optimisation au sein d'un même programme (objets - langage structuré classique).

VII. Conclusion

En effet, l'idée de l'algorithme du simplexe est de passer itérativement d'un sommet du polyèdre des contraintes á un sommet adjacent de façon á augmenter la valeur de la fonction á optimiser jusqu'á trouver un sommet où le maximum est atteint. C'est grâce á la convexité du polyèdre et á la linéarité de la fonction dont on cherche le maximum que l'on peut se contenter de « monter suivant les arêtes du polyèdre de sommet en sommet » pour trouver le maximum.

VIII. Organisation

Ci-dessous un diagramme de Gant illustrant l'organisation temporelle des taches à effectuer, ainsi que leur répartition sur l'ensemble des membres du groupe.

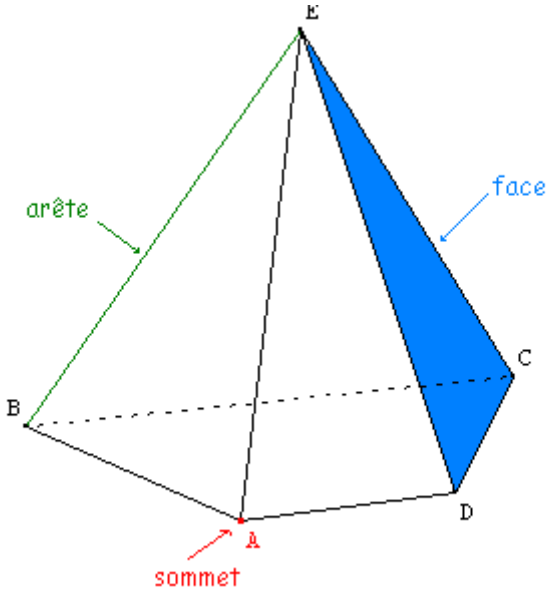


Nom	Date de début
Cahier des charges	06/02/09
Implémentation de la méthode matricielle du simplexe • Choix du langage • Implémentation de l'algorithme o Avec 2 variables o Avec 3 variables • Responsables de la parie : BENYAHIA Zakaria et LAGHA Nazim	09/03/09
Implémentation de la plate-forme graphique • Choix du software • Résolution matricielle à la main o Visualisation en statique o Etapes intermédiaires • Responsables de la partie : GALINDO Benjamin et LACOMBE Jonathan	09/03/09
Passage à la 3D, Optimisation, Débogage • Passage à la 3D • Visualisation 3D • Débogage • Optimisation • Responsables de la partie : Tous les membres du groupe	06/04/09

IX. Annexe

A. Les polyèdres

Un polyèdre est un solide de l'espace délimité par un nombre fini de polygones, tel que chaque côté de chaque polygone est commun avec un côté d'un autre polygone. Les sommets des polygones sont appelés sommets du polyèdre, les côtés des polygones sont appelés arêtes du polyèdre, tandis que les polygones sont les faces du polyèdre.



Un polyèdre est dit convexe si, pour chaque plan de l'espace qui contient une face du polyèdre, le polyèdre est tout entier dans un des demi-espaces délimité par le plan. Pour ces polyèdres convexes, on a la célèbre relation d'Euler, qui relie le nombre de faces F, le nombre de sommets S, le nombre d'arêtes A :

$$F+S=A+2$$

B. Polyèdres réguliers

Il n'existe que 5 polyèdres convexes qui sont réguliers (c'est-à-dire que leurs faces sont des polygones réguliers). On les appelle aussi les solides platoniciens. Ils sont :

- Le tétraèdre régulier : 4 faces (des triangles équilatéraux), 4 sommets, 6 arêtes.
- Le cube : 6 faces (des carrés), 8 sommets et 12 arêtes.
- L'octaèdre régulier : huit faces (des triangles équilatéraux), 6 sommets, 12 arêtes.
- Le dodécaèdre régulier : 12 faces (des pentagones réguliers), 20 sommets, 30 arêtes.
- L'icosaèdre régulier : 20 faces (des triangles équilatéraux), douze sommets, et trente arêtes.

Remarquons que sur tous ces polyèdres, la relation d'Euler est vérifiée.