

1 Intro, Optimisation, Problème Linéaire

2 Résolution d'un PL avec le simplexe

- Exemple introductif
- Formalisation
- Exercice, démo

1 Intro, Optimisation, Problème Linéaire

- Optimiser ?
- Problème Linéaire
- Un peu de maths préliminaires
- La géométrie des PL

2 Résolution d'un PL avec le simplexe

- Exemple introductif
- Formalisation
- Exercice, démo

Premier Exemple

Source : Chvatal.

Maximiser $5x_1 + 4x_2 + 3x_3$

Sous les contraintes :

$$\begin{cases} 2x_1 + 3x_2 + x_3 & \leq & 5 \\ 4x_1 + x_2 + 2x_3 & \leq & 11 \\ 3x_1 + 4x_2 + 2x_3 & \leq & 8 \\ x_1, x_2, x_3 & \geq & 0 \end{cases}$$

Premier Exemple - Slack variables

Variables « différences » :

$$\begin{cases} s_1 = 5 - 2x_1 - 3x_2 - x_3 \\ s_2 = 11 - 4x_1 - x_2 - 2x_3 \\ s_3 = 8 - 3x_1 - 4x_2 - 2x_3 \\ z = 5x_1 + 4x_2 + 3x_3 \end{cases}$$

- ▶ Minimiser z sous les contraintes $x_1, \dots, x_6 \geq 0$.
- ▶ Y-a-t-il équivalence ? Entre quoi et quoi ?

Premier Exemple - Améliorer une solution

$$\begin{cases} 5 = s_1 + 2x_1 + 3x_2 + x_3 \\ 11 = s_2 + 4x_1 + x_2 + 2x_3 \\ 8 = s_3 + 3x_1 + 4x_2 + 2x_3 \\ z = 5x_1 + 4x_2 + 3x_3 \end{cases}$$

ou mieux :

$$\begin{cases} 5 = s_1 + 2x_1 + 3x_2 + x_3 \\ 11 = s_2 + 4x_1 + x_2 + 2x_3 \\ 8 = s_3 + 3x_1 + 4x_2 + 2x_3 \\ z = 5x_1 + 4x_2 + 3x_3 \end{cases}$$

► En augmentant x_1 jusqu'à $\frac{5}{2}$ on fait tomber s_1 à 0. Pourquoi ?

Premier Exemple - Améliorer une solution - 2

$$\begin{cases} 5/2 = x_1 + 3/2x_2 + 1/2x_3 + 1/2s_1 \\ 1 = s_2 - 5x_2 - 2s_1 \\ 1/2 = s_3 - 1/2x_2 + 1/2x_3 - 3/2s_1 \\ z = 25/2 - 7/2x_2 + 1/2x_3 - 5/2s_1 \end{cases}$$

► Augmenter x_3 jusqu'à 1 fait tomber s_3 à 0.

$$\begin{cases} 1 = x_3 - x_2 - 3s_1 + 2s_3 \\ 2 = x_1 + 2x_2 + 2s_1 - s_3 \\ 1 = s_2 - 5x_2 - 2s_1 \\ z = 13 - 3x_2 - s_1 - s_3 \end{cases}$$

► Aucune variable x_2, s_1, s_3 ne peut être augmentée. FIN

► $z = 13$ optimal car ??

Mise en œuvre

1 Intro, Optimisation, Problème Linéaire

- Optimiser ?
- Problème Linéaire
- Un peu de maths préliminaires
- La géométrie des PL

2 Résolution d'un PL avec le simplexe

- Exemple introductif
- **Formalisation**
- Exercice, démo

Algorithme du simplexe :

- Introduction des variables d'écart $s_i = \dots$
- Introduction de la variable objectif $z = \dots$
- Pivots jusqu'à obtenir un **Tableau**. Les variables formant la base sont appelées **basiques**, les autres **non basiques**.

Notation usuelle : tableaux

Tableau

$$\begin{cases} 1 = & x_3 & - & x_2 & - & 3s_1 & + & 2s_3 \\ 2 = & & x_1 & + & 2x_2 & + & 2s_1 & - & s_3 \\ 1 = & & & s_2 & - & 5x_2 & - & 2s_1 \\ \hline z = & 13 & - & 3x_2 & - & s_1 & - & s_3 \end{cases}$$

► On peut immédiatement conclure.

Notation usuelle : tableaux (définitions)

Solution Basique

Le point de coordonnées $(0, \dots, 0)$ dans les variables non-basiques est appelé **solution basique** du tableau.

Feasibility

Un tableau est **réalisable** (faisable) si la solution basique $(0, \dots, 0)$ est une solution réalisable.

Étapes de l'algorithme

Bien écrit dans le cours de Nicolas Thierry :

`nicolas.thierry.name/RO/Notes/ProgrammationLineaire.html`

► À lire !

1 Intro, Optimisation, Problème Linéaire

- Optimiser ?
- Problème Linéaire
- Un peu de maths préliminaires
- La géométrie des PL

2 Résolution d'un PL avec le simplexe

- Exemple introductif
- Formalisation
- Exercice, démo

Résolution d'un PL

Résoudre à l'aide du simplexe le PL suivant :

Maximiser $3x_1 + 2x_2 + 4x_3$

Sous les contraintes :

$$\begin{cases} x_1 + x_2 + 2x_3 \leq 4 \\ 2x_1 + \quad + 3x_3 \leq 5 \\ 2x_1 + 2x_2 + 4x_3 \leq 7 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

► À la main, à l'ordi.

Résolution d'un PL, étapes

Choix du pivot à l'aide de la **règle du plus grand coefficient**

► Cf démo Maple.

Quelques bibliothèques, logiciels

Deux types d'outils :

- Outils de « calcul formel » type « calculette » : Maple, Mupad, **SAGE**...
 - Bibliothèques C, Ocaml etc : cvxopt, **lpsolve**...
- Différents encodages, en général matriciels.

DÉFINITION. Problème de programmation linéaire sous *forme standard*:
Maximiser:

$$z := \sum_{j=1}^n c_j x_j$$

Sous les contraintes:

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \text{ pour } i = 1, \dots, m$$

$$x_j \geq 0, \text{ pour } j = 1, \dots, n$$

Un choix des variables (x_1, \dots, x_n) est appelé *solution* du problème.

Une solution est *faisable* si elle vérifie les contraintes.

z est appelé *fonction objective* . À chaque solution elle associe une valeur.

Une solution est *optimale* si elle est faisable et maximise la fonction objective.

EXERCICE 1. Peut-on mettre sous forme standard les exemples précédents ?

1.1.3. Existence de solutions optimales ?

PROBLÈME 1.1.2. [1, p. 7]

On considère les trois problèmes de programmation linéaire standard suivants, écrits avec la syntaxe du système de calcul formel MuPAD:

```

Chvatal7a := [ [ x1 <= 3,
                x2 <= 7 ],
               3 +x1 +x2,
               NonNegative]:
Chvatal7b := [ [ x1 +x2 <= 2,
                -2*x1-2*x2 <= -10 ],
               3*x1 -x2,
               NonNegative]:
Chvatal7c := [ [-2*x1 +x2 <= -1,
                -x1-2*x2 <= -2 ],
               x1 -x2,
               NonNegative]:
extra      := [ [ x1 +x2 <= 1 ],
               x1 +x2,
               NonNegative]:

```

PROBLÈME 1.1.3. Déterminer pour ces trois problèmes s'il y a des solutions optimales.

CHAPTER 1

Programmation linéaire

1.1. Qu'est-ce que la programmation linéaire

1.1.1. Exemple: le problème du régime de Polly [1, p.3].

- Besoins journaliers:
Énergie: 2000 kcal
Protéines: 55g
Calcium: 800 mg
- Nourriture disponible

	Portion	Énergie (kcal)	Protéines (g)	Calcium (mg)	Prix/portion
Céréales	28g	110	4	2	3
Poulet	100g	205	32	12	24
Oeufs	2 gros	160	13	54	13
Lait entier	237cc	160	8	285	9
Tarte	170g	420	4	22	20
Porc et haricots	260g	260	14	80	19

Quels choix pour Polly ?

- Contraintes:
Céréales: au plus 4 portions par jour
Poulet: au plus 3 portions par jour
Oeufs: au plus 2 portions par jour
Lait: au plus 8 portions par jour
Tarte: au plus 2 portions par jour
Porc et haricots: au plus 2 portions par jour

PROBLÈME 1.1.1. Polly peut-elle trouver une solution ?

Comment formaliser le problème ? (modélisation)

Qu'est-ce qui fait la spécificité du problème ?

Savez-vous résoudre des problèmes similaires ?

1.1.2. Forme standard d'un problème de programmation linéaire.

PROBLÈME. [1, p. 5]

- Premier cas: une solution optimale unique

- Deuxième cas: pas de solution faisable
- Troisième cas: pas de solution optimale, car on peut faire tendre la fonction objective vers l'infini avec des solutions faisables.
- Quatrième cas: une infinité de solutions optimales.

1.2. Algorithme du simplexe

1.2.1. Premier problème.

PROBLEME 1.2.1. [1, p. 13]

$$\text{Chvatal13} := \left[\begin{array}{l} 2x_1 + 3x_2 + x_3 \leq 5, \\ 4x_1 + x_2 + 2x_3 \leq 11, \\ 3x_1 + 4x_2 + 2x_3 \leq 8, \\ 5x_1 + 4x_2 + 3x_3, \\ \text{NonNegative} \end{array} \right],$$

Solution faisable ?

Amélioration de la solution ?

Introduction de variables d'écart:

$$\begin{array}{rcl} 5 & = & s_1 + 2x_1 + 3x_2 + x_3 \\ 11 & = & s_2 + 4x_1 + x_2 + 2x_3 \\ 8 & = & s_3 + 3x_1 + 4x_2 + 2x_3 \\ \hline z & = & 5x_1 + 4x_2 + 3x_3 \end{array}$$

En augmentant x_1 jusqu'à $5/2$, on fait tomber s_1 à zéro.

On transforme le système, pour se ramener à une situation similaire à la précédente:

$$\begin{array}{rcl} 5/2 & = & x_1 + 3/2x_2 + 1/2x_3 + 1/2s_1 \\ 1 & = & s_2 - 5x_2 - 2s_1 \\ 1/2 & = & s_3 - 1/2x_2 + 1/2x_3 - 3/2s_1 \\ \hline z & = & 25/2 - 7/2x_2 + 1/2x_3 - 5/2s_1 \end{array}$$

On augmente x_3 jusqu'à 1, ce qui fait tomber s_3 à 0:

$$\begin{array}{rcl} 1 & = & x_3 - x_2 - 3s_1 + 2s_3 \\ 2 & = & x_1 + 2x_2 + 2s_1 - s_3 \\ 1 & = & s_2 - 5x_2 - 2s_1 \\ \hline z & = & 13 - 3x_2 - s_1 - s_3 \end{array}$$

Et maintenant, que fait-on ?

1.2.2. Variables d'écart.

PROBLEME 1.2.2. Est-ce que l'introduction de ces variables change le problème ?

1.2.3. Tableaux.

PROBLEME 1.2.3. [1, p. 19]

$$\text{Chvatal19} := \left[\begin{array}{l} x_1 + 3x_2 + x_3 \leq 3, \\ -x_1 + 3x_3 \leq 2, \\ 2x_1 + 3x_2 - x_3 \leq 2, \\ 2x_1 - x_2 + 2x_3 \leq 4, \\ 5x_1 + 5x_2 + 3x_3, \\ \text{NonNegative} \end{array} \right];$$

DÉFINITION. *Tableau initial:*

$$b_i = s_i + \sum_{j=1}^n a_{ij}x_j, \text{ pour } i = 1, \dots, m$$

$$z = \sum_{j=1}^n c_jx_j$$

Ou sous forme matricielle:

$$\begin{array}{rcl} B & = & S + AX \\ z & = & CX \\ X & \geq & 0 \end{array}$$

EXEMPLE. Tableau initial du problème précédent:

$$\begin{array}{rcl} 3 & = & s_1 + x_1 + 3x_2 + x_3 \\ 2 & = & s_2 - x_1 + 3x_3 \\ 2 & = & s_3 + 2x_1 + 3x_2 - x_3 \\ 4 & = & s_4 + 2x_1 - x_2 + 2x_3 \\ \hline z & = & 0 + 5x_1 + 5x_2 + 3x_3 \end{array}$$

EXEMPLE 1.2.4. On peut l'abrégé sous forme matricielle:

```
read("tableaux.mu");
linopt::Transparent(Chvatal19);
+-----+
| "linopt", "restr", slk[1], slk[2], slk[3], slk[4], x3, x1, x2 |
| "obj", 0, 0, 0, 0, 0, 3, 5, 5 |
| slk[1], 3, 1, 0, 0, 0, 1, 1, 3 |
| slk[2], 2, 0, 1, 0, 0, 3, -1, 0 |
| slk[3], 2, 0, 0, 1, 0, -1, 2, 3 |
| slk[4], 4, 0, 0, 0, 0, 1, 2, -1 |
+-----+
```

DÉFINITION. De manière générale, un *tableau* est un ensemble d'équations de la forme:

$$\begin{array}{rcl} 4 & = & x_1 + 3/2x_2 - 1/2x_3 + 1/2s_4 \\ 2 & = & s_1 + 3/2x_2 + 3/2x_3 - 1/2s_4 \\ 3 & = & s_2 + 3/2x_2 + 5/2x_3 + 1/2s_4 \\ 2 & = & s_3 - 4x_2 + 3x_3 - s_4 \\ \hline z & = & 5 - 5/2x_2 + 11/2x_3 - 5/2s_4 \end{array}$$

x_1, s_1, s_2, s_3 sont les variables *basiques*; $\{x_1, s_1, s_2, s_3\}$ est la *base*.
 x_2, x_3, s_4 sont les variables *non basiques*.

REMARQUE 1.2.5. Terminologie: on utilise dans ce cours les tableaux, plutôt que les *dictionnaires* utilisés par exemple dans [1]. La différence est minime: on fait juste passer les variables non basiques d'un côté ou de l'autre des équations. D'autre part, on utilise s_1, s_2, s_3, s_4 plutôt que x_4, x_5, x_6, x_7 comme noms pour les variables d'écart.

Voici le dictionnaire correspondant au tableau précédent:

$$\begin{array}{r} x1 = 1 - 3/2 x2 + 1/2 x3 - 1/2 x7 \\ x4 = 2 - 3/2 x2 - 3/2 x3 + 1/2 x7 \\ x5 = 3 - 3/2 x2 - 5/2 x3 - 1/2 x7 \\ x6 = 2 + 4 x2 - 3 x3 + x7 \\ \hline z = 5 - 5/2 x2 + 11/2 x3 - 5/2 x7 \end{array}$$

REMARQUE 1.2.6. La caractéristique essentielle d'un tableau est que, connaissant les variables non-basiques, on peut immédiatement calculer les variables basiques et la fonction objective (d'où le terme de *dictionnaire*). Le calcul devient même immédiat si toutes les variables non-basiques sont nulles.

Les équations d'un tableau décrivent un sous-espace affine E de \mathbb{R}^{n+m} .

Un point p de cet espace est caractérisé par ses coordonnées dans les variables non-basiques.

EXERCICE 2. Calculer directement le tableau correspondant aux variables non-basiques x_1, s_2, s_3 du programme linéaire Chvatal13.

EXERCICE 3. Soit t_1 et t_2 deux tableaux correspondant au même programme linéaire.

Que peut-on dire des deux sous-espaces affine de \mathbb{R}^{n+m} qu'ils définissent ?

Chaque choix de variables non-basiques correspond à une base affine de ce sous-espace.

DÉFINITION 1.2.7. Le point de coordonnées $(0, \dots, 0)$ dans les variables non-basiques est appelé *solution basique* du tableau.

Un tableau est *faisable* si la solution basique $(0, \dots, 0)$ est une solution faisable.

De manière équivalente, un tableau est faisable si les constantes dans les équations du haut sont toutes positives ou nulles.

Revenons à l'exemple [1, p. 19]:

```
read("tableaux.mu");
t:=linopt::Transparent(Chvatal19);
t:=linopt::Transparent::userstep(t, slk[3], x3);
```

EXERCICE 4. [1, 2.1 p. 26]

Utilisez l'algorithme du simplexe pour résoudre les programmes linéaires suivants:

```
Chvatal26_21a :=
[[ x1 +x2+2*x3 <= 4,
 2*x1 +3*x3 <= 5,
```

```
2*x1 +x2+3*x3 <= 7],
3*x1+2*x2+4*x3,
NonNegative]:
Chvatal26_21c :=
[[2*x1+3*x2 <= 3,
 x1+5*x2 <= 1,
 2*x1 +x2 <= 4,
 4*x1 +x2 <= 5],
 2*x1 +x2,
NonNegative]:
```

EXERCICE 5. Essayez d'appliquer l'algorithme du simplexe aux programmes linéaires de l'exercice [1, p. 7] (cf. ci-dessus). Que se passe-t-il ?

1.3. Pièges et comment les éviter

1.3.1. Bilan des épisodes précédents. On a un algorithme qui marche sur quelques exemples.

Il faut vérifier trois points pour savoir s'il marche en général:

- (1) Initialisation
- (2) Itération
- (3) Terminaison

1.3.2. Itération.

PROPOSITION. *Étant donné un tableau faisable, on peut toujours effectuer l'une des opérations suivantes:*

- (1) Conclure que le système a une solution optimale unique, la calculer et la certifier;
- (2) Conclure que le système a une infinité de solutions optimales, les calculer et les certifier;
- (3) Conclure que le système est non borné, et le certifier en décrivant une demi-droite de solutions sur laquelle z prend des valeurs aussi grandes que voulu.
- (4) Trouver une variable entrante, une variable sortante, et effectuer un pivot. Par construction, le tableau obtenu est équivalent au tableau précédent, et est encore faisable. De plus, z a *augmenté au sens large* (i.e. la constante z^* dans la nouvelle expression de z est supérieure ou égale à l'ancienne).

PROOF. Il suffit d'analyser le tableau faisable. Notons S_1, \dots, S_m les variables basiques, X_1, \dots, X_n les variables non-basiques, et C_1, \dots, C_n, z^* les coefficients tels que $z = z^* + \sum C_i X_i$. Par exemple, dans le tableau final du problème 1.2.1, on a $X_1 = x_2, X_2 = s_1, X_3 = s_2, S_1 = x_1, S_2 = x_3, S_3 = s_3, C_1 = -3, C_2 = -1, C_3 = -1$ et $z^* = 13$.

- (1) Si $C_i < 0$, pour tout i , alors la solution basique du tableau, de coordonnées $X_1^* = \dots = X_n^* = 0$ est l'unique solution optimale. Vérifiez-le en prouvant qu'une toute solution faisable quelconque de coordonnées X_1, \dots, X_n donnant la même valeur $z = z^*$ à la fonction objective est égale à la solution basique du tableau.
- (2) Si $C_i \leq 0$ pour tout i , la solution basique du tableau est optimale, et l'ensemble des solutions optimales est décrit par les inéquations linéaires du système et l'annulation des variables non-basiques X_i pour lesquelles on a $C_i < 0$. Les détails sont similaires au 1.
- (3) Sinon, on peut prendre X_i , variable non-basique avec un coefficient $C_i > 0$. Si les équations du tableau n'imposent pas de limite sur X_i , le système est non borné: la demi-droite décrite par $(0, \dots, 0, X_i, 0, \dots, 0)$ pour $X_i \geq 0$ est composée de solutions faisables qui donnent des valeurs aussi grandes que voulu à z .

(4) Autrement, une des variables basiques S_j tombe à zéro, et on peut faire un pivot entre la variable entrante X_i et la variable sortante S_j . Par construction, la nouvelle solution basique correspond à une solution faisable $(0, \dots, 0, X_i, 0, \dots, 0)$ pour un $X_i \geq 0$. En particulier le nouveau tableau est faisable, et comme $C_i \geq 0$, la constante z^* a augmenté au sens large. \square

EXEMPLE. [1, p. 29] Système où z n'augmente pas strictement lors du pivot:

```
Chvatal29 := [[
                2*x3 <= 1,
               - x1 + 3*x2 + 4*x3 <= 2,
               2*x1 - 4*x2 + 6*x3 <= 3],
             NonNegative];
t0:= linopt::Transparent(Chvatal29);
t1:= linopt::Transparent::userstep(t0, slk[1], x3);
t2:= linopt::Transparent::userstep(t1, slk[3], x1);
t3:= linopt::Transparent::userstep(t2, slk[2], x2);
t4:= linopt::Transparent::userstep(t3, x3, slk[1]);
```

REMARQUE. Lorsque z n'augmente pas, on est forcément dans une situation de dégénérescence: le pivot change le tableau, mais pas la solution basique décrite par le tableau.

1.3.3. Terminaison.

PROBLÈME 1.3.1. Peut-on garantir que l'algorithme va finir par s'arrêter ?

THÉORÈME. Si l'algorithme du simplexe ne cycle pas, il termine en au plus $C(n+m, m)$ itérations.

PROOF. (Résumé)

Chaque itération correspond à un tableau faisable.

Un tableau faisable est entièrement caractérisé par le choix des variables basiques.

Il n'y a que $C(n+m, m)$ choix possibles de variables basiques. \square

REMARQUE. L'algorithme ne peut cycler qu'en présence de dégénérescence.

Avec une stratégie incorrecte, l'algorithme du simplexe peut cycler éternellement.

EXEMPLE. [1, p. 31] Système cyclant en 6 itérations avec la stratégie:

- Choix de la variable entrante avec le coefficient dans l'expression de z le plus fort
- Choix de la variable sortante avec le plus petit index

```
Chvatal31 := [[0.5*x1 - 5.5*x2 - 2.5*x3 + 9*x4 <= 0,
               0.5*x1 - 1.5*x2 - 0.5*x3 + x4 <= 0,
               x1 <= 1],
             NonNegative];
t0 := linopt::Transparent(Chvatal31);
```

```
t1 := linopt::Transparent::userstep(t0, slk[1], x1);
t2 := linopt::Transparent::userstep(t1, slk[2], x2);
t3 := linopt::Transparent::userstep(t2, x1, x3);
t4 := linopt::Transparent::userstep(t3, x2, x4);
t5 := linopt::Transparent::userstep(t4, x3, slk[1]);
t6 := linopt::Transparent::userstep(t5, x4, slk[2]);
```

Comment garantir que l'algorithme ne cyclera pas ?

1.3.3.1. *La méthode des perturbations.* L'algorithme du simplexe ne peut cycler qu'en présence de dégénérescence.

PROBLÈME 1.3.2. Comment se débarrasser des dégénérescences ?

Idée: supprimer les dégénérescences en perturbant légèrement le système!

EXEMPLE. [1, p. 34,35]

On introduit des constantes $\varepsilon_1 \gg \dots \gg \varepsilon_n$.

Inconvénient: solution approchée, ou introduction de calcul symbolique

1.3.3.2. *La méthode du plus petit index.*

THÉORÈME. L'algorithme du simplexe termine si, lorsqu'il y a ambiguïté sur le choix de la variable entrante ou sortante, on choisit toujours la variable de plus petit index.

Cette méthode est simple et élégante.

Par contre, elle empêche toute stratégie pour faire converger l'algorithme plus vite.

1.3.3.3. *Méthodes intermédiaires.* Stratégie au choix, mais si z n'augmente pas pendant plus d'un certain nombre d'itérations, on bascule sur la stratégie du plus petit index jusqu'à ce que l'on soit sorti de la dégénérescence.

1.3.4. **Initialisation.** Pour le moment, l'algorithme du simplexe nécessite de partir d'un tableau faisable.

PROBLÈME. Dans le cas général, comment se ramener à un tableau faisable?

Le système pourrait même ne pas avoir de solution!

EXEMPLE. [1, p. 39] Système P_1 :

Maximiser: $x_1 - x_2 + x_3$

Sous les contraintes:

$$2x_1 - x_2 + 2x_3 \leq 4$$

$$2x_1 - 3x_2 + x_3 \leq -5$$

$$-x_1 + x_2 - 2x_3 \leq -1$$

$$x_1, x_2, x_3 \geq 0$$

EXEMPLE. Introduction d'un système auxiliaire P_0 pour déterminer si P est faisable:

Maximiser: $-x_0$

Sous les contraintes:

$$2x_1 - x_2 + 2x_3 - x_0 \leq 4$$

$$2x_1 - 3x_2 + x_3 - x_0 \leq -5$$

$$-x_1 + x_2 - 2x_3 - x_0 \leq -1$$

$$x_0, x_1, x_2, x_3 \geq 0$$

Remarques:

- P_0 est faisable (prendre x_0 suffisamment grand);
- Les solutions faisables de P correspondent aux solutions faisables de P_0 avec $x_0 = 0$;
- P est faisable si et seulement si P_0 a une solution faisable avec $x_0 = 0$.

Étudions ce nouveau système:

```
Chvatal40 := [[ -x1 + x2 - 2*x3 - x0 <= -1,
                2*x1 - 3*x2 + x3 - x0 <= -5,
                2*x1 - x2 + 2*x3 - x0 <= 4],
               -x0,
               NonNegative]:
t0:=linopt::Transparent(Chvatal40);
t1:=linopt::Transparent::userstep(t0, slk[2], x0);
t2:=linopt::Transparent::userstep(t1, slk[1], x2);
t3:=linopt::Transparent::userstep(t2, x0, x3);
```

Maintenant, nous savons que le système P est faisable.

En fait, en éliminant x_0 on obtient même un tableau faisable pour P !

Algorithme du simplexe en deux phases pour résoudre un problème P sous forme standard:

Phase I:

- (1) Si $(0, \dots, 0)$ est solution faisable de P , on passe directement à la phase II.
- (2) Définir un problème auxiliaire P_0 .
- (3) Le premier tableau pour P_0 est infaisable.
- (4) Le rendre faisable par un pivot approprié de x_0 .
- (5) Appliquer le simplexe habituel:
 - (a) Si à une étape donnée, x_0 peut sortir de la base, le faire en priorité:

En effet, il y a une solution faisable avec $x_0 = 0$, et on peut passer en phase II.
 - (b) Si à une étape donnée on atteint une solution optimale:
 - (i) Si x_0 n'est pas basique:

Il y a une solution faisable avec $x_0 = 0$. On peut donc passer en phase II.
 - (ii) Si x_0 est basique et $z_0 < 0$:

P est infaisable, et on s'arrête.
 - (iii) Sinon x_0 est basique et $z_0 = 0$:

Situation impossible si on fait toujours sortir x_0 en priorité de la base.

- (6) Tirer de P_0 un tableau faisable pour P ;

Phase II:

- (1) Appliquer le simplexe habituel à partir du tableau donné par P_0 .

EXERCICE. [1, ex 3.9a p. 44]

Maximiser $3x_1 + x_2$

Sous les contraintes:

$$x_1 - x_2 \leq -1$$

$$-x_1 - x_2 \leq -3$$

$$2x_1 + x_2 \leq 4$$

$$x_1, x_2 \geq 0$$

```
t0:=linopt::Transparent(Chvatal44_39a0)
t1:=linopt::Transparent::userstep(t0, slk[2], x0)
t2:=linopt::Transparent::userstep(t1, slk[1], x1)
t3:=linopt::Transparent::userstep(t2, x0, x2)
t0:=linopt::Transparent(Chvatal44_39a)
t1:=linopt::Transparent::userstep(t0, slk[1], x1)
t2:=linopt::Transparent::userstep(t1, slk[2], x2)
t3:=linopt::Transparent::userstep(t2, slk[3], slk[2])
```

1.3.5. Le théorème fondamental de la programmation linéaire.

THÉORÈME. *Tout programme linéaire P sous forme standard a l'une des propriétés suivantes:*

- (1) Si P n'a pas de solutions optimales, alors P est infaisable ou non borné;
- (2) Si P a une solutions faisable, alors P a une solution basique faisable;
- (3) Si P a une solution optimale, alors P a une solution basique optimale.

1.4. Efficacité de l'algorithme du simplexe

Pour une discussion complète sur ce thème, nous renvoyons au livre de référence [1, 4. How fast is the simplex method], ainsi qu'à l'excellente Foire Aux Questions <http://rutcor.rutgers.edu/~mnk/1p-faq.html> pour les évolutions récentes.

EXERCICE. [1, 4.2 et 4.3, p. 53]

1.5. Le théorème de dualité

1.5.1. Motivation: estimer la valeur optimale de la fonction objective.

EXEMPLE. Maximiser: $z = 4x_1 + x_2 + 5x_3 + 3x_4$

Sous les contraintes:

$$x_1 - x_2 - x_3 + 3x_4 \leq 1$$

$$5x_1 + x_2 + 3x_3 + 8x_4 \leq 55$$

$$-x_1 + 2x_2 + 3x_3 - 5x_4 \leq 3$$

$$x_1, x_2, x_3, x_4 \geq 0$$

PROBLÈME. Borne inférieure sur la valeur optimale z^* ?

Borne supérieure sur la valeur optimale z^* ?

Le simplexe avec Maple 9.5

```
> ?simplex;with(simplex):
```

Warning, the protected names maximize and minimize have been redefined and unprotected

[La doc, indispensable...]

L'exemple du cours

```
> initpb:={x1+x2+2*x3 <= 4, 2*x1+3*x3<=5,2*x1+x2+3*x3<=7};
initpb:= {x1 + x2 + 2 x3 ≤ 4, 2 x1 + 3 x3 ≤ 5, 2 x1 + x2 + 3 x3 ≤ 7}
```

```
> simplexb:=setup( initpb );
simplexb:= {_SL1 = 4 - x1 - x2 - 2 x3, _SL2 = 5 - 2 x1 - 3 x3,
_SL3 = 7 - 2 x1 - x2 - 3 x3}
```

```
> display(initpb, [x1,x2,x3]);
```

$$\begin{bmatrix} 1 & 1 & 2 \\ 2 & 0 & 3 \\ 2 & 1 & 3 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} \leq \begin{bmatrix} 4 \\ 5 \\ 7 \end{bmatrix}$$

```
> feasible(simplexb, NONNEGATIVE);
true
```

```
> obj:=3*x1+2*x2+4*x3;
obj:= 3 x1 + 2 x2 + 4 x3
```

```
> pivotvar(obj);
x1
```

```
> injection:=pivoteqn(simplexb, x3);
injection:= [_SL2 = 5 - 2 x1 - 3 x3]
```

The function pivoteqn(C, var) returns a sublist of equations from C. Each equation eq returned achieves the minimum non-negative ratio of
(- cterm(eq)) / coeff(eq, var, 1)

```
> ratio(simplexb, x3);
{ 2, 5/3, 7/3 }
```

The call ratio(C, x) returns a list of ratios which can be used to determine which equation in C is most restrictive with respect to the next simplex pivot. For each equation, the ratio computed, if possible, is
- cterm(eq) / coeff(eq, x, 1)

```
> step1a:=pivot(simplexb, x3, injection );
step1a:= { _SL1 = 2/3 + 1/3 x1 - x2 + 2/3 _SL2, x3 = -1/3 _SL2 + 5/3 - 2/3 x1,
_SL3 = 2 - x2 + _SL2 }
```

The call pivot(C, x, eqn) constructs a new set of equations in a form compatible with the forms used in simplex[setup], by solving the specified equation eqn for x, then substituting the result into C. This is equivalent to the standard simplex pivot on an array of coefficients.

To re-express the objective function f in terms of this new description, one can substitute the result of pivot into f.

```
> obj2:=subs(step1a, obj);
obj2:= 1/3 x1 + 2 x2 - 4/3 _SL2 + 20/3
```

```
> pivotvar(obj2);
x1
```

[je prends quand meme x2]

```
> injection2:=pivoteqn(step1a, x2);
injection2:= [ _SL1 = 2/3 + 1/3 x1 - x2 + 2/3 _SL2 ]
```

```
> ratio(step1a, x2);
{ 2, ∞, 2/3 }
```

```
> step2a:=pivot(step1a, x2, injection2 );
step2a:= { _SL3 = 4/3 + _SL1 - 1/3 x1 + 1/3 _SL2, x3 = -1/3 _SL2 + 5/3 - 2/3 x1,
x2 = -_SL1 + 2/3 + 1/3 x1 + 2/3 _SL2 }
```

```
> obj3:=subs(step2a, obj2);
obj3:= x1 - 2 _SL1 + 8
```

```
> injection3:=pivoteqn(step2a, x1);
injection3:= [ x3 = -1/3 _SL2 + 5/3 - 2/3 x1 ]
```

```
> ratio(step2a,x1);
```

$$\left\{ 4, \infty, \frac{5}{2} \right\}$$

```
> step3a:=pivot(step2a,x1,injection3 );
```

$$\text{step3a} := \left\{ \begin{array}{l} x1 = -\frac{3}{2}x3 - \frac{1}{2}_{SL2} + \frac{5}{2}, \quad _{SL3} = \frac{1}{2} + _{SL1} + \frac{1}{2}x3 + \frac{1}{2}_{SL2}, \\ x2 = -_{SL1} + \frac{3}{2} - \frac{1}{2}x3 + \frac{1}{2}_{SL2} \end{array} \right\}$$

```
> obj4:=subs(step3a,obj3);
```

$$\text{obj4} := -\frac{3}{2}x3 - \frac{1}{2}_{SL2} + \frac{21}{2} - 2_{SL1}$$

[donc le max est 21/2 et x3=s2=s1=0, on en déduit x1 et x2...

Td2a, pb 1

```
> maximize(5*x1+6*x2+9*x3+8*x4, {x1+2*x2+2*x3+x4<=5, x1+x2+2*x3+3*x4<=3}, NONNEGATIVE);
```

$$\{x3 = 0, x4 = 0, x2 = 2, x1 = 1\}$$

Td2a, pb 2

```
> maximize(2*x1+x2, {2*x1+3*x2<=3, x1+2*x2<=1, 2*x1+x2<=4, 4*x1+x2<=5}, NONNEGATIVE);
```

$$\{x2 = 0, x1 = 1\}$$

Td2a, pb 3

```
> maximize(3*x1+5*x2+8*x3, {x1+x2+x3<=100, 3*x1+2*x2+4*x3<=200, x1+2*x2<=15}, NONNEGATIVE);
```

$$\left\{ x2 = 75, x1 = 0, x3 = \frac{25}{2} \right\}$$

LIF062- Optim Partie A
TD2a - Simplexe et utilisation de bibliothèques

1 Simplexe à la main

EXERCICE 1 Résoudre les simplexes suivants à l'aide de la méthode du simplexe (source : Chvatal)

1. Maximiser $5x_1 + 6x_2 + 9x_3 + 8x_4$

Sous les contraintes :

$$\begin{cases} x_1 + 2x_2 + 2x_3 + x_4 \leq 5 \\ x_1 + x_2 + 2x_3 + 3x_4 \leq 3 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

2. Maximiser $2x_1 + x_2$

Sous les contraintes :

$$\begin{cases} 2x_1 + 3x_2 \leq 3 \\ x_1 + 2x_2 \leq 1 \\ 2x_1 + x_2 \leq 4 \\ 4x_1 + x_2 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

3. (cf univ Nice)

Maximiser $3x_1 + 5x_2 + 8x_3$

Sous les contraintes :

$$\begin{cases} x_1 + x_2 + x_3 \leq 100 \\ 3x_1 + 2x_2 + 4x_3 \leq 200 \\ x_1 + 2x_2 \leq 150 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

2 Utilisation de SAGE - « calculette »

Vous pouvez vous créer un compte SAGE ici : <https://eole.univ-lyon1.fr:8000/>

EXERCICE 2 À l'aide des exemples de CVXOPT :

<http://abel.ee.ucla.edu/cvxopt/documentation/users-guide/node58.html>

retrouver les résultats obtenus à la main avec SAGE+CVXOPT :

<http://www.sagemath.org>,

3 Utilisation de LPSOLVE - bibliothèque

EXERCICE 3 Utiliser la librairie *LpSolve* (après l'avoir installée), et sa documentation <http://lpsolve.sourceforge.net/5.5/>¹ pour résoudre dans le langage de votre choix les problèmes linéaires précédents.

¹chercher « Formulation of an lp problem in lpsolve »